Anything but history, for history must be false.

— Sir Robert Walpole

[Remark to his son, who offered to read to him. *Walpoliana*, Vol. i.]

Chapter 2

Background and Related Work

The purpose of this chapter is to introduce the reader to related work in reliable transport protocols and reliable data delivery over wireless networks. We start with a description of the best-effort Internet service model, motivate the need for reliable transport, and discuss several reliable transport protocols proposed in the literature. In Section 2.2, we discuss the Transmission Control Protocol, TCP [116], the dominant reliable transport protocol in the Internet today. We follow this with a discussion of a number of recently proposed end-to-end enhancements to TCP in Section 2.3. In Section 2.4, we survey some key concepts in wireless networks. In Section 2.5, we discuss conventional approaches to tackling the problems caused by wireless bit-errors, highlighting their weaknesses. Sections 2.6 and 2.7 describe related work in transport over bandwidth-asymmetric and packet radio networks respectively. We conclude this chapter with a summary in Section 2.8.

2.1 Reliable Transport Protocols

Today's Internet is a *best-effort* network that does not guarantee reliable or ordered data delivery. Packets sent from an end-host are forwarded through a sequence of routers until they reach their destination, but might get lost, reordered or duplicated en route. While this permits a simple and robust internal architecture and is responsible in part for the impressive scaling properties of backbone routers in the Internet, it leaves reliable data delivery to higher level protocols. Applications like the World Wide Web [18], file transfer [119], remote terminals [118] and electronic mail [117] that need *reliable* and *ordered* data delivery require a transport protocol to provide this functionality, freeing them of the need to achieve reliability on a per-application basis. Any reliable transport protocol must perform the following important functions:

1. *Loss recovery:* The major function of these protocols is to provide reliability and recover from lost packets. Most reliable transport protocols use some form of *Acknowledgment-Repeat-reQuest (ARQ)* retransmission scheme to recover from packet loss. Here, the receiver sends acknowledgments (ACKs) signifying successful receipt of data packets. The sender uses this information to deduce which packets are missing and retransmits them. There are three basic kinds of ARQ schemes. In *Stop-and-wait* ARQ, the transmitter sends a packet and waits for an ACK from the receiver. Upon receiving an ACK, it transmits the next packet. If the receiver sends a negative ACK (NAK) or if the transmitter times out waiting for the ACK, a retransmission occurs. While it is simple to implement, stop-and-wait is inefficient because its throughput is bounded by one packet per round-trip. *Go-Back-N (GBN)* protocols continually send data and receive ACKs spaced by a round-trip time. When a NAK arrives or a timeout occurs, the sender retransmits all packets sent thus far starting from the first missing one. The receiver implementation in these protocols is simple because no data buffering is required, but the protocol is inefficient because of the redundant retransmissions of already-received packets. *Selective repeat* protocols address this problem by having the receiver specify exactly which packets are missing (or conversely, which packets have arrived), so the sender can retransmit only the missing ones. These protocols are the most efficient among the three basic schemes, but are also the most complex to implement.

2. *Congestion and flow control:* Congestion control is the ability for the transmitter to recognize that the network is overloaded and respond to it by reducing its transmission rate. Congestion occurs when queues in the routers in the network are not empty; additionally, when a packet enters a full (or sometimes, non-empty) queue, packets are dropped. Flow control is the ability for the sender to recognize that it is overwhelming the receiver by sending data too rapidly and react to it. Continually overrunning the network or receiver buffers is not a stable situation because it leads to excessive losses, followed by retransmissions.

3. *Connection management:* Connection management involves the management of connection state and the machinery to initiate and terminate connections in the presence of packet loss, packet duplication, and router/host failures.

Several reliable transport protocols have been proposed in the literature for best-effort networks: Delta-t [144], NETBLT [32], VMTP [27], OSI/TP4 [105], XTP [137] and TCP [116]. In all these protocols, data items are identified by sequence numbers that are either byte-based or packet-based. These sequence numbers are used to detect losses, reordering and data duplication. Packet-based sequence numbers are simpler to implement but are less flexible because they often constrain the sender to use fixed-size packets. Byte-based sequence numbers indicate to the receiver the exact amount of missing data and permit the sender to precisely identify and retransmit lost data.

The rest of this section briefly surveys the key ideas in these protocols. Other than TCP, these are hardly used in the Internet today [141].

2.1.1 Delta-t

Delta-t [144] provides an end-to-end, stream-oriented, reliable transport service. Delta-t pioneered the concept of timer-based connection management and light-weight connections that are automatically established upon receipt of data and torn down after a timeout following the end of data transmission. Data is identified using byte-based sequence numbers and the receiver sends positive and optional negative ACKs signifying losses to the sender. In response, the sender implements a GBN retransmission strategy, retransmitting all missing data from the point of indicated loss.

2.1.2 NETBLT

The NETwork BLock Transfer protocol was developed for high-throughput bulk data transfer [32], especially over long-delay links. Connections using NETBLT are uni-directional and the unit of transmission is a *buffer,* several of which may be pipelined in transit as permitted by a flow control window. Transmissions are *rate-controlled,* where the number of buffers sent in unit time is controlled. NAKs and selective retransmissions are used for loss recovery, and a timeout mechanism causes all unacknowledged buffers to be retransmitted as a single block. Thus, while NETBLT has an efficient loss recovery mechanism, its weakness is the absence of a specified *adaptive* congestion control scheme to set its rate parameters in response to network congestion. Furthermore, it is optimized for large transfers and does not perform well for short ones.

2.1.3 VMTP

The Versatile Message Transport Protocol (VMTP) was originally developed to provide communication for a networked distributed system [27]. The main focus of VMTP is on transaction-oriented, RPC-style communication that requires low latency for small data transfers. VMTP also has a streaming mode with rate control and selective repeat [28]. The other innovative aspect of VMTP was its support for multicast and call forwarding functions. However, it does not have an adaptive congestion control algorithm.

2.1.4 XTP

The eXpress Transport Protocol (XTP) was designed to operate efficiently over high-speed networks, with ease of VLSI implementation as the main objective [137]. It provides a variety of different services, including reliable bulk transfer, with implicit connection establishment/teardown, rate control, and selective retransmissions. It attempts to adapt its rate and parameters to the capabilities of the network and end-hosts.

2.1.5 OSI/TP4

A unique feature of OSI's TP4 [105] transport protocol is the ability to negotiate certain quality-of-service parameters such as expected throughput, loss rate, and delay. However, retransmissions are not data-driven; instead, all retransmissions are timer-driven, making loss recovery sluggish. OSI/TP4 also allowed an end-to-end connection to be split into many cascaded network connections with hop-by-hop reliability. It is unclear that this is a desirable property, since it adds enormous complexity in the internals of the network, but does not alter the need to perform end-to-end reliability. Viewed in terms of the end-to-end argument in systems design [127], this design choice is inappropriate since it does not usually lead to significant improvement in end-to-end performance.

An excellent survey of transport protocols for high-speed networks by Doeringer et al. appears in [40].

2.2 Transmission Control Protocol (TCP)

TCP is the de facto standard for reliable data transport in the Internet today. Measurements made in 1997 of the MCI backbone show that over 95% of all bytes, 90% of all packets and 75% of all flows use TCP [141]. This section discusses its salient features and highlights its main weaknesses.

While the original formal specification of TCP is in RFC 793 [116], numerous variants of it have been developed over the past several years, such as Tahoe, Reno, Vegas, etc. [135, 21]. This section discusses the Reno variant of TCP[1], which is the predominantly deployed version today. Section 2.3 discusses a number of recently proposed enhancements to TCP, such as Selective Acknowledgments.

2.2.1 Cumulative Acknowledgments

TCP is an ARQ-based reliable transport protocol that uses cumulative ACKs and byte-based sequence numbers for reliability. TCP provides a *fully reliable, in-order, byte-stream* delivery abstraction to the higher-layer application, which typically uses a socket interface [134] to interface with the transport layer. The basic unit of transmission is called a *segment,* which is a contiguous sequence of bytes identified by its 32-bit long start and end sequence numbers. The transmitted segments are smaller than or equal to the connection's maximum segment size (MSS), which is negotiated at the start of the connection.

