A survey of TCP in wired-cum-wireless environments

K. Pentikousis
Department of Computer Science
State University of New York at Stony Brook
kostas@cs.sunysb.edu

Abstract

The Internet has evolved the last decade, reaching a larger number of users, and encompassing several new technologies. New classes of hosts such as mobile devices are gaining popularity, while the transmission media become more heterogeneous. Wireless networks exhibit different characteristics than wired ones. Mobile hosts have different needs and limitations than desktop computers. TCP has served well the wired Internet for almost 20 years but is not ready for wired-cum-wireless environments. In this paper we present the challenges that have to be met in order to provide reliable transport services to all hosts regardless of the type of network connectivity used. We survey the recently proposed solutions and evaluate them with respect to a wired-cum-wireless environment.

1 A wired-cum-wireless Internet

The Internet has been in constant evolution since the mid 80's, after the introduction of the Transmission Control Protocol (TCP) [50]. Lately though, the Internet has become ever more heterogeneous. Today, powerful PCs and workstations coexist with WebTVs [39], wireless phones, and Personal Digital Assistants (PDAs) [72]. Although, it is desirable in principle to provide to all hosts the same kind of network services, it is an open debate whether it is possible to do so.

Diversification in end-host capabilities limits to some extent the applications that would be running on each category of devices. Differences in computing power, memory size, input and output devices, in addition to mobility and power consumption issues, determine the potential uses of each end-host category. At the same time, end hosts use a far larger spectrum of networking technologies to connect to the Internet, that includes traditional wires and optical fibers, as well as wireless and infrared media.

Users need reliable transmissions for web browsing, email, file transfers, etc., and TCP is the dominant reliable transport protocol on top of which all of these services run. However, TCP was designed [50], and later modified as needed (e.g. [5]), with certain assumptions in mind. For example, segment losses are assumed to be due to congestion, because in wired media transmission errors are relatively rare. This is not true for wireless media where, due to fading channels and user mobility, transmission losses are more frequent. Since certain assumptions behind TCP's highly tuned algorithms do not hold for such transmission-heterogeneous media, current TCP implementations do not perform well in such environments [8][15][19][63]. In addition to random errors and hand-offs, TCP must also cope with connections that, in certain cases, exhibit limited

bandwidth and large RTTs. Moreover, for mobile battery-operated devices, power consumption is an important issue which is not addressed by current TCP versions.

TCP assumes that the underlying network infrastructure has limited service capabilities, i.e. it provides a best effort, unreliable packet delivery service. The Internet community has preferred so far a more end-to-end approach when attempting to provide reliability to applications [52]. With the advances in hardware over the last decade it has become apparent that networks can offer reliable services (e.g. ATM), or at least that this is becoming feasible. Therefore, adding more functionality at the layers below TCP is an active area of research.

1.1 End-user wireless networks

End-user wireless networks, i.e. networks that are not part of a backbone, can be classified using different criteria. Wireless networks can be classified as Local Area Networks (LANs) or Wide Area Networks (WANs), depending on the service area of the access point (or base station). They can also be classified according to the type of service they offer to the user, as explained in the *Appendix*.

Wireless LANs. Wireless LANs can provide sufficient bandwidth for office applications but relatively limited mobility: typically the user may roam inside a building or a campus. Wireless LANs have not yet replaced wired ones, though they are used as an extension to wired LANs. They offer a great service to a number of vertical markets like retail stores, warehouses and manufacturing plants [58]. There are two main standards, the High Performance Radio Local Area Network (HIPERLAN), and the IEEE 802.11 standard, also known as wireless Ethernet.

A European Telecommunications Standard Institute (ETSI) committee designed the HIPERLAN Type 1 standard (HIPERLAN/1) [20]. HIPERLAN/1 uses a Carrier Sense Multiple Access (CSMA) Medium Access Control (MAC) protocol [48], and can achieve net data rates of up to 20 Mbps in a 50-meter range, or up to 1 Mbps in an 800-meter range. HIPERLAN/1 handles mobile hosts that can be moving at up to 36 km/h, and can provide quality of service based on the different categories of data [58]. The standard includes a provision for handoff handling, but does not provide the actual specification for this per se. HIPERLAN/1 was designed so that it can offer small delays because it is based on small messages that are exchanged relatively frequently. This fact in combination with the relatively high bandwidth qualifies HIPERLAN not only for data transfers but for other services as well, such as teleconferencing, video, and medical data transmissions [58].

The original IEEE 802.11 wireless LAN standard [26] was designed to achieve raw bit rates of 1 or 2 Mbps within a 100-meter range. Later, IEEE published two supplements to this 802.11 standard: 802.11a (40 Mbps in the 5.8 GHz band), and 802.11b, (11 Mbps in the 2.4 GHz band). The IEEE 802.11 uses a CSMA with Collision Avoidance (CSMA/CA) scheme [26][58] to regulate access to the wireless medium. Most of-the-shelf products, for example the ORiNOCO RG-1000 Residential Gateway [42], are compliant with IEEE 802.11b and operate in the unlicensed 2.4 GHz band, allowing the user to roam up to 150 meters at 11 Mbps in open environments, while in closed environments the maximum range at the lowest fallback rate (1 Mbps) is about 50 m.

The term **Wide Area Wireless Data Networks** (WAWDN) refers to WANs that are dedicated to data traffic [58]. The current generation of WAWDN includes the cellular

digital packet data (CDPD) system and the general packet radio service (GPRS), and uses packet switching. WAWDN provide bit rates of one to two orders of magnitude less than wireless LANs. On the other hand, a single base station in such WANs can cover a lot more area, usually some tens of square kilometers. The mobile user can enjoy network services at roaming speeds higher than in the case of wireless LANs. The mobile host, typically a laptop, is equipped with a radio modem, which is used to connect with the network.

Finally, **Cellular Networks** that can handle mixed traffic, i.e. both voice and data, are used for wireless access. First generation (analog) cellular networks like AMPS became obsolete due to the limited capacity and services they offer. Second generation cellular networks, like GSM/DCS 1800/PCS 1900, D-AMPS (IS-54), and IS-95 [17][20][58] are widely deployed in many countries [11]. Such networks can be used for data transfers, but are not very economical. The user must dialup to achieve connectivity with the network and is usually charged according to airtime spent, not the amount of data transmitted or received. The offered bandwidth is even less than in the case WAWDN. For example, GSM offers bit rates up to 9.6 Kbps (with GSM Phase 2+ up to 14.4 Kbps) [60], while IS-95 provides rates up to 19.2 Kbps [58]. We note that in the case of a WAWDN, like CDPD, the user is usually charged by the amount of data transferred so it is economically reasonable for a user to be constantly connected and able to transfer data any time without the need to dialup [1]. Third generation cellular networks, like UMTS and 3G CDMA, offer significantly higher bit rates. For the interested reader we provide more information on the services and specifications of wireless networks in the *Appendix*.

1.2 The case of TCP in the wireless environment

We will not delve into the details of TCP [50][5] in this report. The interested reader is referred to one of several excellent books on TCP, such as [57][73]. We will however discuss the parameters that affect the performance of TCP in a wired-cum-wireless environment.

Limited bandwidth. As explained in Section 1.1 wireless wide area networks offer limited raw bit rates. For example, CDPD networks [1] offer a raw bit rate of 19.2 Kbps, which is shared amongst up to 30 users. On the other hand, the current generation of wireless LAN standards offers sufficient bandwidth, for example the IEEE 802.11b standard offers raw bit rates of up to 11 Mbps, while HIPERLAN offers up to 20 Mbps.

Long Round Trip Times. In general, wireless media exhibit longer latency delays than wired ones [31][33][56][60]. This affects TCP throughput and increases the interactive delays perceived by the user. In addition, Lakshman et al. [29] show that under certain scenarios TCP-Tahoe is “unfair” for connections with longer round trip times (RTTs). Since the congestion window increases proportionally to the rate of incoming ACKs, two competing connections will experience different growth rates in their congestion window and therefore will achieve different throughput levels. In particular, the connection with the longer RTTs will see a much more moderate growth in the congestion window, and consequently will experience a smaller throughput. One can see that connections that include a wireless path are not favored by TCP when compared to other flows with the same number of hops.

Random losses. Wireless media are more prone to transmission losses, for example due to fading channels, shadowing, etc. TCP was designed with a wired medium (copper

cables) in mind, which has bit error rates **(**BER) in the order of 10^{-6}-10^{-8}. This translates into a segment loss rate of about 1.2% for 1500-byte segments. BER are much higher in the wireless domain, usually in the order of 10^{-3}, and sometimes as high as 10^{-1} [31]. With BER in the order of 10^{-3} the packet loss rate is an order of magnitude more in a wireless environment (about 12%). However, we should note that usually Forward Error Correction (FEC) is employed which effectively brings the False Alarm Error rate down to the order of 10^{-6} [31].

Random losses are the most prominent problem addressed in the literature, which is why many of the solutions aim at alleviating this deficiency in TCP. The problem with retransmissions does not so much lie in the very fact of sending a segment again. After all, the segment was lost, so the only way to deliver it to the receiver is to resend it. The problem is that a lost segment triggers congestion avoidance mechanisms [5], which essentially make the sender's window much smaller. In this way, a transient error leads TCP to back off too much and not be able to sustain a good throughput level [63].

User mobility. Wireless networks enable the user to move around. Perkins [47] makes a clear distinction between *portability* and *mobility*. In the first case, the user may use, for example, a laptop that can plug into the network at several different access points. However, this implies that there is some time (practically, the time spent between each transition) during which the user does not enjoy network services. The term *mobility* means that the user can roam freely and at the same time seamlessly enjoy network services without interruptions. A number of issues related to user mobility led to the creation of the Mobile IP working group by the Internet Engineering Task Force (IETF) [28].

Wireless networks are usually designed in a cellular fashion, where each cell covers a particular area. Users inside this area are serviced from a single base station (BS) or access point (AP), a host that is connected to the wired network and offers wireless connectivity to wireless hosts. When a host is moving from one cell to another a procedure named handoff (or handover) must be followed. During a handoff, all necessary information must be transferred between the two base stations so that the mobile host can continue to be connected. Caceres and Iftode were amongst the first to study the implications of mobility on TCP performance [15] (see Section 2.4.4.5). The research community consents that it is desirable for reliable transport protocols to be able to differentiate between congestion-related, transmission (or random), and motion-related losses.