A cumulative ACK from a receiver for byte k implies that all bytes less than k have been successfully received, and that a segment beginning at byte k has not yet arrived. During normal loss-free operation, the cumulative ACK sequence steadily increases as segments arrive in sequence. In what follows, we describe how TCP performs loss recovery, congestion control, and connection management, and highlight weaknesses in its congestion control scheme.

2.2.2 Loss Recovery

When the TCP sender discovers that data has been lost in the network, it recovers from it by retransmitting the missing segments. TCP has two mechanisms for discovering and recovering from losses: *timer-driven retransmissions* and *data-driven retransmissions*.

---

1. Unless explicitly mentioned otherwise, "TCP" in this section implies the popular Reno variant.

Timer-driven recovery: When the TCP sender does not receive a positive cumulative ACK for a segment within a certain timeout interval, it retransmits the missing data. To determine the timeout interval, it maintains a running estimate of the connection's round-trip time using an exponential weighted moving average (EWMA) formula, $srtt = \alpha * rtt + (1-\alpha) * srtt$, where *srtt* is the smoothed round-trip time average, *rtt* the current round-trip sample, and α the EWMA constant set to 0.125 in the TCP specification. It also estimates the mean linear deviation, *rttvar,* using a similar EWMA filter, with α set to 0.25. A timeout occurs if the sender does not receive an ACK for a segment within $srtt + 4 * rttvar$ since the arrival of the last new cumulative ACK[1]. Furthermore, the retransmission timer is exponentially backed off after each unsuccessful retransmission. The details of the round-trip time calculations and timer management can be found in [71, 59, 135].

It is important to use the mean deviation in the timeout calculation to reduce the number of spurious retransmissions for data that are still in the network. Timer-driven retransmissions are coarse-grained and conservative, typically with a 500 ms timer granularity, which keep the connection idle for long periods of time. Ideally, they should be used only when losses happen because of *persistent* congestion, allowing the state of congestion to dissipate before the retransmission. For reasons of efficiency and performance, a reliable transport protocol should not rely on timeouts as the primary mode of recovery; rather, timers should be used conservatively and only as a last resort when other methods fail, since it is fundamentally impossible for retransmission timers to be accurate [150].

Data-driven recovery: TCP's data-driven retransmission mechanism uses a technique called *Fast Retransmission*. It relies on the information conveyed by cumulative ACKs and takes advantage of the receipt of later data segments after a lost one. Because ACKs are cumulative, all segments after a missing one generate *duplicate* cumulative ACKs that are sent to the TCP sender. The sender uses these duplicate ACKs to deduce that a segment is missing and retransmits it, as shown in Figure 2.1.

---

1. Thus, a timeout occurs roughly $2 * srtt + 4 * rttvar$ after the original transmission in the steady state, since a window's worth of ACKs arrive in one round-trip.

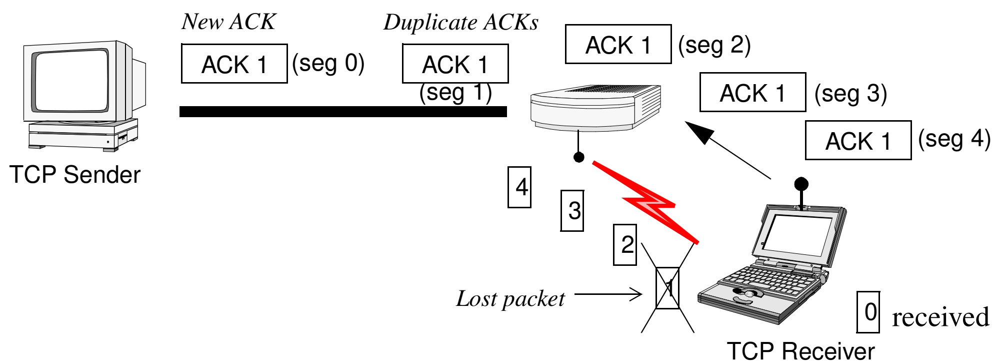

Figure 2.1 TCP Fast Retransmissions. When the receiver gets packet 0, it sends a new ACK. Then, when packets 1 through 4 arrive in succession, the cumulative ACK it sends remains at 1, which is the next packet it expects. Upon receiving three of these *duplicate* cumulative ACKs for 1, the sender retransmits packet 1.

However, the sender must not retransmit a segment upon the arrival of the very first duplicate ACK. This is because the Internet service model does not preclude the reordering of packets in the network; such reordering causes later segments to be received ahead of earlier ones, and triggers duplicate ACKs in the same way that losses do. Furthermore, the degree of packet reordering on the Internet seems to be increasing, according to studies by Paxson [111]. Thus, to avoid prematurely retransmitting segments, the sender waits for three duplicate ACKs, the current standard fast retransmit threshold.

This is followed by the *fast recovery* phase, where additional packets are transmitted after the sender is sure that at least half the current window has reached the receiver, based on a count of the number of received duplicate ACKs. Fast recovery ensures that a fast retransmission is followed by congestion avoidance and not by slow start. Since the arrival of duplicate ACKs signals to the sender that data is indeed flowing between the two ends, there is no reason to suddenly throttle the sender by invoking slow start.

Fast retransmissions are often not sufficient to recover from multiple losses in a single window, because all the cumulative duplicate ACKs arrive for the first loss in the window. This usually results in a coarse-grained timeout before the packet is retransmitted.

TCP currently makes the tacit assumption that *all* losses occur because of network congestion. Thus, coupled with either of these modes of retransmission is congestion control that reduces the

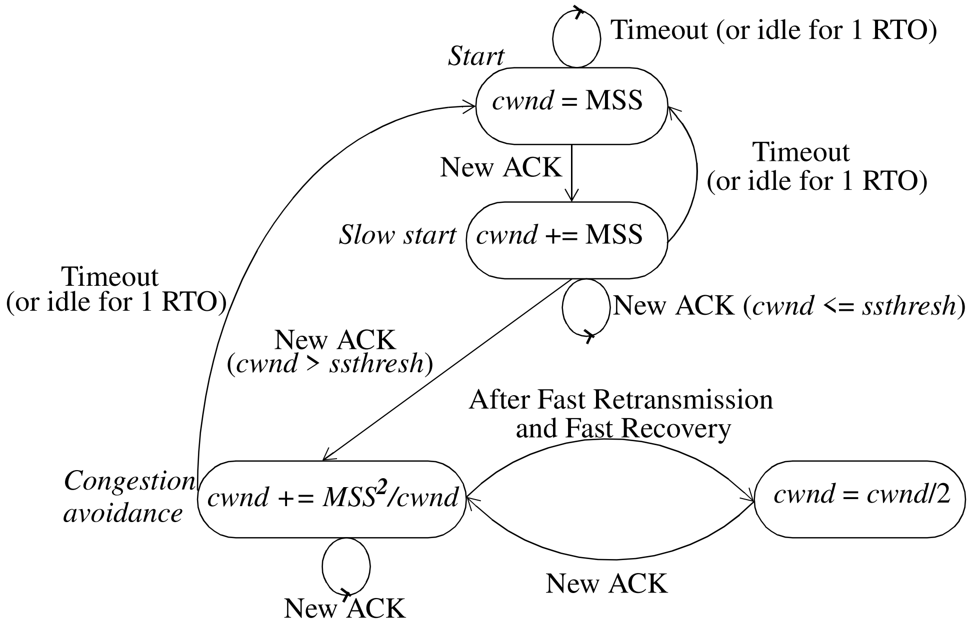

Figure 2.2 Dynamic evolution of TCP's congestion window. RTO is the retransmission timeout duration equal to *srtt + 4*rttvar.*

sender's transmission rate. The next section discusses how TCP manages congestion by probing for sustainable bandwidth and reacting to congestion.

2.2.3 Congestion Avoidance and Control

TCP's congestion management is based largely on Jacobson's seminal paper [59] and on the principles of multiplicative-decrease/additive-increase expounded by Chiu and Jain [30]. TCP uses a window-based algorithm to manage congestion, where the window is an estimate of the number of bytes currently unacknowledged and outstanding in the network.

The TCP sender performs *flow control* by ensuring that the transmission window does not exceed the receiver's advertised window size. It performs congestion control by using a window-based scheme, where the sender regulates the amount of transmitted data using a congestion window. When a connection starts or resumes after an idle period of time, *slow start* is performed. Here, the congestion window is initialized to one segment and every new ACK increases the window by one MSS. After a certain threshold (called the slow start threshold, *ssthresh*) is reached, the connection moves into the *congestion avoidance* phase, in which the congestion window effectively increases by one segment for each successfully transmitted window. In response to a packet loss, the sender

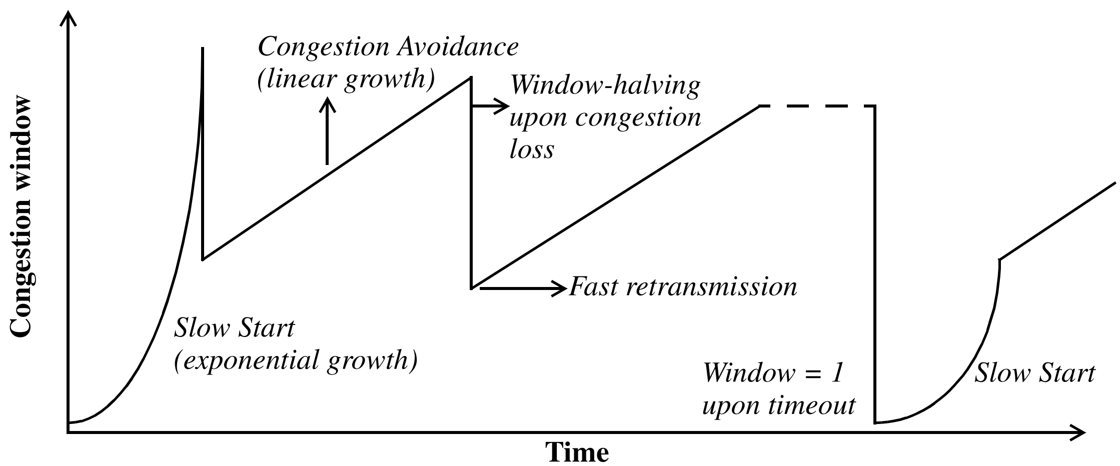

Figure 2.3 Dynamic time evolution of the TCP congestion window.

halves its congestion window; if a timeout occurs, the congestion window is set to one segment and the connection goes through slow start once again. This state evolution is shown in Figure 2.2 and the dynamic time evolution is shown in Figure 2.3.

In addition, *ssthresh* is set to half the value of *cwnd* at the time of loss, and at each stage slow start is performed until *cwnd* reaches *ssthresh*. At any point in time, the TCP sender ensures that the number of unacknowledged bytes is not larger than the smaller of the receiver's advertised flow control window and *cwnd*. In the steady state of the connection, the sender transmits a window's worth of data every round-trip, at a long-term rate equal to the bottleneck bandwidth. Since the ratio of window to round-trip delay equals the bottleneck bandwidth, TCP's congestion window size tends towards the connection's bandwidth-delay product. This is the number of outstanding bytes in transit in the steady state. Thus, it is clear that a low-bandwidth connection implies small transmission windows. In turn, this affects the fast retransmission mechanism because not enough duplicate ACKs arrive, leading to expensive timeouts.

Data transmissions are triggered and clocked by ACKs that arrive for previous segments; every time an ACK arrives, the transmission window shifts by an amount equal to the sum of the number of bytes acknowledged and the increase in *cwnd*, subject to the constraints imposed by the flow control window. This elegant notion of transmitting data based on incoming ACKs, called *ACK-clocking,* makes TCP a tightly-coupled feedback system and frees the sender from maintaining explicit software timers for transmission. However, it also leads to significant performance degradation when ACK feedback is imperfect or variable.

2.2.4 Connection Management

TCP has a sophisticated connection initiation mechanism using a three-way handshake, where a SYN exchange occurs before the connection is established and data can flow. A connection is uniquely identified by the source and destination IP addresses and port numbers. Details of the connection setup, teardown and state transitions are orthogonal to our work (see [116, 135] for details).

2.2.5 Window-based vs. Rate-based Congestion Control

TCP's window-based congestion control algorithm is elegant because it does not need an explicit clock to pace its data transmissions, since ACK clocking ensures that new data segments are sent out triggered by incoming ACKs. However, it is well-known that several problems arise from using the sender's transmission window to perform both congestion and error control in window-based protocols such as TCP. These arise because of the tacit assumption that all losses are the result of network congestion, and the use of the same window for both loss recovery and congestion management. The alternate approaches proposed by several authors attempt to achieve a separation of congestion control and loss recovery by performing congestion avoidance using a *rate-based* scheme. The basic idea is that the sender controls the rate at which packets are transmitted into the network using a timer. That is, the ACK-clocking mechanism used in TCP, is avoided in favor of a mechanism that is more explicit and not window-based.

This separates the process of error control from congestion control, because loss recovery is no longer tied to the congestion window. Transport protocols in connection-oriented networks often use this method [40]. Other protocols that use this method include NETBLT [32] and XTP [137].

The fundamental problem with pure rate-based schemes is that all packet transmissions are triggered by timer events at the source. This results in the source becoming the transmission bottleneck in many circumstances, especially as network bandwidths and number of connections increase (such as at a busy Web server). Usually, rate-based flow and congestion control schemes rely on some sort of connection establishment phase at start-up, and either reserve bandwidth (e.g., in connection-oriented networks like Asynchronous Transfer Mode networks), or estimate bandwidth as in Keshav's packet-pair scheme [75] and adapt to it. When congestion occurs, the sender decreases its sending rate by a multiplicative factor (e.g., by halving its rate) to perform congestion control. The other problem with a pure rate-based scheme for congestion control is that it requires support from the internal routers in order to enforce the rate negotiated between the sender and receiver on a per-connection basis [63], or at least provide isolation between flows (as in the packet-pair scheme that requires routers in the network to perform fair queueing [37]).

2.3 End-to-end TCP Enhancements

Over the past decade, TCP has been tuned to work well in wired networks in the face of network congestion, reacting to congestion by reducing its rate, recovering from lost segments, and probing for bandwidth in a careful way. In this section, we survey some proposed enhancements to TCP that improve its ability to recover from losses in a timely manner and/or perform better congestion control.

2.3.1 Selective Acknowledgments (SACK)

It is well known that TCP performance suffers due to coarse-grained timeouts when multiple segments are lost in a single window because it uses only cumulative ACKs. There has therefore been recent[1] interest in adding *selective acknowledgments* (SACK) to the standard TCP specification to reduce the time it takes to recover from multiple losses in a window. Fall and Floyd [42] describe detailed simulation results highlighting the benefits of SACK in large delay-bandwidth product networks.

TCP Selective Acknowledgments can be implemented in many ways. One possible approach is to use Keshav and Morgan's SMART (Selective Mechanism to Aid ReTransmission) scheme, where the receiver communicates the segment number that just arrived in addition to the cumulative ACK, whenever it sends a duplicate ACK. We elaborate on this scheme in Chapter 5 in the context of wireless links, and discuss its performance. A second approach, described in RFC 2018, is cur-

1. Actually, RFC 1323 [61] written in 1992 outlined a SACK mechanism, but some unresolved technical details delayed its acceptance as a standard for several years. RFC 2018 resolves these details.

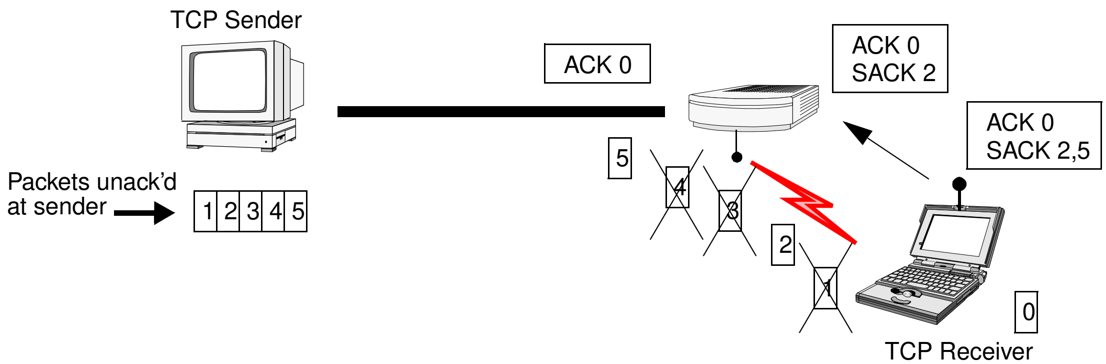

Figure 2.4 TCP Selective Acknowledgments according to RFC 2018. When segments 2 and 5 reach the receiver, the receiver sends two duplicate cumulative ACKs with the displayed SACK blocks to the TCP sender.

rently on track to become an Internet standard. In this scheme, the receiver reports up to three of the last received, out-of-order, maximal contiguous blocks of data, in addition to the cumulative ACK, so that the sender can accurately deduce which segments have reached the receiver (Figure 2.4). The information reported by the receiver in this scheme is a superset of the information reported in SMART. By reporting up to three blocks of data, it ensures that every block is reported at least thrice to the sender; this makes it robust to the loss of some SACKs on the reverse path.