Short Flows. Web browsing and email account for the majority of today's Internet traffic. These services usually include the transmission of a rather small amount of data. This means that when the application layer protocol opens a TCP connection for data transfer there is a very large possibility that the whole transfer is completed while the TCP sender is in the slow start phase. Therefore, the TCP connection never manages to fully utilize the available bandwidth. Savage et al. [53], for example, claim that a 10 KB download under perfect conditions and with infinite bandwidth will not proceed with a throughput faster than 300 Kbps! Indeed, if the client is 35 ms "away" from the server, the 10 KB transfer will need a 70 ms RTT for the connection establishment, and at least another 3 RTTs for the actual transfer. This adds up to 280 ms, making the perceived throughput 286 Kbps. Of course, evolution in application level protocols that are aware of TCP workings allow for better performance in some cases. For instance, the first

version of HTTP (Hypertext Transfer Protocol) would open a new TCP connection for every single object in a web page[1]: a page with 3 images could mean 4 TCP connections. HTTP/1.1 [25] solves this problem by introducing the concept of persistent connections: in the above scenario, the first TCP connection that is used to fetch the HTML document will be used to fetch the 3 images as well.

Power consumption. For battery-operated devices like laptops, PDAs and wireless phones, power consumption is a very important aspect. Typically, communicating over a wireless medium consumes more battery power than CPU processing [33]. Of course TCP was not designed with energy expenditure in mind. However, TCP has been shown in several studies, including [9][46], to be very efficient in terms of retransmitted segments. TCP manages to have a very low overhead, i.e. it does not waste bandwidth. For example, in [9] TCP is shown to achieve almost perfect goodput (defined therein as the ratio of the actually transmitted bytes divided by the number of bytes to be transferred). On the other hand, energy efficiency does not only depend on the amount of avoidable extra data, but also on the total duration of the connection. Prolonged communication times lead to power consumption too.

New wireless-oriented protocols, like [56][64][69], can be designed with all the above considerations in mind, and therefore perform a lot better than TCP under a number of scenarios. Although specialized transport protocols, or even protocol stacks like WAP, may provide the framework for the development of wireless browsing, email and calendar services, users will still require access to all other critical information like databases, file sharing, and other network services. Today, this can be done only on top of TCP. Otherwise specialized new applications must be built for all currently existing applications, which is rather uneconomical, and maybe not even realistic. TCP has an advantage over any new proposal because it has proven to be robust over the years, it is widely deployed, there is a vast store of experience with it, and it makes maintaining compatibility with the rest of the Internet easier.

It is also important to note that almost all past studies did not take power consumption under consideration. Although it is for the market to decide, solutions that do take power consumption into account would have a clear-cut advantage.

2 Taxonomy of solutions

The research community has been very active in trying to solve the issues related to TCP performance in the wireless domain. Some researchers have attempted to provide solutions at the Data Link Layer (LL), thereby attempting to hide the deficiencies of the wireless channel from TCP. Others have introduced modifications to TCP so that it performs better under the new conditions. Last but not least, there are a number of proposals for new transport protocols, optimized for wireless networks.

2.1 Link layer solutions

As noted earlier the major problem for a reliable transport protocol like TCP arises because of the different nature of the wireless medium. Therefore, it is reasonable to attempt to attack the problem at its root cause. The LL protocol that runs on top of the Physical Layer has immediate knowledge of dropped **frames** and thus can respond faster

[1] An HTML (Hypertext Markup Language) document.

than higher level protocols. At the same time, the LL protocol has more control over the Physical Layer protocol. Hence, alleviating the wireless medium inefficiencies at the LL provides the transport layer protocol with a dependable communication channel, similar in characteristics to a wired one. This way, the transmission media heterogeneity introduced in the network remains transparent to the existing software and hardware infrastructure, and does not necessitate any changes to current TCP implementations

Unfortunately, making a wireless channel look like a wired one is not an easy task. The LL protocol will have to ensure relatively reliable delivery of packets. This is usually implemented using an automatic repeat request (ARQ) scheme and/or by means of FEC. ARQ works well when the error rates are low. High error rates can lead to a lot of retransmissions and can even cause a complete "black-out" in the connection. FEC on the other hand, is not very well suited for channels that do not have large bandwidth, as is the case for many categories of wireless. After all, FEC can detect and reverse a limited number of bits, but the penalty in bandwidth can be considerable. For example, the raw bit rate for CDPD, which uses FEC, is 19.2 Kbps, while the actual user bit rate is 9-13Kbps [58]. In addition, the authors in [44] argue that the use of FEC means more power consumption and computation delays per packet. Tabbane [58] also notes that FEC requires fast digital signal processing chips, which indeed consume extra power.

A question that still remains unresolved is whether the LL protocol should be aware of TCP workings or not. Early studies [19] have shown that a TCP-unaware LL protocol may hide the wireless link errors from TCP, but at the same time lead to worse overall performance. For example, upon the loss of a packet over the wireless link the LL protocol will retransmit the packet, without waiting for a TCP retransmission. However, a TCP-unaware LL protocol will not suppress the duplicate acknowledgements, which can cause a TCP retransmission in addition to the LL one [9]. This duplicated effort leads to worse performance and waste of scarce resources, i.e. wireless bandwidth and battery power. It has been noted that LL retransmission timers must expire faster than TCP's in order to avoid this duplicated effort. Moreover, the LL protocol should try to avoid triggering Fast Retransmit at the TCP sender, for example by not delivering incoming frames out-of-order.

2.1.1 TCP-aware LL protocols

The most prominent proposal in this category is the snoop protocol [8]. Snoop was designed so that the wired infrastructure of the network would need no changes. A snoop agent must, however, reside at the base station (BS), while the mobile host would need to run the snoop protocol. The BS snoop agent inspects every TCP segment that is sent to or received from the wireless hosts that are roaming in its BS cell. We will not go into the details of the protocol, but it is useful to present the salient points of its workings.

The snoop agent maintains a cache of unacknowledged data segments destined for the mobile host. When the snoop agent receives a duplicate acknowledgment from the mobile host destined for the fixed host, it retransmits the missing segment and suppresses the duplicate acknowledgment instead of forwarding it to the fixed host. Snoop also uses timeouts to locally retransmit segments, if needed. These timeouts are less coarse than the de facto TCP timeout , and therefore expire sooner, thus leading to a local retransmission within the time span of a TCP timeout. Snoop utilizes the ability of a LL protocol to respond fast, and at the same time uses the available information to keep TCP "happy"

with the existing network connection. However, it should be noted that TCP-level retransmissions do happen, mostly due to timeouts at the sender.

TCP Reno with snoop has been shown to achieve a lot better performance than, both TCP Reno alone, and other end-to-end protocols [9]. Moreover, the authors demonstrated that the use of a selective acknowledgement scheme improves performance. Snoop has been tested mostly under scenarios involving an environment composed of a two-hop path with a wireless LAN at the receiver side. The authors also made tests in an environment composed of the wired WAN and a wireless LAN: the connection path included a 16-hop route inside a wired WAN, with the 17th hop being the wireless LAN. In this configuration, the proportion of the total RTT that is due to the wireless part of the connection path is smaller than in the two-hop scenario.

One of the key aspects of snoop's superior performance is the ability to respond to losses faster than TCP. However, snoop was designed for small RTTs in the wireless part of the path so it may not achieve superior performance when these RTTs are large. Indeed, Sinha et al. found that TCP NewReno [24] with snoop does not yield better performance in wireless WAN scenarios [56]. As explained in the *Appendix*, wireless WANs suffer from large RTTs and low data rates, two factors that are important for snoop to work efficiently. Essentially, the proportion of the total connection RTT that is due to the wireless path is very high, so snoop cannot react to losses fast enough to prevent TCP retransmissions: LL and TCP retransmissions interfere, leading to worse performance.

2.1.2 TCP-unaware LL protocols

2.1.2.1 TULIP

The Transport Unaware Link Improvement Protocol (TULIP) [44] was designed for half-duplex wireless channels with limited bandwidth. TULIP is not aware of the transport protocol per se, but it is aware of the type of service requested (e.g. reliable service for TCP vs. unreliable for UDP). In other words, TULIP requires the network protocol (in this case IP) to indicate if a particular packet requests a reliable packet delivery service or not. TULIP is indeed unaware of the TCP workings, however it was designed with TCP in mind. For example, TULIP attempts to prevent the TCP sender from receiving 3 duplicate acknowledgements by delivering only in-order incoming frames to IP.

Like snoop, TULIP locally buffers packets and uses an ARQ scheme to attempt to recover from losses on the wireless link, before the TCP sender times out. TULIP's timers rely on a maximum propagation time in the wireless channel. An interesting point is that TULIP does not offer a reliable delivery for TCP acknowledgements. However, the authors do not specify how a TCP-unaware LL protocol is able to differentiate between pure TCP acknowledgements and TCP segments. The developers of TULIP show that, at least over half-duplex radio links, a TCP-uncoupled LL solution is indeed possible and can yield better performance than TCP-aware LL protocols [17].

2.1.2.2 Delayed duplicate acknowledgements

Delayed duplicate acknowledgements (DDA) [67] is a LL TCP-unaware proposal that attempts to mimic the workings of snoop. The principle is the same as with snoop, i.e. wireless losses are detected at the BS and lost segments are retransmitted locally. The

authors specify that each TCP data segment must be encapsulated in a single LL packet, and each TCP ACK should be encapsulated in single LL ACK. DDA uses different sequence numbers for its packets than does TCP for its segments. The protocol does not attempt to deliver the packets in-order to higher protocols.