While the RFC standardizes the format and mechanism used by the receivers, it specifies little about what the sender should do in response. The sender must ensure that appropriate congestion control actions continue to be taken as in TCP Reno. In addition to more responsive loss recovery, SACKs allow the sender to perform better congestion management since it now has better information about how many bytes have cleared the network pipe on each SACK arrival.

Chapter 7 describes the details of our SACK implementation and the sender mechanisms that use SACK information. It also describes the limitations of SACK when transmission windows are small, and presents a novel approach that significantly improves loss recovery when losses occur in a small window.

2.3.2 Newreno

Hoe's Newreno[1] modification to TCP Reno reduces the number of timeouts incurred by the TCP sender when multiple losses happen in a transmission window [55]. When multiple losses occur in

TCP Reno, a fast retransmission occurs after three duplicate ACKs arrive. Later duplicate ACKs are ignored until half the window is acknowledged, after which fast recovery sends a new segment for every incoming duplicate ACK. Now, when a new ACK arrives after the successful fast retransmission, its value will be within the original window (recall that there were multiple losses in the original window). In many cases TCP Reno would time out for this second loss if the second loss is "close" to the location of the first one, because the sender's window would now have been reduced to half its original value and the sliding window not have shifted the original window to beyond the original right edge.

In Newreno, however, the sender *remains* in fast recovery when this happens, when the new ACK arriving after a fast retransmission is *partial*. A partial ACK is defined as a cumulative ACK that does not acknowledge the original window completely. By remaining in fast recovery, the sender continues to send segments new segments, which would elicit more duplicate ACKs and eventually trigger another fast retransmission without incurring a timeout.

2.3.3 Forward Acknowledgments

Mathis and Mahdavi's TCP with Forward Acknowledgments (FACK) [91] uses SACK information at the sender to perform better congestion management. Rather than assume that an MSS-worth of data has been received by the receiver on every duplicate ACK, FACK calculates this precisely using information from the SACK fields. It explicitly maintains an "available window" variable, *awnd*, that keeps track of the number of bytes that are unacknowledged, to perform better congestion control. As long as *awnd* is smaller than *cwnd*, the sender is permitted to transmit more data.

2.3.4 NetReno

Concurrent with our work, Lin and Kung [82] discuss some mechanisms for improving TCP loss recovery and propose some modifications to TCP. They argue that their modifications are more sensitive to network conditions than current TCP is, motivating the name NetReno (for "Network-Sensitive Reno"). Some of our solutions in Chapter 7 on improving TCP loss recovery are similar to some of their independently proposed modifications.

---

1. To the best of our knowledge, Hoe's original paper did not use the term "Newreno." We saw it first in an earlier version of Fall & Floyd's paper [42].

2.3.5 Explicit Congestion Notification

In Section 2.2, we noted that TCP assumes that all losses signify congestion. In addition, TCP[1] has no other way of deducing that the network is congested, or that congestion is imminent. Deriving from earlier work in the DECBit scheme by Ramakrishnan and Jain [122], Floyd investigated the addition of Explicit Congestion Notification (ECN) to TCP [45].

When a gateway en route is congested or close to congestion, it sets a bit in the packet header and forwards it on. A *Random Early Detection (RED)* [47] gateway in marking mode is apt for this. A RED gateway detects incipient congestion by tracking the average queue size over a time window in the recent past. If the average exceeds a threshold, the gateway selects a packet at random and probabilistically marks it, by setting the *Explicit Congestion Notification* (ECN) bit in the IP packet header [45, 123]. This notification is echoed to the sender of the packet by the receiver. For TCP, this echo is piggybacked on an ACK. Now, when the sender receives and ACK with ECN, it multiplicatively reduces its congestion window, as it would if a packet loss had occurred. Thus, this mechanism with explicit support from gateways allows TCP to perform *proactive* congestion control, over and above the *reactive* one triggered by packet loss.

The ECN mechanism and the reactions of TCP to ECN are currently in the Internet Engineering Task Force (IETF) standards track [123].

2.4 Wireless Networks

The use of radio waves to communicate information has been known for over a century now due to pioneering work by inventors like Marconi, De Forest, and Armstrong. Starting from Marconi's demonstration of Trans-Atlantic radio communication in 1901, the field has grown by leaps and bounds to where it is today [80]. This section surveys today's state of wireless data technology.

The past three decades have seen numerous research and commercial efforts in building and deploying systems to transmit digital data over wireless media. In the late 1960s and early 1970s, the ALOHA project led by Norm Abramson at the University of Hawaii investigated packet-

1. Other recent proposals like TCP Vegas [21] perform proactive congestion control by comparing achieved with effective throughput, using the ratio *window/delay* to calculate throughput.

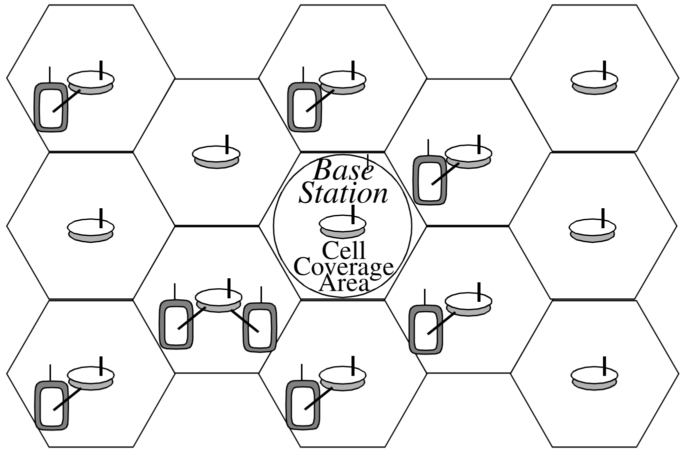

Figure 2.5 Schematic arrangement of cells in a cellular wireless network.
(Reprinted with permission from [129]).

switched networks over fixed-site radio links [1]. The contention resolution protocols developed in ALOHANET laid the framework for later Carrier Sense Multiple Access (CSMA) protocols for channel access. In 1972, the U.S. Defence Advanced Research Projects Agency (DARPA) launched the Packet Radio Program and its successor program on Survivable Adaptive Networks (SURAN) to study issues in packet radio networks such as channel access, link protocols, and routing [65, 79]. However, despite this work, wireless data communication remained largely a laboratory curiosity for several years.

In the 1990s, this has changed dramatically. Advances in hardware technology have enabled cheap and portable wireless devices for data communication. This, coupled with the rapid expansion of the Internet and the World Wide Web, has resulted in a variety of wireless technologies becoming commercially available.

The rest of this section surveys the key architectural ideas in wireless systems design and discusses some properties of wireless media. We then describe the pressing problems that hamper usability and degrade performance, as well as current solutions to these problems.

2.4.1 Cellular Network Topology

Many wireless and mobile networks are organized in a *cellular topology.* These topologies are composed of a wired, packet-switched, backbone network and a wireless network. The wireless network is organized into geographically defined cells, with a control point called a *base station*

(BS) in each of the cells, as shown schematically in Figure 2.5. These base stations are also directly connected to the wired network, routing packets between the wireless and the backbone network as shown in Figure 2.6. A mobile host (MH) receives data from a fixed host (FH) in the Internet routed via the BS of the cell it is currently in. As the MH moves between wireless cells, the task of forwarding data between the wired network and the mobile host must be transferred to the new cell's BS. This involves updating routing information in the wired infrastructure to reflect movement, and is known as a *handoff* [57, 58, 129, 100, 25]. The IETF standard for mobility is the Mobile IP protocol [112].

It is important for several applications and higher-level protocols that handoffs be as seamless as possible, incurring low latencies and causing little or no packet loss [25, 131, 22]. Numerous schemes have been proposed in the literature and several systems have been deployed to achieve these goals [e.g., 131, 50, 2, 26].

2.4.2 Wireless Technology Overview

Wireless data networks typically operate in the Radio Frequency (RF) (3 KHz to 300 GHz) or Infrared (IR) (1000 GHz to 30000 GHz) frequency range. IR offers high speeds over limited distances. Experimental research IR networks that offer up to 50 Mbps over a few meters [66], while commercial, IR-based wireless LANs have maximum bandwidths between 1 and 10 Mbps today. The IR frequency range is very close to that of visible light, which precludes transmission through walls or floors. Thus, while its range is more constrained than its RF counterparts, it offers a much higher degree of frequency reuse. The existence of impenetrable objects and reflections often causes IR dead spots in rooms, which in turn causes packet loss. The Infrared Data Association (IrDA) [6] has standardized the lower layer protocols and interfaces for IR devices, which is almost certain to be a part of every portable computer and handheld device in the future. Commercial IR-based LAN products available today include IBM's Infrared wireless LAN and InfraLAN.

In contrast to IR, RF can usually penetrate walls, which makes it an ideal technology for networks that must breach walls and floors. However, compared to IR, RF devices typically suffer from higher levels of interference and noise, which degrade the received signal and cause bit-errors. Because a large number of users in the same frequency band can degrade overall usability and performance, the use of different wireless frequencies in the United States is governed by the FCC and

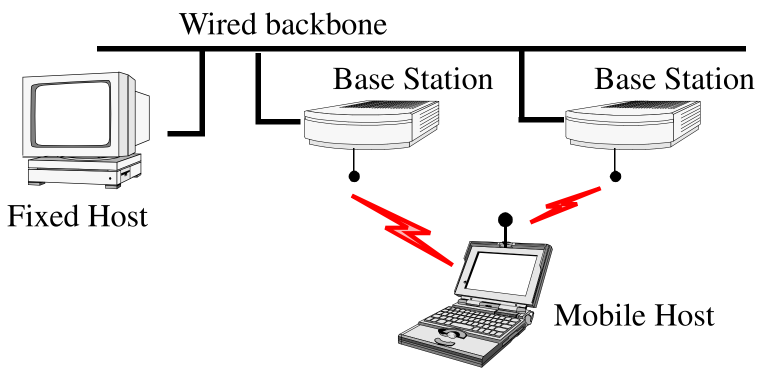

Figure 2.6 Network topology of a cellular wireless network. At any instant, one of the base stations routes packets between the wired backbone and the wireless network to the MH.

licenses are required to operate devices in many frequencies. An exception to this are the *unlicensed bands* discussed at the end of this section.

For several years, the application of RF to data communication exploited *narrowband* technology. Here, the input signal is modulated over a constant frequency wave, called the *carrier wave*, and the resultant signal is transmitted through the sender's antenna. When it reaches the receiver (often tainted by interference or noise), the receiver picks up this signal through its antenna, demodulates it, filters out the carrier frequency, and attempts to estimate the signal that was transmitted based on certain *decoding rules*. An error could occur in the decoding process if the level of noise or interference is too high—this manifests itself in the form of corrupted bits, which higher-level data protocols or applications must handle.

One of the major problems with narrowband systems that makes it vulnerable to interference is its lack of *frequency diversity*, since the entire carrier is forever concentrated in a single frequency. *Spread spectrum* [128] is a technique to combat this problem. Here, the transmitted signal is spread over a broad frequency band, which makes it more robust to interference or jamming and more secure against eavesdropping.

There are two common kinds of spread spectrum techniques: *direct-sequence* and *frequency-hopping*. In direct-sequence spread spectrum, the input signal is transmitted simultaneously over a broad range according to a pre-assigned code. In frequency-hopping spread spectrum, the transmitter sends data on one narrowband frequency for a short amount of time, then jumps to another narrowband frequency, using a pseudo-random hopping sequence. The receiver synchronizes with the sender's hopping sequence and tunes in to those frequencies to receive and decode the signal.

While frequency-hopping is more robust to sources of interference than spread-spectrum, maximum data rates are typically lower. Thus, it is not as suitable for high-speed wireless LAN access as for slower, wide-area links.

Examples of RF-based wireless LAN products include Solectek's AirLAN, Aironet's ARLAN [5], Motorola's Altair Plus-II, Proxim's RangeLAN products [120], and Lucent Technologies' WaveLAN [145]. Most of these operate in the unlicensed Industrial, Scientific and Medical (ISM) bands at 915 MHz and 2.4 GHz that have been set aside by the FCC for experimental purposes. In our work, we experimented with Lucent's WaveLAN product operating in the 915 MHz range. In this frequency range, products have to follow the "etiquette rules" for transmission mandated by the FCC [43].

Today's wireless LAN technologies typically provide peak link bandwidths in the 1 to 4 Mbps range. In the future, this is sure to increase; for example, the proposed HiperLAN network technology is rated at 20 Mbps. In Chapter 4, we "time travel" and evaluate TCP performance over these higher bandwidth wireless networks.

Examples of wide-area RF networks include Metricom's Ricochet network [98] that operates in the 915 MHz band and uses frequency-hopping spread spectrum technology, the Cellular Digital Packet Data (CDPD) network [121], and the GSM data service [99]. Peak link bandwidths in these outdoor networks today are typically between 10 and 100 Kbps, while future systems are likely to be in the 100-500 Kbps range (e.g., Metricom's proposed "Autobahn" network [126]). In Chapter 3, we discuss the details of the specific networks we experimented with in our work.

2.5 Wireless Bit-Errors and User Mobility

Figure 2.7 shows a typical loss situation over a wireless link. Here, the TCP sender is in the middle of a transfer to a mobile host. At the depicted time, the sender's congestion window is 5 packets. Of the five packets in the network, the first two packets are lost on the wireless link. TCP tacitly assumes that all packet losses are the result of network congestion, which is an invalid assumption in networks that are error-prone due to wireless link corruption or loss-prone due to user mobility.

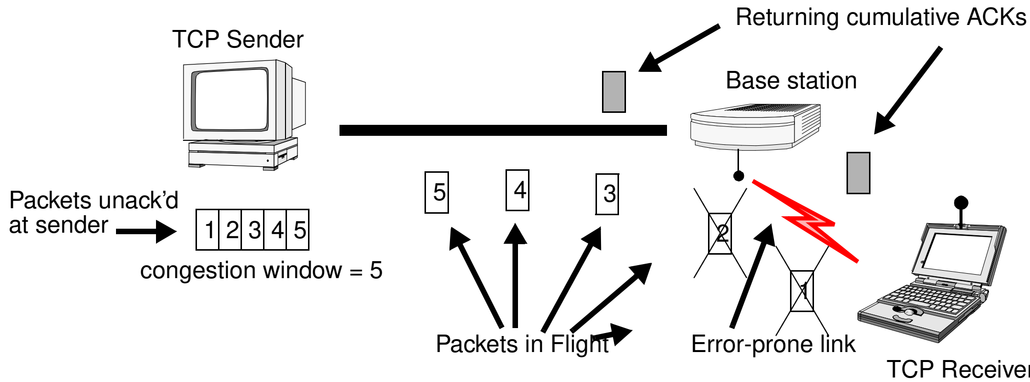

Figure 2.7 An example loss situation over a wireless link.

2.5.1 Reasons for Wireless Data Corruption

Wireless bit-errors occurs for a variety of reasons such as *path loss, fading, noise,* and *interference.* Path loss is defined as the ratio of signal power at the receiver to that at the sender and is a strong function of the distance between the receiver and transmitter. There are numerous path loss models that apply to different propagation mechanisms, such as free space, reflection, diffraction, and scattering. Rappaport [124] provides an excellent presentation of these empirical models. Typically, the path loss relationship has the form $L = P_r/P_t = K/f^2 d^n$, where P_r is the received power, P_t is the transmitter power, f is the central frequency of transmission, d is the distance between transmitter and receiver, n is the path loss exponent, and K is a constant. In free space, $n = 2$, whereas it varies between 3 and 5 for typical outdoor environments [124]. Thus, the higher the path loss exponent, the higher the bit-error rate for the same transmitter power.