A loss is detected at the LL when a LL-duplicate acknowledgement (which encapsulates a TCP ACK), reaches the base station. The BS delays the sending of the duplicate ACKs to the TCP sender by an amount of time *d*. At the same time it retransmits the lost segment locally. Further duplicate ACKs are also delayed. If an ACK arrives indicating that the retransmitted packet has been received, then the duplicate acknowledgments are not sent on to the TCP sender. If the timer *d* expires then all duplicate ACKs are released and, probably, the TCP sender goes into Fast Retransmit. This scheme works well, according to the authors, in cases where snoop works well too, thus it will not perform well on slow wireless links. The protocol is still in its infancy and a number of issues remain to be resolved, such as the optimal value for *d*. Nevertheless, it seems that under certain conditions it can perform well, almost equivalently to snoop.

2.2 Split connections

Proposals in this category, such as Indirect TCP (I-TCP) [6], came out very early. The basic idea here is that since we have two completely different classes of subnetworks (namely wired and wireless), we could split each TCP connection into two connections at the point where the two subnetworks meet, i.e. at the base station. The BS keeps one TCP connection with the fixed host, while at the same time it uses another protocol designed for better performance over wireless links for the mobile host. The BS is entitled to acknowledge the segments as soon as it receives them [6]. However, this means that it is possible that the acknowledgement for a particular segment to arrive at the sender before the segment actually reaches the recipient [9]. Obviously this violates TCP semantics.

I-TCP does not handle handoffs efficiently, since the TCP state must be transferred between base stations. According to [7], handoffs may take several hundreds of milliseconds to be completed, thus leading to degraded TCP performance [16]. Moreover, crashes in the base station result in TCP connection termination. I-TCP is also not suitable for cases where the wireless link is not the last part of a connection path, because we could end up splitting a particular connection several times if different combinations of subnetworks are involved, leading to performance degradation. M-TCP [14] also splits TCP connections but preserves TCP semantics. M-TCP attempts to combat the effect of long periods of disconnection in TCP performance; it requires a LL protocol to recover from losses in the wireless link.

2.3 Wireless Application Protocol

The Wireless Application Protocol (WAP) is an attempt to define an industry-wide specification for developing applications that operate over wireless networks [69]. The WAP Forum [70] decided not to define a specification aimed for a particular set of actual devices, but rather to create a protocol stack and application environment that can allow a broader class of devices to communicate efficiently over wireless networks.

WAP has appeared in the popular press as the “wireless Web.” However, this is not true, not only because other “wireless Web” solutions have been proposed, but also because this goes against the very philosophy of WAP, at least in the current incarnation.

Mann [33] notes that the "WAP user paradigm is one where users are able to make small, specific, focused requests for small chunks (typically less than 1000 bytes) – not browse random Web sites."

Although the first generation of WAP is more or less targeted to "smart" mobile phones, PDAs, two-way pagers and other devices are not excluded. In fact, even more powerful devices, like laptop computers, can use WAP. WAP assumes that the mobile device has limited processing power and memory, and is connected to a network with rather limited data rates (typically less than 10Kbps) and high latencies. The network is assumed to be unreliable [33]. Moreover, WAP was designed with power consumption in mind: the target devices are assumed to have limited battery lifetime, so every effort must be made to conserve energy. Other device characteristics are small screen size and limited data entry capabilities.

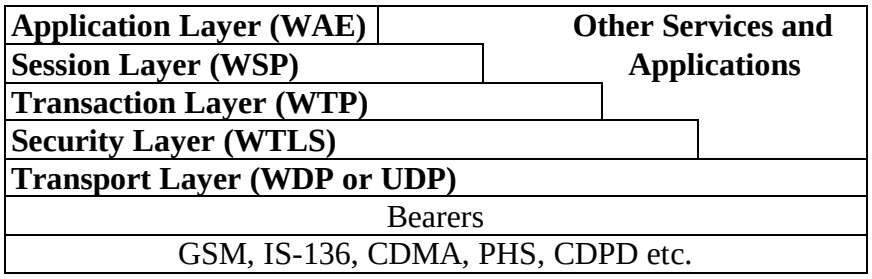

Figure 1 WAP architecture [69].

WAP introduces a new protocol stack (Figure 1). WSP, the Wireless Session Protocol, provides functionality similar to HTTP/1.1, including long-lived sessions, and the ability to suspend and resume sessions [69]. WDP, the Wireless Datagram Protocol, is the transport layer protocol of the architecture and aims at providing a consistent network service to the higher layers, independent of the data-capable bearer services of the different underlying network types. WDP is used when the bearer network does not support IP and provides an unreliable datagram service. If the network supports IP, then UDP is used instead. Clearly, WAP follows an approach whereby transfer reliability is added on top of an unreliable datagram service. Hence, much of TCP's functionality is positioned at the presentation and application layers. This choice imposes certain limitations, but using a lightweight transport protocol such as UDP or WDP has its merits, e.g. implementation simplicity and smaller memory (ROM) requirements.

Another key aspect of WAP is that all connections between a mobile host and a fixed one have to go through a proxy, the WAP gateway (Figure 2). The proxy uses WAP protocols to communicate with the wireless device, and standard TCP/IP to communicate with the origin servers. A typical scenario involves a user issuing a request on his device. The device will use the WAP stack of protocols to communicate with the proxy. The proxy will receive the request, translate it into an HTTP request and use a standard TCP/IP wired network to connect with the origin server. The WAP gateway is responsible for encoding the responses of the origin server into a compact binary format and conveying it to the mobile device using WAP protocols. WAP uses binary formats, instead of, say, the standard text formats of HTTP in order to minimize the amount of

data that has to be transmitted, and limit the processing that needs to be done at the mobile host. This way less power is expended for each request.

WAP is intended to be as compatible as possible with the existing Internet, given the device limitations mentioned above. Potentially, one could foresee a merging of the WAP stack with TCP/IP. We shall not delve into any more details of WAP; the interested reader can find a wealth of information in [33][69] and at the WAP Forum web site [70].

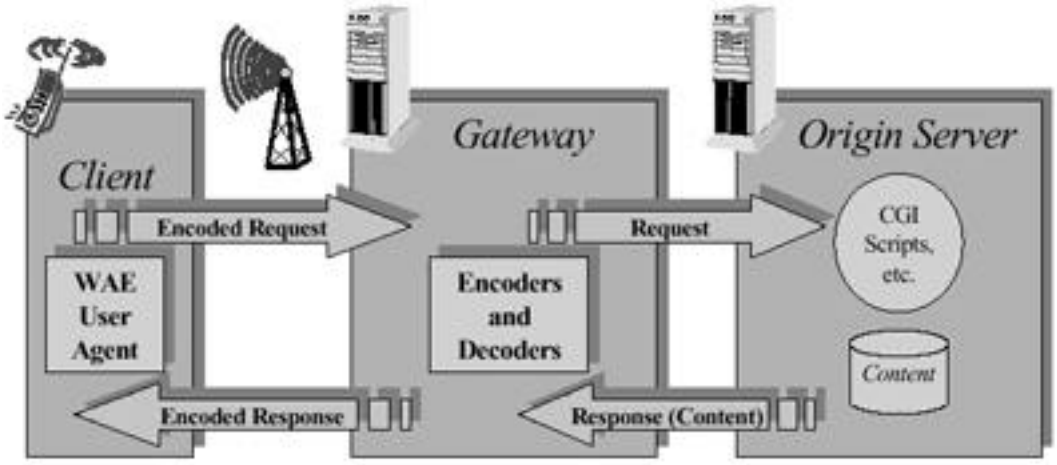

Figure 2 WAP programming model [69]

2.4 TCP modifications

2.4.1 TCP SACK

TCP selective acknowledgments options (TCP SACK) [34] were proposed as a means to alleviate TCP's inefficiency in handling multiple drops in a single window of data [22][63]. TCP SACK is not currently part of the TCP specification [50][5], but seems to be implemented in a number of popular operating systems, including Microsoft Windows 98 [49]. The TCP SACK specification states explicitly that standard congestion control algorithms, including TCP timeouts, are not affected by this proposal. Using selective acknowledgements is optional, and the two communicating parties have to negotiate during the connection setup of whether SACK is to be supported.

In contrast with the standard cumulative TCP ACKs, a SACK can convey information to the sender about multiple non-contiguous blocks of successfully transmitted data segments. TCP SACK allows the sender to avoid retransmitting segments whose successful delivery at the other end is not evident from the cumulative ACKs that arrive at the sender. TCP SACK kicks in only when 3 duplicate ACKs arrive at the sender. At this point of time, the sender can skip the retransmission of all SACKed segments, and retransmit the first unacknowledged, un-SACKed data segment. Therefore, SACK will not improve the performance in cases where the sender window size is not sufficiently large to allow for at least 4 segments in flight. This problem arises in cases where the delay×bandwidth product is rather small, or after a number of consecutive

segment losses which cause the congestion window to shrink [46]. After a retransmission timeout occurs, all SACKed segments are now considered "un-SACKed", and the sender must retransmit the oldest outstanding segment in the sending window [34].

Although TCP SACK has been shown to achieve better performance than standard TCP under certain scenarios [22][9], there are a number of cases for which TCP SACK does not offer any significant performance improvement. For example, Balakrishnan et al. reported that for certain traces TCP Reno enhanced with SACK avoided only 4% of the retransmission timeouts because of the small congestion windows [10]. Particularly for mobile hosts, it has not yet been demonstrated that the SACK-introduced complexity, which means more power consumption, is worth implementing. Even for wired-only networks different studies have reached different conclusions, so a consensus has not yet been achieved on the use of SACKs.

2.4.2 TCP FACK

TCP forward acknowledgment (TCP FACK) [35] attempts to decouple the TCP congestion control algorithm from data recovery. The aim in TCP FACK is to estimate more accurately the amount of transmitted but unacknowledged data in the network, and hence make more intelligent decisions about the data that should be (re)transmitted. TCP FACK is designed to complement SACK in achieving superior performance. It introduces a sender variable that keeps track of the "forward-most" data which has arrived at the receiver (*snd.fack*). In addition, the sender must keep track of the amount of retransmitted data (*retran_data*). Based on these variables and the information stored in standard TCP sender variables, the sender is able to accurately estimate the amount of outstanding data in the network (*acwnd*) [35]:

$$acwnd = snd.nxt^{2} - snd.fack + retran\_data$$

TCP FACK has been shown to perform better than TCP Reno, and TCP Reno with SACK under certain conditions [35]. However, it was never really tested for wired-cum-wireless environments, and is more or less targeted towards improving TCP's performance when losses are due to congestion rather than random losses.