Shadow fading [84] occurs because of objects and obstructions in the path of the transmitted signal. Various studies have shown that the power (in dB) of signals subject to shadow fading exhibits a Gaussian distribution, with the random attenuation due to shadow fading changing as the mobile host moves in a region. *Multipath (or small-scale) fading* occurs because of interference between multiple versions of the same signal arriving at the receiver at different times due to reflections from surrounding obstacles. Unless they interfere constructively, an uncommon occurrence, this results in signal degradation because of destructive interference.

Wireless networks experience interference from a variety of other sources. For example, *co-channel interference* can occur due to frequency reuse, where users in nearby cells using the same frequencies interfere with each other. Adjacent frequency channels in narrowband technologies cause interference too, although this can be ameliorated using spread spectrum techniques (Section 2.4.2) or *notch filters* [128], which selectively block a band of interfering frequencies.

Several techniques can be applied to radio and cell design to alleviate the severe problems that degrade wireless transmissions [49, 84]. However, despite these techniques, wireless bit-error rates that must be dealt with by higher-layer protocols and applications are often high and time-varying. Several studies have shown average bit-error rates to vary between 10^{-2} and 10^{-8} as conditions change [93, 102, 12].

Because TCP tacitly assumes that *all* losses are due to network congestion, it reacts to these corruption-induced losses by reducing its congestion window. In addition, the burst nature of wireless losses often leads to coarse-grained timeouts, especially in the absence of SACKs and when the transmission window is small. The result is unacceptably low throughput and under-utilization of available bandwidth.

2.5.2 Link-layer Protocols

Link-layer protocols are the conventional approach to handling error-prone wireless links [7, 69, 101]. They attempt to shield the TCP sender from the wireless link by performing local loss recovery and present it with a link with an effectively lower error rate. The main attraction of employing a link-layer protocol for loss recovery is that it fits naturally into the layered structure of network protocols. Traditional approaches include mechanisms such as Forward Error Correction (FEC), Acknowledgment-Repeat-reQuest (ARQ), and hybrid approaches that combine these two techniques [83, 7]. FEC is effective when errors occur at random and when the error characteristics of the channel are well-understood. Such schemes are not well-suited to links where the underlying bandwidths are not large (less than a few Mbps) and where the link latency has to be low. FEC schemes to tackle burst errors and long error sequences tend to use interleaving and increase latencies. In addition, if the error characteristics are not known in advance, the effective available bandwidth available for data communication after FEC overhead is only a fraction of the maximum. This makes ARQ and hybrid ARQ/FEC techniques attractive for low to medium bandwidth wireless systems, compared to pure FEC techniques.

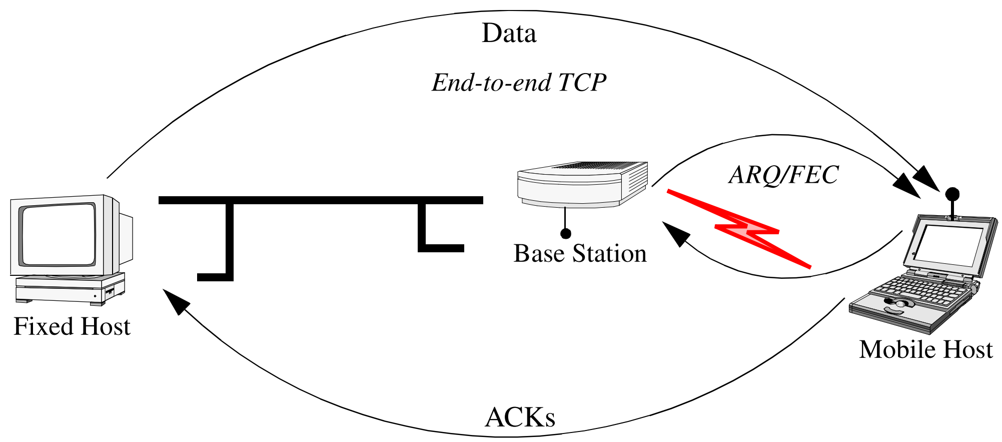

Figure 2.8 Reliable link-layer protocols perform local loss recovery over the wireless link, but could interact adversely with independent end-to-end mechanisms.

Standard link-layer ARQ schemes are usually stop-and-wait, Go-Back-N (GBN), or selective-repeat. Stop-and-wait ARQ is inefficient, especially when there are only a small number of wireless hops as is the case in cellular wireless networks, since it reduces the effective transmission window size to one. GBN is also inefficient because the transmission window does not slide, and an error anywhere in the window causes the retransmission of the entire window starting from that point. Selective repeat using a small window is the best scheme among these, and can be implemented using negative acknowledgments. It is important for the link-layer to be very simple and efficient to permit an implementation in hardware or firmware, and a link-layer protocol should not attempt to do too much at this layer (e.g., implementing a TCP-like protocol with almost complete reliability and in-order delivery) since that makes it very complex. Since higher layer protocols provide end-to-end reliability, it is wasteful to implement a very complex and sophisticated link-layer protocol—this is a consequence of the well-known end-to-end argument in system design [127], applied to network protocol design.

However, there are several occasions when end-to-end performance does not improve when conventional link-layer protocols are used. There are three chief forms of adverse interaction that could arise between independent reliable transport and link layers:

1. *Timer interactions:* Independent timers at both layers could trigger at the same time, leading to redundant retransmissions at both layers and degraded performance [38]. As a result, the transport sender is not shielded from the problems over the wireless link by the link-layer protocol. While this is a concern in general, TCP does not suffer from this problem in practice because of the coarse-grained and conservative timeout intervals.

2. *Fast retransmission interactions:* This arises when a link layer protocol achieves reliability by local retransmissions, but does not preserve the in-order sequential delivery of TCP segments to the receiver. Then, although local recovery occurs, the receipt of later segments causes duplicate ACKs from the receiver as well, leading to redundant sender fast retransmissions, sender window reduction, and reduced throughput [14]. This is a common occurrence in many link-layer protocols that do not preserve packet ordering. In this case, while local loss recovery is quick, the sender is not shielded from wireless losses because of the duplicate ACKs propagating back to it.

3. *Large round-trip variations:* If a link layer protocol is designed to be overly robust and attempt, for example, to provide a TCP-like service, it often results in long latencies and large round-trip time deviations at the TCP sender. This leads to long and conservative timeouts when congestion-related losses occur on the path. Other problems that arise include a large number of redundant retransmissions if a sender timeout occurs for a wireless loss [87], a problem observed in GSM networks running the RLP protocol [99]. The fundamental problem in this approach is that the link-layer protocol tries to do too much in terms of providing reliability.

Thus, we see that while link-layer protocols provide local loss recovery, they do not always shield the TCP sender from wireless errors. This leads to sub-optimal performance.

2.5.3 Split-Connection Protocols

In the mid-1990s, split-connection approaches gained in popularity as a way to improve TCP performance over error-prone wireless networks. They were motivated by the observation that the problems arose because the sender was imperfectly shielded from the error-prone link. To effect better shielding, split-connection protocols [e.g., 149, 10, 81, 143] split each TCP connection between a sender and receiver into two separate connections at the base station—one TCP connec-

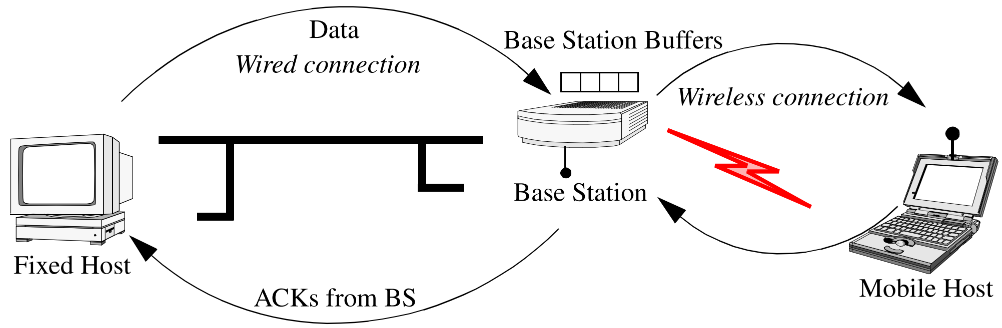

Figure 2.9 Split-connection protocols split the end-to-end TCP connection in two, with the sender's TCP terminating at the base station

tion between the sender and the base station, and the other between the base station and the receiver (Figure 2.9). Over the wireless hop, a specialized protocol tuned to the wireless environment may be used. Now, loss recovery for wireless losses can be done on the second connection without the data sender retransmitting data or reducing its window.