2.4.3 TCP Santa Cruz

TCP Santa Cruz [45] includes new congestion control and error recovery strategies. It is a protocol that was designed with transmission-media heterogeneity in mind, and can handle well asymmetric and/or lossy links, limited bandwidth and variable delays. TCP Santa Cruz keeps for each segment an entry containing the time the segment was sent by the sender and the time it was received by the receiver. Thus, the sender can calculate the time interval, $S_{j,i}$, between the transmission of segment j and some other, preceding segment i, and the inter-arrival time, $R_{j,i}$, of the corresponding data packets at the receiver. Based on this information the protocol can deduce whether congestion is building up, or network conditions are improving, or staying steady [45]. TCP Santa Cruz proposes changes in the Congestion Control, Slow Start, and retransmission algorithms in order to take advantage of the increased amount of information that is available to the sender. The growth of the congestion window is decoupled from the number of returned

[2] Next sequence number to be sent [50]

ACKs. Moreover, the protocol uses a selective acknowledgements scheme to improve performance for multiple losses in a single window of data.

TCP Santa Cruz performs better than TCP Reno and Vegas under certain simulated scenarios [45], which include file transfers over a 3-hop path with a bottleneck link, and a one-hop transfer over an asymmetric link [30][45]. However, the authors do no address the issue of how to calculate accurately $S_{j,i}$ and $R_{j,i}$ when the receiver uses delayed ACKs. In addition, the relative complexity introduced raises power consumption issues. On the other hand, these complexities are mostly introduced in the sender, which for most practical purposes could mean limited additional power consumption for mobile hosts.

2.4.4 Changes to other TCP mechanics

2.4.4.1 ACK generation TCP strategies

ACKs returning from a receiver play multiple roles in TCP. They are used for reliability purposes (as part of the sliding window algorithm), for increasing the sender's transmission rate, and for congestion control. According to [5], during slow start a TCP sender increments the congestion window (`cwnd`) by at most one Sender's Maximum Segment Size (SMSS) for each ACK received that acknowledges new data. For example, a cumulative ACK for two segments and a cumulative ACK for one segment, both acknowledging new data, would cause the same increase in `cwnd`, possibly one SMSS.

According to the original TCP specification [50], receivers acknowledge every incoming segment. On the other hand, the latest RFC on TCP congestion control [5], states that a TCP receiver *should* [13] use the delayed ACKs algorithm. The delayed ACKs algorithm [12] gives the receiver the option to delay an ACK, if it is for an in-order segment, but must acknowledge every other full-sized segment. Of course, this can happen only if the receiver does not have any data to send back to the sender, and thus does not piggyback ACKs. Moreover, the receiver must send an ACK within 500 ms of the arrival of the first unacknowledged segment. In the event of an out-of-order incoming segment, the receiver must immediately send a duplicate acknowledgement. Most current TCP implementations do indeed implement the delayed ACKs algorithm. Delayed ACKs save network resources and, in the case of a battery-operated wireless host, energy expenditure.

Allman has studied [3] the effect of the delayed ACKs algorithm, as well as other schemes, for wired networks where drops occur only due to congestion. The implications of random losses due to fading wireless channels and the effect of a limited bandwidth/longer RTTs environment have not been studied with respect to the different ACK generation algorithms. Nevertheless, we present Allman's conclusions here because the proposed algorithms may be of some use in the wired-cum-wireless domain.

First, the delayed ACKs algorithm essentially doubles the amount of time a TCP sender spends in slow start. This means that an even larger proportion of short flows (see Section 1.2) spend most of the time in slow start, without fully taking advantage of the network bandwidth. Alternative mechanisms studied include acknowledging every segment, Delayed ACKs After Slow Start (DAASS), unlimited byte counting and limited byte counting [3].

Acknowledging every segment is more aggressive than using delayed ACKs, and achieves slightly better performance in terms of throughput [3]. In particular, for short

flows the improvement is 28%. On the other hand, acknowledging every segment leads to more segment losses, because the TCP sender is instructed to be more aggressive, which often causes congestion build-up. Moreover, as was demonstrated in [46], acknowledging every segment leads, at least in a number of scenarios, to worse performance: First, because the sender is instructed to undertake unnecessary segment transmissions; and second, because the receiver consumes twice the power and bandwidth for the transmission of ACKs.

DAASS attempts to limit the amount of time spent in slow start by employing the ACK-every-segment algorithm while in slow start and using the delayed ACKs algorithm only during congestion avoidance. DAASS also improves TCP throughput, and achieves exactly the same results for short flows with acknowledging of every segment mentioned above, since in this case the algorithms are the same. DAASS causes less congestion build-up than acknowledging every segment, but more than the standard TCP delayed ACKs algorithm [3].

Unlimited byte counting (UBC) uses the number of bytes acknowledged as the metric for the expansion of `cwnd`: `Cwnd` is increased by as many bytes as are acknowledged. This decouples the window expansion from the acknowledgment generation algorithm at the receiver. However, it proves to be too aggressive and may lead to worse performance. On the other hand, limited byte counting (LBC), where the total increase in `cwnd` limited to 2 SMSS, seems to improve the performance, but not as much as acknowledging every segment and DAASS.

As a last word we would like to note that other researchers have proposed to lower the TCP retransmission threshold from 3 duplicate acknowledgments to 2. This proposal stems from the fact that for a wide category of networks the bandwidth×delay product is not enough to allow for at least 4 SMSS segments to be unacknowledged. Consequently, the number of outstanding segments is not sufficient to permit Fast Retransmit to kick in, and losses are detected only by means of a time out. For example, Lin and Kung reported that 85% of the timeouts found in a set of Internet traces were not preceded by a Fast Retransmit [32]. Lowering the retransmission threshold may solve this problem. Reference [46] shows that lowering the retransmission threshold does not lead to better performance in all cases.

2.4.4.2 Increase TCP's initial congestion window

Allman et al. [4] proposed the increase of the initial congestion window (IW) to more than one SMSS. Simulation studies have shown that this modification can lead to better performance (up to 25%) [4], especially for short flows. Concerns were raised, however, regarding the increased possibility of dropped segments, in particular at congested routers or links. Shepard and Partridge [54] studied the case where a TCP receiver is connected to the Internet over a 9.6 Kbps link, through a router with enough buffer space to accommodate only 3 segments, each containing 1024 bytes. When the sender uses an IW of 4 segments it is guaranteed that it will experience a drop in this initial phase of the connection. The study concluded that the performance achieved when IW is one segment is no better than when it is four segments. Nevertheless, the last RFC defining TCP's congestion control [5] specifies that "the initial value of `cwnd`, MUST be less than or equal to 2×SMSS bytes and MUST NOT be more than 2 segments". The members of the IETF TCP Implementation Working Group opted for a gradual increase in IW since

empirical data from actual networks on the effect of increasing IW were not available at the time.

Increasing the initial congestion can be advantageous for short flows and connections that exhibit large propagation delays. Note that these changes are aimed only for the initial congestion window: The loss window, i.e. the window used after a loss has been detected, and the restart window, which is used when the sender restarts the connection after an idle period are not affected by this proposal. The specified value for these windows is one SMSS.

2.4.4.3 Explicit Congestion Notification

As pointed out earlier, TCP Congestion Control is mainly governed by the detection of lost or dropped segments. When a sender detects segment losses, it assumes that this is due to congestion and that it therefore should lower its sending rate. This approach is based on the fact that until recently routers did not provide feedback to the sender, and used a rather simple queuing strategy, namely drop tail: when a router runs out of buffer space, the latest incoming segments are dropped. TCP with Explicit Congestion Notification (ECN) calls for more functionality from the routers [23]. The routers should inform the TCP sender of incipient congestion, signaling that the sending rate should be lowered. This way, the sender is informed on time about congestion build-up and segment drops can be avoided.

By explicitly signaling out when congestion is building up, ECN can also be used for wired-cum-wireless environments. If the entire internetworking infrastructure were capable of conveying ECNs, a mobile host could implicitly assume that drops in the absence of ECNs must be due to random losses, and hence that congestion control algorithms should not be employed, but rather that the sending rate should be sustained. ECN is also power conserving, because ideally no segments are lost, except for the ones lost in the wireless path. ECN can also lead to improved overall TCP performance by avoiding retransmissions and timeouts [23]. We are currently reviewing the potential gain from the use of TCP and ECN in a wired-cum-wireless environment.

2.4.4.4 Explicit Loss Notification

In contrast to ECN, Explicit Loss Notification (ELN) was proposed for wireless networks [9]. Instead of implicitly deducing when a drop is due to random losses, it is preferable to be informed explicitly of a loss resulting from errors in a wireless link. If it were possible to explicitly indicate when a particular loss is due to errors induced by the wireless link, then TCP could be modified so as to refrain from going into congestion avoidance. However, such a scheme is very difficult to implement and, to our knowledge, no efficient solution has appeared in the literature.

2.4.4.5 Fast Retransmits

Caceres and Iftode discuss [16] TCP's degradation in performance when handoffs occur as a mobile user roams, e.g. inside a building. The authors have experimented with microcellular networks, i.e. wireless LANs with cells of a few meters in diameter. These networks offer a good raw bit rate (at the time the paper was published that was 2 Mbps for IEEE 802.11; nowadays, with IEEE 802.11b, it can be up to 11 Mbps). While

bandwidth is not the main issue in these networks, handoffs can have a big impact on TCP throughput, and therefore on the mobile user's experience.

The authors explore situations where the cells are overlapping, adjacent, or otherwise. TCP throughput suffers in all these cases: even in the case of overlapping cells throughput is less, though by a mere 6%. The case where the handoff needs one second to complete leads to complete loss of communication at the transport level for a period of 2.8 seconds.

During this period, TCP loses segments on the forward path and acknowledgments on the return path. The absence of acknowledgments means that Fast Retransmit will not kick in, and so the TCP sender must wait for a timeout. The authors propose changes to Mobile IP at the mobile host so that Mobile IP informs TCP when a handoff is completed. Then, the receiver's TCP sends duplicate acknowledgements to initiate Fast Retransmit at the sender TCP. Under this approach the sender does not have to wait for a timeout, and takes full advantage of Mobile IP's knowledge about handoffs. However, the Fast Retransmits approach aims only at dealing with handoffs and does not handle losses due to wireless channel fading.

2.5 New transport protocols

2.5.1 Wireless Transmission Control Protocol

The Wireless Transmission Control Protocol (WTCP) [56] is designed to provide a reliable transport protocol for CDPD [1][58], and in general for wireless WANs (WWANs) that exhibit low bandwidth and high latencies. WTCP is used when a mobile host needs to connect through a proxy to access information. Sinha et al. [56] note that this is the case for most current deployments of WWANs. Hence, WTCP is a proposal that attempts to solve the problem of transmission-media heterogeneity in a "split connections" fashion.

WTCP uses algorithms similar to those of standard TCP for connection management and flow control. However, it follows a different approach with regard to congestion control. First, WTCP is rate-based and the rate control is done at the receiver. That is, the receiver is responsible for setting the appropriate rate for the sender to use. The receiver, upon the arrival of each incoming segment, employs an algorithm [56] to determine whether to ask the sender to increase, decrease or keep the same sending rate. This information is conveyed to the sender through cumulative ACKs. Transmission control in WTCP is governed by the inter-packet delay as it is measured at the receiver.

Second, WTCP attempts to predict when a segment loss is due to transmission errors or due to congestion. In short, while the network is deemed to be uncongested, the receiver keeps statistics for non-congestion related segment losses. Based on this information, the receiver signals the sender to continue transmitting with the same rate if the loss is estimated to be due to transmission errors, or to decrease the sending rate if congestion is deemed to be the cause of these losses. This mechanism, though promising, has not yet been exhaustively tested.

In contrast with TCP, WTCP does not use an ARQ scheme to assure reliability of transfers. The authors believe that one of the main reasons for the sub-optimal performance achieved by TCP in WWANs is due to the inaccurate retransmission timeout estimates. In order to alleviate this inefficiency WTCP employs a scheme with

SACKs and probes. The receiver-generated acknowledgements act as cumulative ACKs but also carry SACK information. Thus the sender can verify whether certain segments were received and if not, it can retransmit the missing ones. However, ACKs are always generated in response to incoming segments. If the sender does not have any more segments to send, and at the same time it has not received a confirmation that all outstanding segments were received, then it will use probe segments to elicit receiver ACKs.

WTCP was shown [56] to perform better than TCP NewReno [24] and TCP Vegas [48], at least under certain scenarios. Be that as it may, the receiver in WTCP is a lot more complicated than in TCP; this could lead to increased power consumption, since usually the mobile host plays the role of the receiver.

3 TCP performance tradeoffs

As noted earlier, TCP segment losses trigger congestion control algorithms at the sender. In order to understand the impact of these algorithms on TCP performance, we experimented with three different TCP versions: Tahoe, Reno, and NewReno [5][24]. The reader interested in the methodology and details of these experiments is referred to [46][63]; we will not delve into these here, but simply present the results and comment on their significance.

We first explored the impact of random losses on TCP performance. In [46] we showed that even light error conditions can lead to significant performance degradation. For example, when we introduced random segment drops with probability 1.16%, i.e. one segment in 86, on the average, was dropped (column named R86 in Table 1), the total connection time to complete a 5-MB file transfer was increased by approximately 47%. Note that this error level is equivalent to a 10^{-6} BER. Another way to view the results presented in Table 1 is to consider the amount of time that the network is error-free. When the network corrupts segments only 2% of the time (column R50 in Table 1), the total connection time to complete the 5-MB file transfer increases by 75% to 92.5%.

Table 1
Total transmission time in seconds to complete a 5-MB file transfer under random errors in a LAN-like environment [46].

	Error free	**R86**	**R50**	**R20**	**R10**
Tahoe	40	58.220	70.005	137.418	282.548
Reno	40	58.925	77.186	118.525	257.807
NewReno	40	58.755	77.648	136.428	272.414

As noted in [46], the proportion of time that the channel is impaired is only one factor affecting TCP performance. The timing of a segment loss can also be very important. The loss of two relatively closely spaced segments causes a chain reaction. First of all, the congestion window shrinks down to 25% of its existing size, i.e. the size before the first loss. If these two losses were coincidental, and do not reflect congestion build-up, TCP will not be able to fully utilize the available bandwidth for some considerable amount of time. In [63] we argued that Tahoe's ability to outperform the other two versions under certain scenarios is due to the use of Slow Start after Fast Retransmit instead of

Congestion Avoidance. Slow Start probes the network for available bandwidth more aggressively than Congestion Avoidance, and thus a Tahoe sender can recover faster from losses.

Moreover, in many instances the second segment loss will not be detected by Fast Retransmit but by means of a timeout. Consequently, the sender will spend a considerable amount of time waiting for the timeout to occur without transmitting anything, yielding under-utilization of the network connection.

In [46] we showed that TCP handles burst errors more efficiently than random errors. For example, TCP performed almost the same under a scenario that involved random errors corrupting segments 2% of the time, and under a scenario that involved burst errors corrupting segments 10% of the time. TCP interprets all errors as being congestion-related and responds accordingly. This happens to be a more effective response to burst than random errors.

3.1 The energy-throughput tradeoff

In [63] we discussed the energy-throughput tradeoff of the aforementioned TCP versions. It is important to note that power consumption depends on the device hardware, the operating system, and the application being run. For example, new processors, like Crusoe [62] and operating systems for handheld devices, e.g. PalmOS [43], are designed with power efficiency in mind. Furthermore, modern operating systems, such as Microsoft Windows [38] and Linux [61], are being modified in order to waste less energy, by avoiding power hungry operations.

However, the power efficiency of a reliable transport layer protocol is important too, and has not been thoroughly examined in the literature. Analytical studies like [75] have yielded different results than experimental ones [63]. No TCP version has a clear-cut performance advantage under all scenarios: we used a wide set of experimental scenarios in [46][63], but were unable to explicitly point out a best "overall" TCP version. Instead it became clear that under light error conditions TCP Reno performs better than the other two, while, as conditions become more severe, the "conservative" approach of Tahoe (in the sense that it always applies Slow Start rather than Fast Recovery) achieves superior performance.

Most TCP performance studies have focused on throughput. Although it may seem that achieving better throughput is congruent with energy efficiency, this is not always the case. Some protocols attempt to achieve better throughput by being more aggressive. For example, NewReno was shown to outperform other TCP versions [22] (under certain scenarios). However, as was demonstrated in [63], being more aggressive with retransmissions can lead to excessive power consumption for marginal throughput gains. The transport protocol can potentially be "trapped" in a bad phase of the communication and attempt to retransmit over and over again without success: all retransmitted segments will be corrupted and the sender will waste power.

Lost segments can also lead to retransmission timeouts, which induce minimally sized congestion windows. In addition, the sender may suffer from erroneously calculated timeout periods because of the lack of feedback from the receiver. This in turn can lead to inability to detect when bad periods in the communication terminate and so degrade performance even more. Ideally we would like TCP to immediately sense when bad periods start and terminate so that precious energy is not wasted with retransmissions

during the bad periods, and good periods are fully taken advantage of. This is the case, for instance, in FM radio transmissions: an automobile radio receiver may loose "communication" with the radio station while, say, the vehicle passes through a tunnel, but it resumes receiving the radio signal immediately upon exit from it. Unfortunately, the equivalent in data communications cannot be achieved with current TCP versions.

One may assume that a TCP sender can enter a vicious cycle where it skips all the "good" periods and interprets random losses as indicative of relatively prolonged bad, congestion periods, which lead to significant degradation in performance. However, in our tests the overhead introduced by TCP, as measured by the total amount bytes transmitted over and above the 5-MB application level payload, is not unreasonable (Figure 3). In other words, one may say that TCP is power saving, in the sense that it does not retransmit segments excessively. On the other hand, TCP is not capable of detecting in an efficient manner good transmission periods, and even when it does, it cannot fully utilize them. Hence, the combination of these factors leads to prolonged communication times (Figure 4 and Figure 5).

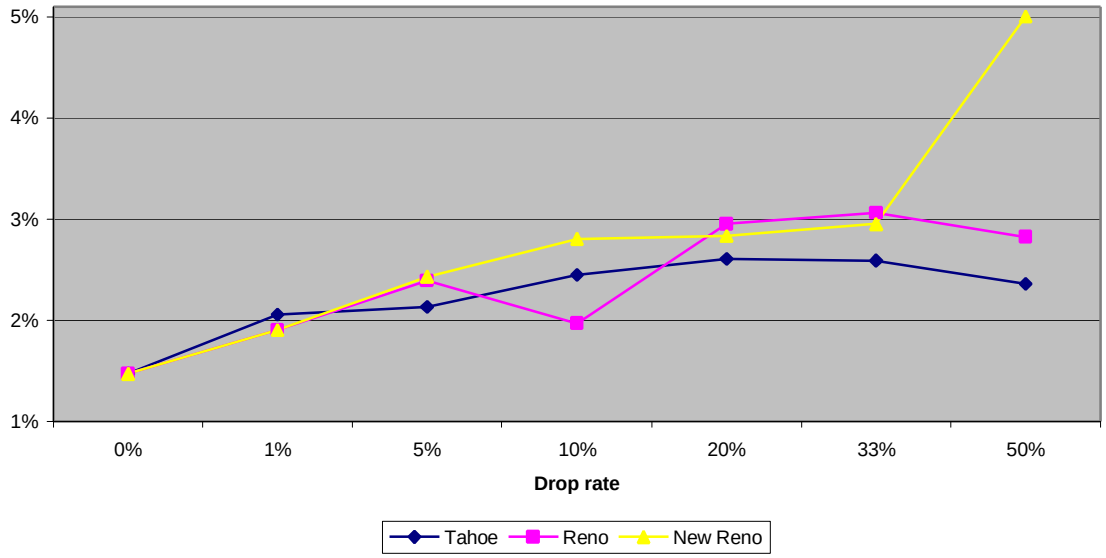

Figure 3 TCP overhead incurred for a 5-MB file transfer; ten second ON/OFF dropping phases [63].