In [149], the Yavatkar and Bhagwat propose two protocols: one in which the wireless hop uses TCP and another in which the wireless hop uses a selective repeat protocol (SRP) on top of UDP. They study the impact of handoffs on performance and conclude that they obtain no significant advantage by using SRP instead of TCP over the wireless connection in their experiments. However, our experiments demonstrate benefits in using a wireless-optimized simple SACK protocol over the wireless connection to combat errors (Chapter 5).

Bakre and Badrinath's Indirect-TCP (I-TCP) [10] is a split-connection solution that uses standard TCP for its connection over the wireless link. Like other split-connection proposals, it attempts to separate loss recovery over the wireless link from that across the wireline network, thereby shielding the original TCP sender from the wireless link. However, as our experiments in Chapter 5 demonstrate, the choice of TCP over the wireless link does not really improve performance. Since TCP is not well-tuned for the error-prone link, the TCP sender of the wireless connection often times out, causing the original sender to stall. In addition, every packet incurs the overhead of going through TCP protocol processing twice at the base station (as compared to zero times for a non-split-connection approach), although extra copies are avoided by an efficient kernel implementation. A major disadvantage of split connections is that the end-to-end semantics of TCP ACKs is violated, since ACKs to segments can now reach the source even before the packets actually reach the mobile host. This makes the system vulnerable to base station failures or temporary changes in routing to or from the MH. Also, since split-connection protocols maintain a significant amount of state at the base station per TCP connection, handoff procedures tend to be complicated and slow [8], often taking over an order of magnitude longer than alternate approaches [131].

Chapter 5 investigates the performance of two split-connection protocols and compares them to other approaches. We show that the performance improvements of these solutions compared to TCP are due to the special wireless-optimized techniques that can be deployed over the wireless connection, and *not* due to the split architecture. Furthermore, we show that wireless optimizations can be designed and implemented using a transport-aware reliable link protocol without incurring the architectural drawbacks of the split-connection approaches.

2.5.4 M-TCP

Inspired by the transport-aware ideas in the Snoop protocol (Chapter 4), Brown and Singh [22] propose a protocol called M-TCP that is designed to provide good performance when bit-error rates are low but disconnections due to mobility are frequent. They develop a split-connection approach that does not violate the end-to-end semantics of TCP ACKs, by taking advantage of the persist state in the TCP specification [20]. They throttle the sender completely when the mobile host is disconnected and resume activity when connectivity to the mobile host is reestablished. In the meantime, they ensure that no sender timeouts or retransmissions occur by sending ACKs that put the sender into persist mode. M-TCP does not readily address the problems due to varying and non-negligible error rates, focusing instead on disconnections due to mobility or link outages.

2.5.5 User Mobility

Caceres and Iftode [25] address the issue of TCP performance when communication resumes after a handoff. The TCP sender interprets the delay caused by a handoff process to be due to congestion. In many systems, handoffs complete quickly (relative to retransmission timer granularities), in between a few tens to a few hundred milliseconds, and long waits are required by the mobile host before timeouts occur at the sender and packets are retransmitted. They propose a fast retransmit approach to mitigate this problem by having the mobile host send a certain threshold number of duplicate ACKs to the sender and thereby trigger a fast retransmission. Thus, a connection can now resume without incurring a timeout.

Manzoni et al. [90] use the above results to motivate changes to TCP sender implementations to explicitly recognize ongoing mobility using messages sent from the base station. During this time, they extend the slow start phase to reduce the adverse effects of packet loss due to host mobility. We believe that the previous approach [25] solves the key problems in a simpler manner because it requires no changes to TCP fixed host implementations.

Maltz and Bhagwat propose the use of transport layer mobility and TCP splicing [89] to overcome the problems associated with mobility on transport performance. The innovative aspect of their approach is the use of a split connection approach that does not violate the end-to-end semantics of TCP ACKs. The connection is "split" in the kernel of the base station by changing certain contents of the TCP/IP header before forwarding the segment to the mobile host. This prevents any disruption during a handoff. A transport-aware link protocol like the Snoop protocol can coexist with this scheme, providing good performance when wireless corruption occurs.

We observe that the above solutions mainly address the problems due to mobility and not the error characteristics of wireless links. In Chapter 4, we show how the Berkeley Snoop Protocol and a multicast-based low-latency handoff scheme combine to provide good performance in the face of both wireless bit-errors and user mobility.

2.6 Asymmetric Networks

A wide variety of asymmetric technologies are becoming an increasingly popular alternative for Internet access. Bandwidth asymmetric networks, where the bandwidth in one direction (the *forward* direction) is several times that in the reverse, are becoming commonplace. The motivation for such networks is primarily economic—it is much cheaper and easier to provide high bandwidth in one direction than in both—and it is possible to take advantage of existing telephone or cable infrastructure to do this. In addition, today's dominant Internet application, Web access, displays asymmetric characteristics, with more data typically originating from a server than from a client.

Typical bandwidth ratios between the forward and reverse directions in these networks vary from between 10 and 1000. Prominent examples of bandwidth asymmetric networks include cable modem networks, Direct Broadcast Satellite (DBS) networks and wireless cable networks. In wired cable modem networks, the forward channel uses conventional cable lines that are also used for cable television distribution. The reverse channel is either out-of-band, using a dialup telephone line, or is in-band using the same cable lines but at significantly lower bandwidth than in the forward direction. DBS networks such as Hughes' DBS system [39] provide large forward bandwidths over a large area. The reverse channel is usually via a telephone dialup line or a wireless link. Wireless cable networks are similar to their wired counterparts in terms of packet framing and the media-access protocols, but use an entirely wireless forward path. This is the network we use in our research; more technical details of this network are presented in Chapter 3.

There have been some recent efforts to analyze and solve the problems raised by bandwidth asymmetry on TCP performance. These problems arise because of slow ACK feedback, loss of ACKs, or the presence of bi-directional traffic in the network.

2.6.1 Asymmetric Satellite Networks

Durst et al. [41] investigate the performance of TCP over asymmetric, error-prone satellite networks. They propose the following extensions to TCP for this environment, as part of a the Space Communications Protocol Standards-Transport Protocol (SCPS-TP): signaling different types of loss to the sender (congestion, corruption and link outage) based on explicit default assumptions about link loss characteristics, delaying ACKs based on receiver estimates of round-trip time from the timestamp option [61], header compression that tolerates some packet losses (unlike the scheme for serial links in RFC 1144) and the use of selective negative ACKs (SNACK) for better loss recovery.

Inspired in part by the Service Specific Connection-Oriented Protocol (SSCOP), an ATM adaptation layer protocol, Henderson and Katz discuss the design and implementation of the Satellite Transport Protocol (STP) for better transport performance over loss-prone satellite networks [53]. An STP sender does not rely on continuous ACK feedback from the receiver, which helps it perform better than TCP Reno over asymmetric paths. Periodically, the receiver sends a STAT (status) message in response to a sender POLL, which informs the latter of missing packets or blocks of data, which the sender retransmits. The frequency of STATUS messages is a tunable parameters; in addition, the receiver can send an unsolicited status (USTAT) message informing the sender of its state.

2.6.2 Bandwidth Asymmetry Analysis

Lakshman and Madhow [78] study the problems caused by bandwidth asymmetry using analysis and simulation. They define the *normalized asymmetry ratio* as the ratio of the bandwidth ratio to the average packet size ratio, and show that performance degrades if this quantity is greater than one. They also propose the use of the drop-from-front strategy on ACKs in the constrained reverse channel buffer as a way to alleviate the problems that arise.

2.6.3 Bi-directional Traffic

Kalampoukas et al. [67] study the problems that arise when bi-directional traffic is present in a bandwidth-asymmetric network, using a combination of analysis and simulation. They show that *ACK compression,* where ACKs arrive in quick succession disrupting smooth ACK clocking, sometimes occurs in these environments reducing throughput. They investigate three approaches to counter this: (i) prioritizing ACKs over data on the reverse channel, which unfortunately results in starvation, (ii) using a connection-level backpressure mechanism to limit the maximum amount of data buffered in the reverse channel router, which unfortunately results in unfair bandwidth utilization and (iii) using connection-level bandwidth allocation, which provides flow isolation.