In the following sections we present the energy-throughput tradeoff for short (one second) and prolonged (ten seconds) ON/OFF dropping phases.

3.1.1 Short and frequent error phases

We tested the three TCP versions under scenarios with varied drop error rates, using a model that provides error free conditions with a mean time of one second, and introduces errors at the specified error rate for an equal mean time (see [63] for more details). Figure 4 presents our results.

Under light error conditions the three versions perform almost the same. As the error rate increases, no TCP version performs consistently better than the others. Essentially what happens is that, due to the frequent errors, the congestion window does not get the

opportunity to expand significantly, and so in many cases there are not enough outstanding segments to allow for duplicate acknowledgements to trigger Fast Retransmit. Hence, many of the losses are detected by means of a timeout. Under these conditions, TCP's performance is mostly governed by the exact sequence of error occurrences.

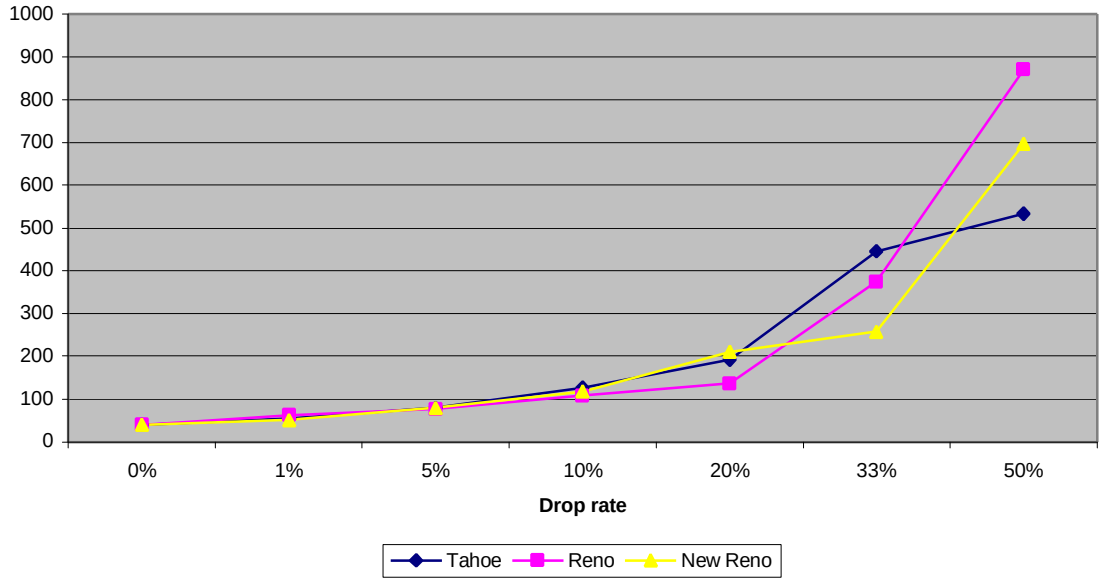

Figure 4 Total transmission time in seconds to complete a 5-MB file transfer; one second ON/OFF dropping phases [63].

3.1.2 Long and persistent error phases

We also tested the three TCP versions under the same set of error rates but with an increased mean time for the ON/OFF dropping phases. We present the results in Figure 5.

Under these scenarios, the prolonged good phases allow for expanded congestion windows. Again, for small error rates there is no significant difference amongst the three versions. However, as the error rates increase the conservative behavior of Tahoe seems to perform better. First of all, Reno is known not to perform well when more than one loss occurs in a window of data [28]. With prolonged error phases it is virtually guaranteed that there will be multiple segment losses in a single window of data; thus, it is clear why Reno does not perform better that the others. NewReno is performing quite well in all scenarios except for the case where the error rate is set to 50%. NewReno handles better than Reno multiple segment losses. Unfortunately, it does so at the expense of energy, attempting to retransmit several times while the bad phase persists, thus getting a larger number of segments corrupted, which lead to more overhead (Figure 3) and longer transmission periods (Figure 5).

Tahoe, on the other hand, backs off under all circumstances. After a segment loss is detected Tahoe will exercise Slow Start, which facilitates things under our scenarios. First, Tahoe does not incorrectly persist in transmitting during bad periods. After the detection of a loss, the congestion window will shrink to one segment, and most losses

will be detected through a timeout. Waiting for a timeout essentially allows Tahoe to incur fewer overheads, thus saving energy. Second, during the prolonged good periods Slow Start allows Tahoe to open up its congestion window a lot faster than Congestion Avoidance. Note that, due to the persistent errors, Congestion Avoidance is achieved at relatively small congestion window sizes. Therefore, Tahoe is able to detect the good periods and take advantage of them more efficiently than Reno and NewReno. These two factors combined allow Tahoe to complete the 5-MB file transfer faster even under severe error conditions (50% drop rate in Figure 5). Tahoe also yields less overhead for most error rates, under this scenario (Figure 3).

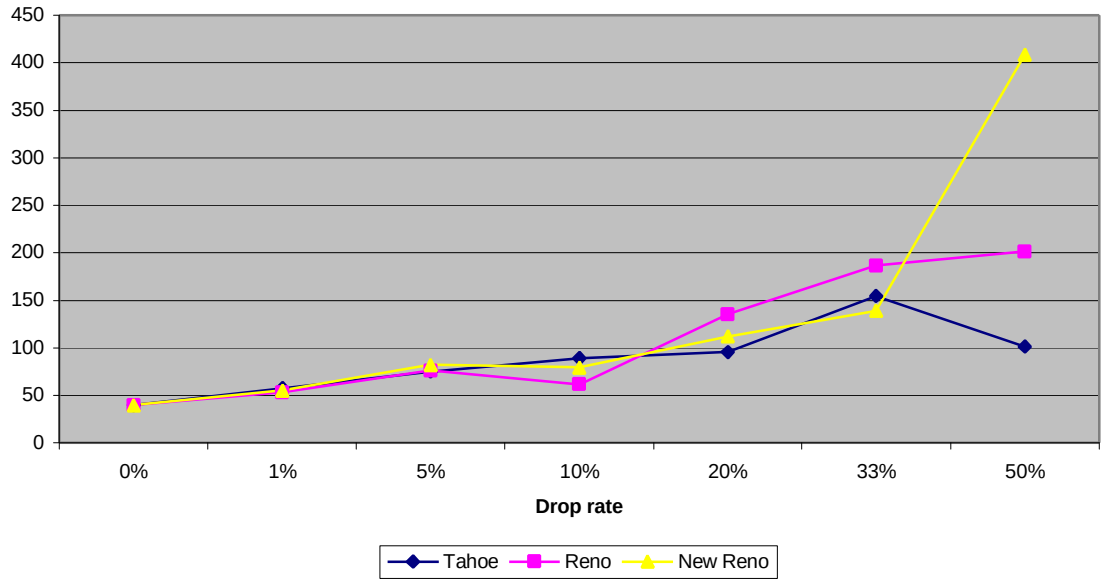

Figure 5 Total transmission time in seconds to complete a 5-MB file transfer; ten second ON/OFF dropping phases [63].

Note that for the same error rates, the total transmission time to complete the file transfer is at least double in the case of short and frequent error phases (Figure 4) than in the case of long and persistent ones (Figure 5).

3.2 The overhead-goodput tradeoff

As was noted earlier, the three TCP versions follow different segment transmission approaches when errors occur, which in turn yield different results in terms of goodput [63] and overhead. In this section we examine the overhead-goodput tradeoff, in other words, how a reliable transport protocol may attempt to trade overhead for improved goodput. Achieving better goodput is important for wired-cum-wireless networks, but comes at no expense only for wired networks. As we explained earlier, aggressiveness in (re)transmissions could mean more overhead, and thus greater power consumption.

Figure 6 presents Tahoe's overhead-goodput tradeoff for one second ON/OFF error phases. As expected, goodput decreases proportionally to the error rate, although the relationship is not linear. Notice that goodput drops sharply for small error rates. For

larger error rates, such as 33% and 50%, the loss in goodput is not so dramatic. Overhead, on the other hand, increases with higher error rates. For small error rates, overhead does not increase dramatically in terms of bytes. However, notice the significant "jump" in overhead, when the error rate increases from 20% to 33%. Figure 6 reinforces the point made earlier that TCP performance does not mainly suffer from excessive retransmissions (limited overhead incurred) during bad periods but because of the inability to quickly and efficiently take advantage of good periods, which leads to prolonged communication times.

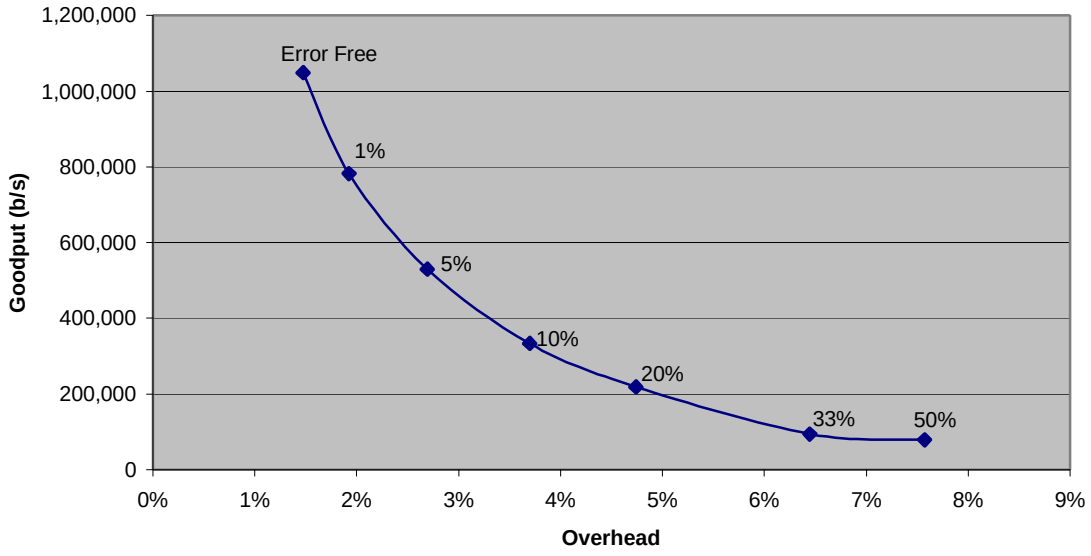

Figure 6 The Tahoe overhead-goodput tradeoff for various error rates; one second ON/OFF dropping phases [63].

Reno has a less smooth overhead-goodput curve than Tahoe (Figure 7). Although Reno seems to perform slightly worse than Tahoe when the error rate is set to 1%, it achieves a lot better performance under error rates of 5%, 10%, and 20%. In fact, Reno manages to keep the overhead almost steady under error rates of 10% and 20%, while achieving only 20% less goodput. Yet the amount of lost segments doubles as we move from a 10% scenario to a 20% one. Reno is not so successful at the two higher error rates presented in Figure 7. Under a 50% error rate, Reno's "aggressiveness" with segment retransmissions (Fast Recovery) proves not to be the best solution, yielding both higher overhead and smaller goodput than Tahoe.

NewReno manages to outperform the other two versions for the 1% error rate scenario (Figure 8). However, when the error rates increases slightly (5%), NewReno performs no better than the other two versions. Moreover, NewReno performs worse than Reno for error rates of 10% and 20%, achieving smaller goodput and larger overhead. Yet, notice that under a 33% error rate NewReno successfully trades off overhead for goodput outperforming Reno. In this particular case, NewReno proves to be the most power conserving of the three TCP versions. Moreover, NewReno performs better than both Tahoe and Reno, in terms of overhead, when the error rate is increased to 50%.

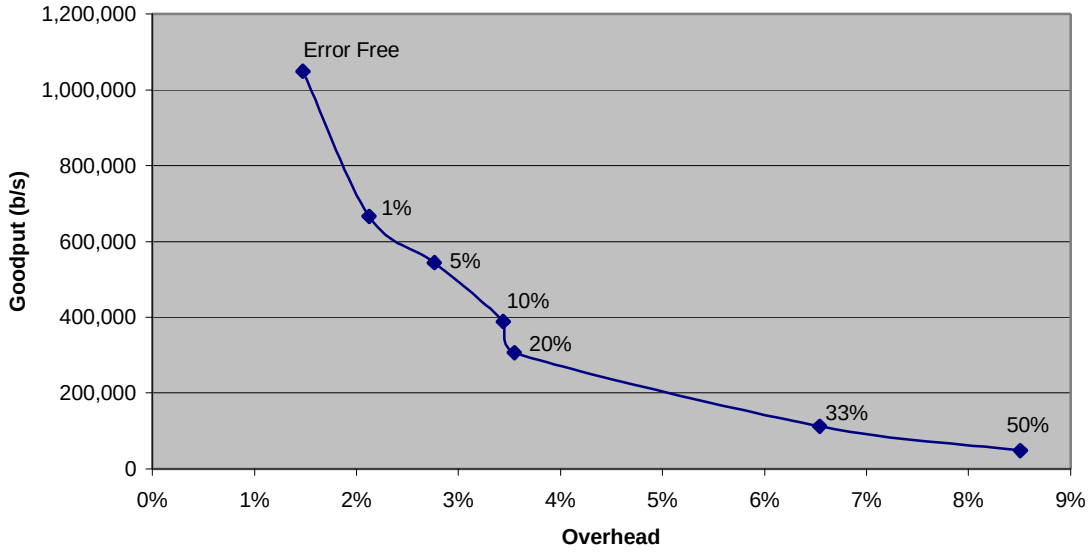

Figure 7 The Reno overhead-goodput tradeoff for various error rates; one second ON/OFF dropping phases [63].

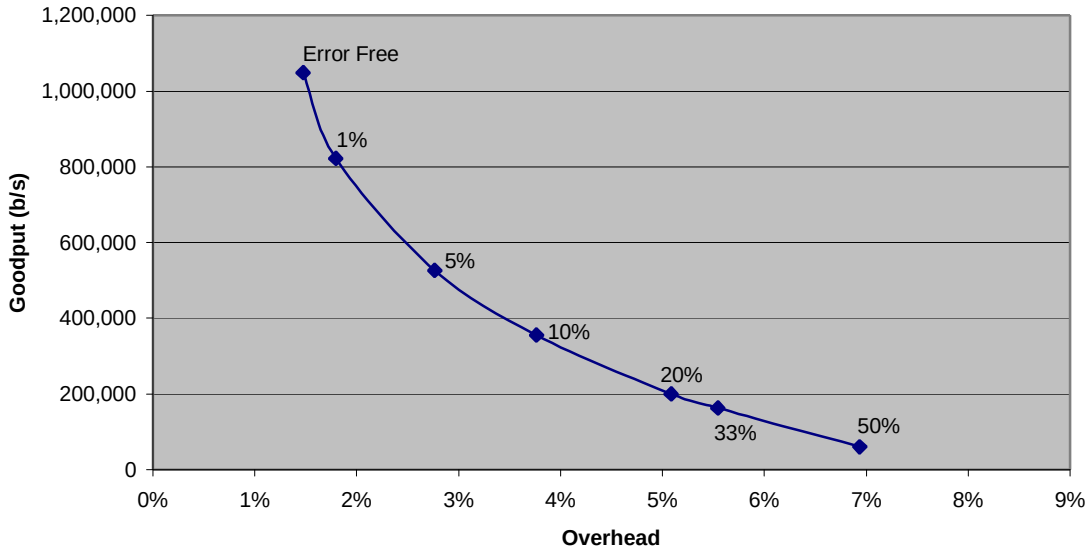

Figure 8 The NewReno overhead-goodput tradeoff for various error rates; one second ON/OFF dropping phases [63].

Although the results for one- and ten-second error phases seem to be contradictory in some respects, they are very consistent with what was noted earlier: No TCP version proves to be the "best overall". For instance, consider NewReno's overhead-goodput tradeoff for 50% error rates (Figure 3, Figure 5, and Figure 8). What proves to be a good

strategy when error phases are short and frequent is in fact the worst approach when dealing with prolonged and persistent error conditions.

4 Conclusion

This survey presents TCP performance-related issues in wired-cum-wireless environments. Although it covers several proposals, it is by no means exhaustive; proposals not discussed here include [2][18][36][40][64][65].

In summary, TCP has been shown to perform poorly in wireless environments in terms of achieved throughput due to the factors explained in Section 1.2. The research community consensus, also confirmed by our experimental results in Section 3, is that the use of "traditional" TCP in wireless environments is not a very attractive choice. While TCP is needed for a wired-cum-wireless Internet, the current version is not adequate for the task. Improving TCP performance under these conditions has been a very active area of research for the last several years. However, researchers have more or less focused on specific, ad hoc problems, and there appears to be no general solution so far. Solutions that work extremely well under some conditions perform poorly when these conditions cease to exist. For example, solutions proposed for wireless LANs do not perform well in wireless WANs and vice versa. Moreover, most of the work in this area used throughput as the main criterion for judging the merit of each proposal. New proposals will have to demonstrate their improved performance not only in terms of throughput but also in terms of power consumption in order to succeed.

Appendix: Wireless Data Networks

In Section 1.1 we introduced the various wireless networks and classified them according to coverage area into wireless LANs and WANs. In addition, computer networks, including wireless, can be classified into packet switching and circuit switching networks [48][58]. This appendix presents yet another categorization, which is based on the kind of services offered, and the supported user mobility.

Full user mobility. Wide Area Wireless Data Networks (WAWDN) include systems like the Cellular Digital Packet Data (CDPD) [1], Mobitex [58], Advanced Radio Data Information System (ARDIS) [41], and the General Packet Radio Service (GPRS) [58]. These packet-switched networks allow the user to roam almost everywhere. For example, special attention has been given in the ARDIS network[3] so that coverage includes not only open spaces but also building interiors [51]. Moreover, an ARDIS base station can cover an area of 15-25 km in diameter. The user is able to move with relatively high speeds and remain connected, though at rather slow data throughput rates (see Table 2).

Table 2
Data transfer rates provided by Wide Area Wireless Data Networks [17][59][71][72].

Wireless Network	Bit rate (Kbps)
Motient (formerly ARDIS)	4.8 &19.2
Mobitex	8
CDPD	19.2
Ricochet (by Metricom)	28.8 & 128
GPRS	up to 170

CDPD was designed to use the channels of the Advanced Mobile Phone Service (AMPS), the first generation analog cellular network, that do not carry voice traffic. These channels can be shared with AMPS, i.e. used only when there are no phone calls, or dedicated to data traffic. As CDPD gains popularity - and AMPS becomes obsolete - more CDPD dedicated channels can be allocated. CDPD can be deployed at less cost than other wireless wide area networks because it utilizes the existing hardware and software AMPS infrastructure. CDPD is based on a CSMA/CD variant called digital sense multiple access [58] and offers IP-based services [1], which is a great advantage.

Similar to CDPD, GPRS is normally embedded in a GSM network [58]. GPRS uses packet switching and is provisioned in GSM Phase 2+. Its main objective is to provide standard data transfer technologies like TCP/IP with a mobile radio network with significantly higher bit rates than other systems, in contrast to which it was also designed with office applications in mind. The standard was finalized by ETSI in late 1997 and to many it is a transitional technology towards third generation cellular networks [58][51]. Currently GPRS is being deployed in many European countries. The reader can find more

[3] ARDIS was created in 1990 when Motorola and IBM merged their bi-directional wide area data networks. In 1995 Motorola took over the 100% of the company [58]. Motient acquired ARDIS in March 1998 [41].

about WAWDNs in [1][11][51][58][60]. In addition, [72] provides a rich glossary of terms used in the wireless industry.