2.6.4 ACK Congestion Control

In [14], we propose the use of ACK Congestion Control (ACC) as a way to adaptively extend TCP's delayed ACK approach at the receiver, using in-band congestion feedback from the constrained reverse router and TCP sender. This approach was further investigated by Padmanabhan

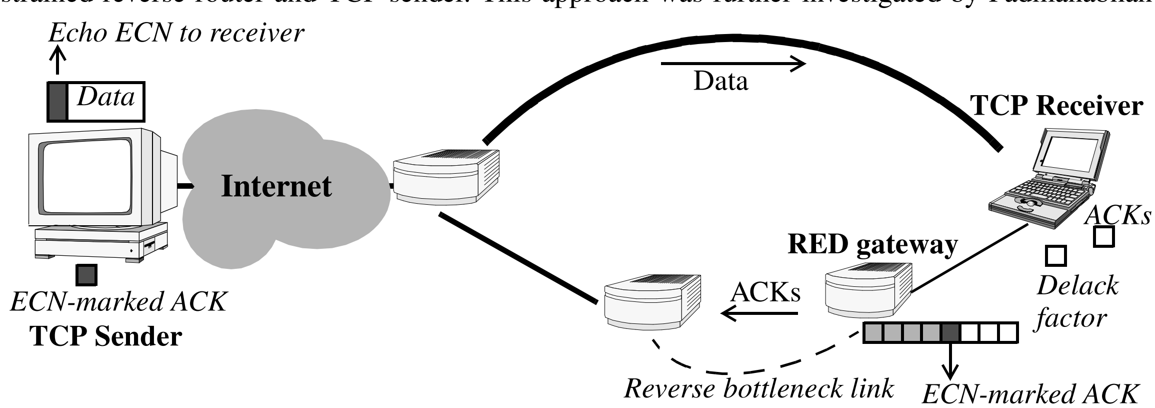

Figure 2.10 ACK congestion control using a RED gateway at the reverse router with ECN support at the sender and receiver to dynamically vary *delack*.

especially in the context of bi-directional traffic [107]. When used in conjunction with a scheduling mechanism that prioritizes ACKs over data on the reverse channel, ACC achieves close to optimal throughput for transfers in both directions, sharing the reverse bottleneck resources fairly.

The idea in ACC is to extend TCP congestion control to TCP ACKs in addition to data, since ACKs consume scarce resources at the low-bandwidth bottleneck link in the reverse direction. In many systems (e.g., BSD/OS), ACKs occupy slots in the reverse channel buffer, whose capacity is often limited to a certain number of *packets* (rather than bytes).

ACC relies on the Random Early Detection (RED) algorithm [47] coupled with ECN [123] at the gateway of the reverse link to aid congestion control (Section 2.3.5). In ACC, *both* data packets and TCP ACKs are candidates for being marked. The TCP receiver maintains a dynamically varying delayed ACK factor, *delack*, and sends an ACK every *delack* data packets. When it receives a packet with the ECN bit set (echoed by the data sender to it), it increases *delack* multiplicatively, thereby multiplicatively reducing the frequency of ACKs. Then, in each subsequent round-trip time (determined using the TCP timestamp option [61]) during which it does not receive an ECN-marked packet, it linearly decreases *delack*, thereby additively increasing the frequency of ACKs. Thus, the receiver mimics the standard congestion control behavior of TCP senders in the manner in which it sends ACKs, extending TCP congestion control to ACKs as well (Figure 2.10).

There are bounds on the delayed-ACK factor *delack*. Obviously, the minimum value of *delack* is 1, since at most one ACK is sent per data packet. The maximum value of *delack* is determined by the sender's window size, which is conveyed to the receiver in a new TCP option. The receiver should send at least one ACK (preferably more) for each window of data from the sender. Otherwise, it could cause the sender to stall until the receiver's delayed-ACK timer (usually set at 200 ms, but no more than 500 ms in the TCP specification) forces an ACK to be sent.

The advantage of ACC is that it is an end-to-end scheme where most of the machinery is at the end points of the TCP connection. The only assumption made of the network is that the reverse bottleneck deploy a RED gateway in marking mode. With the current rapidly increasing interest in deploying RED gateways in the Internet, this does not seem to be a problematic assumption. However, there is a time constant on the order of a round-trip time before the state of congestion is inferred and dealt with by the TCP receiver. In addition, the protocol requires that the appropriate machinery be deployed at both the sender and receiver, as well as the reverse bottleneck gateway, which makes incremental deployment challenging. In Chapter 6, we present some performance results of ACC and compare it with the other schemes described in that chapter for combating the adverse effects of asymmetry in different networks.

2.6.5 TCP over Hybrid Fiber Coaxial Broadband Access Networks

Cohen and Ramanathan [35] study TCP performance over a hybrid coaxial cable distribution system using simulations of that network. They conclude that it is important for receivers to tune their socket buffer sizes; their proposed modifications include an initial window of two segments as opposed to one, smaller segment sizes to overcome problems with delayed TCP ACKs and ACK losses, a finer granularity for TCP timers, and fast retransmission triggered by a single duplicate ACK for better loss recovery. In general, while some of these techniques improve performance in their network, they are not universally applicable and lead to worse performance in other networks. For example, retransmitting a segment upon the arrival of the first duplicate ACK is not correct in general because of the significant amount of packet reordering in the Internet. Our philosophy, in contrast, has been to improve transport performance heterogeneous wireless networks without compromising performance in other situations.

The authors of this paper also claim that the Snoop protocol [16] requires significant overhead in the network and is ill-suited to large deployments. In Chapter 4, we discuss the scaling behavior of the Snoop protocol via simulation and analysis and show that this concern is unfounded.

2.7 Packet Radio Networks

A packet radio network is a network of nodes that communicate with each other over wireless, routing packets through a set of radio nodes. These network nodes may be either static or mobile. We are interested in topologies where end hosts with wireless interfaces use the radio network, traverse multiple wireless hops, and gain access to the wired Internet infrastructure via a gateway to the packet radio network. In this section, we briefly discuss the problems to transport performance that arise in these networks and describe related work in the area.

The first issue is *reliability*, since wireless links are error-prone (Section 2.5): a reliable link-layer protocol is therefore essential. The second issue is *channel access*: a contention-resolution protocol is essential to prevent the network from collapsing under even moderate load. The third issue is *packet reordering*: many packet radio networks explicitly route packets via different parallel routes, leading to potential reordering in the network that the end-points must handle.

The fundamental problems to TCP performance in these networks arise because of interactions between data and ACKs, which lead to imperfect ACK feedback. In this respect, these problems are similar to the ones observed in bandwidth asymmetric networks. In Chapter 6, we discuss this in detail and present solutions to them; to the best of our knowledge, there has been little progress to date on these problems in packet radio networks, despite their importance.

Metricom Inc. [98] has deployed a network using Ricochet modems and poletop packet radios in several areas in the United States. Our work is based mainly on their network and technology. Cheshire and Baker [29] present results of measurements of the Ricochet modems, focusing primarily on end-host overhead and latencies through the modems. Lai et al. [77] log traces of real traffic over an eight-day period over Ricochet modems. They show that mean round-trip latencies in their traces are over 700 ms and that higher delays occur due to packet loss and reordering.

2.8 Summary

This chapter started with a description of the best-effort Internet service model and motivated the need for reliable transport protocols discussing the salient features of several protocols of historic interest. We then described the salient features of TCP, the protocol that makes today's Internet reliable and discussed several recently proposed end-to-end enhancements to it. In Section 2.4, we surveyed some key concepts in wireless networks. In Section 2.5, we discussed and critiqued conventional approaches to tackling the problems caused by wireless bit-errors and user mobility. In Section 2.6 and Section 2.7, we described background material about bandwidth-asymmetric and packet radio networks respectively.

We observe that current approaches towards handling wireless bit-errors, asymmetric effects and small transmission windows do not comprehensively solve observed problems. The material described in this chapter sets the stage for our work. The next chapter outlines the methodology used in our work and describes the details of the wireless testbed and experimental networks that serve as our research platform.