Cellular networks, like the Global System for Mobile communications (GSM) [60] and IS-95 [17] (commonly referred in the States as CDMA), can also be used for wireless network connectivity, though in circuit switching fashion. For example, GSM offers transmission rates of up to 9.6 Kbps (up to 14.4 in GSM Phase 2+). The designers of GSM decided to offer a reliable link layer protocol for data transmission called the *nontransparent mode*. Thus, packets are shielded from corruption using FEC and an ARQ scheme assures that lost or corrupted packets will be delivered to the receiver [11]. However, as was noted in Section 2.1, a reliable TCP-unaware link layer protocol could interfere with TCP and yield worse performance overall. Table 3 presents the available data transfer bit rates for current cellular networks. The forthcoming Enhanced Data GSM Environment (EDGE) will provide speeds up to 560 Kbps, and constitutes yet another transitional step towards the Universal Mobile Telecommunications System (UMTS).

Table 3
Data transfer rates provided by cellular networks [17][37][59][71][72].

Cellular Network	Bit rate (Kbps)
IS-95 (CDMA)	9.6/14.4
3G CDMA	156 1X / 384 3X, up to 2400
GSM	9.6
GSM Phase 2+	14.4, up to 384
EDGE	560
UMTS	up to 2048 (microcell)

The IS-95 (also known as cdmaOne [72]) cellular network standard uses Direct Spectrum Code Division Multiple Access (DS-CDMA) and was designed to replace AMPS. According to Qualcomm, CDMA offers a ten to fifteen-fold efficiency increase in comparison with AMPS [11]. IS-95 can use data transmission rates of up to 14.4 Kbps. Data services can be used simultaneously with voice traffic. Moreover, due to the advanced power control needed for CDMA [11], a mobile terminal never uses more transmission power than needed. Practical tests have shown that 98% of the time a mobile terminal transmits with less than one mW. Moreover, due to the nature of CDMA, Raleigh fading and multipath propagation do not degrade the performance considerably [11].

Third generation cellular networks, like UMTS [59][66] and 3G CDMA [17] promise a lot higher data transfer rates. However, these networks have yet to be realized on a big scale. So far as TCP is concerned, higher transfer rates can alleviate some of the inefficiencies discussed in Section 1.2. Table 4 presents the profiles defined in the International Mobile Telephone Standard 2000 (IMT-2000). Note that the maximum data rate of 2 Mbps is available only to slow moving terminals relatively close to base stations. The "Medium Multimedia" data rate is available to moving terminals in urban

and suburban areas. However, terminals in rural areas are provided with lower data rates (up to 144 Kbps). Under these restrictions UMTS will not be able to deliver the necessary high rates for certain applications.

Table 4
Service profiles defined in IMT-2000 [55].

Service	Bandwidth (Kbps)	Transmission Mode
High Interactive Multimedia	128	Circuit-switched
High Multimedia	2048	Packet-switched
Medium Multimedia	384	Packet-switched
Switched Data	14.4	Circuit -switched
Simple Messaging	14.4	Packet-switched
Voice	16	Circuit -switched

Portable wireless data. Wireless LANs offer portability, and even mobility but in a rather limited area. They offer higher transfer rates than the previous category of wireless networks (see Section 1.1). Moreover, next generation wireless LANs will provide even more bandwidth to the user (Table 5).

There has been considerable effort made to create wireless ATM networks [20] [68]. Wireless ATM can provide a reliable network, solving many of the problems described in Section 1.2. However, wireless ATM has yet to reach a final status; there are many issues that still need to be resolved before it is widely implemented [68].

Ricochet, a service from Metricom [37], offers an alternative solution, providing speeds up to 128 Kbps (Table 2). Users can access the Internet while they are in the coverage area, using a laptop or palmtop computer, and a small wireless modem. However, Ricochet does not offer full mobility, or handoff handling during online sessions. In other words, if a user tries to access the Internet while in a moving car, the connection may drop.

Fixed wireless data (Broadband Wireless). The term fixed wireless data is used for wireless point-to-point networks, which offer services to a location, such as an office or home, through larger customer-premises antennas than seen in the mobile or portable setups. Fixed wireless can offer faster data throughputs than the preceding categories, equivalent to T1 line speeds. Examples include the Broadband Radio Access Networks [21], and the IEEE 802.16 Broadband Wireless Access standards [27].

Table 5
ETSI Broadband Radio Access Network standards.

	Type	Data rates (Mbps)	Distance (m)
HIPERLAN 1	Portable/Mobile LAN	20	50-100
HIPERLAN 2	Portable/Mobile ATM	24	50-100
HIPERLAN 3	Stationary Local Loop (ATM)	48	5000
HIPERLAN 4	Point to point stationary	155	50-500

References

[1] J. Agosta and T. Russel, *CDPD: CellularDigital Packet Data Standards and Technology*. McGraw-Hill, New York, 1996.

[2] A. Aggarwal, S. Savage, and T. Anderson, **"**Understanding the Performance of TCP Pacing", in *Proceedings of the 2000 IEEE Infocom Conference*, Tel-Aviv, Israel, March, 2000.

[3] M. Allman, "On the Generation and Use of TCP Acknowledgements", in *ACM Computer Communications Review*, 28(5), October 1998.

[4] M. Allman, S. Floyd, and C. Partridge, *Increasing TCP's Initial Window*, RFC 2414, Experimental, September 1998.

[5] M. Allman, V. Paxson, and W. Richard Stevens. *TCP Congestion Control*, RFC 2581, April 1999.

[6] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", in *Proceedings of the 15th International Conference on Distributed Computing Systems (IDCS)*, May 1995.

[7] A. Bakre and B. R. Badrinath, "Handoff and system support for Indirect TCP/IP" in *Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing*, April 1995.

[8] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. "Improving TCP/IP Performance Over Wireless Networks". In *Proceedings of ACM MobiCom*, November 1995.

[9] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links", in *IEEE/ACM Transactions on Networking*, December 1997.

[10] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz, "TCP behavior of a busy Internet server: Analysis and improvements", in *Proceedings of IEEE INFOCOM*, pages 252--262, March 1998.

[11] R. Bekkers and J. Smits, *Mobile Telecommunications: Standards, Regulation, and Applications*, Artech House Publishers, 1999.

[12] R. Braden, *Requirements for Internet Hosts – Communication Levels*, STD 3, RFC 1122, October 1989.

[13] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, BCP 14, RFC 2119, March 1997.

[14] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks", in *Computer Communications Review*, volume 27, No. 5, pp. 19-43, October 1997.

[15] R. Cáceres and L. Iftode. "The effects of mobility on reliable transport protocols", in *Proceedings of the 14th Intl. Conf. on Distributed Computing Systems*, pages 12--20, June 1994.

[16] R. Cáceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", in *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 5, June 1995.

[17] CDMA Development Group (www.cdg.org), October 2000.

[18] A. Chockalingam, M. Zorzi, V. Tralli, "Wireless TCP Performance with Link Layer FEC/ARQ", in *Proceedings of IEEE ICC'99***,** June 1999.

[19] A. DeSimone, M. C. Chuah, and O. C. Yue, "Throughput Performance of Transport-Layer Protocols over Wireless LANs", in *Proceedings of Globecom '93*, December 1993.

[20] European Telecommunications Standard Institute, *ETSI HIPERLAN/1 standard*, October 2000. Available at http://www.etsi.org/technicalactiv/hiperlan1.htm.

[21] European Telecommunications Standard Institute, *Broadband Radio Access Networks* (www.etsi.org/bran), October 2000.

[22] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", in *Computer Communication Review*, V. 26 N. 3, pp. 5-21, July 1996.

[23] S. Floyd, "TCP and Explicit Congestion Notification", in *ACM Computer Communication Review*, V. 24 N. 5, p. 10-23, October 1994.

[24] S. Floyd and T. Henderson. *The NewReno Modification to TCP's Fast Recovery Algorithm*. RFC 2582, April 1999.

[25] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP 1.1*. RFC 2616, June 1999.

[26] IEEE P802.11, The Working Group for Wireless Local Area Networks, http://grouper.ieee.org/groups/802/11/, October 2000.

[27] IEEE P802.16, The Working Group for Broadband Wireless Access, http://grouper.ieee.org/groups/802/16/, October 2000.

[28] IP Routing for Wireless/Mobile Hosts (mobileip), http://www.ietf.org/html.charters/mobileip-charter.html

[29] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high delay×bandwidth products and random loss", in *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, June 1997.

[30] T. V. Lakshman, U. Madhow, and B. Suter, "Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance". In *Proceedings of IEEE INFOCOM '97*, April 1997.

[31] W. C. Y. Lee, *Mobile Communications Design Fundamentals, 2nd edition*. John Wiley and Sons, 1993.

[32] D. Lin and H. Hung, "TCP fast recovery strategies: Analysis and improvements", in *Proceedings of IEEE INFOCOM*, April 1998.

[33] S. Mann, *Programming Applications with the Wireless Application Protocol: The Complete Developer's Guide*, John Wiley & Sons, 1999.

[34] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, *TCP Selective Acknowledgment Options***.** RFC 2018, April 1996.

[35] M. Mathis and J. Mahdavi. "Forward acknowledgement: Refining TCP congestion control", in *Proceedings of the ACM SIGCOMM*, August 1996.

[36] M. Mathis, J. Semke, J. Mahdavi, and K. Lahey, *The Rate-Halving Algorithm for TCP Congestion Control*, June 1999. Available at http://www.psc.edu/networking/ftp/papers/draft-ratehalving.txt

[37] Metricom (www.metricom.com), October 2000.

[38] Microsoft Mobile Devices (www.microsoft.com/mobile), November 2000.

[39] Microsoft WebTV Networks Inc. (www.webtv.com), October 2000.

[40] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, N. Vaidya, *Long Thin Networks*, RFC 2757, January 2000.

[41] Motient Corporation (www.motient.com), October 2000.

[42] ORiNOCO RG-1000 Residential Gateway product information, Lucent Technologies, http://www.wavelan.com/products/802.11b, September 2000.
[43] Palm OS (www.palmos.com), November 2000.
[44] C. Parsa and J.J. Garcia-Luna-Aceves, "Improving TCP Performance over Wireless Networks at the Link Layer", in Mobile *Networks and Applications*, 1999.
[45] C. Parsa and J.J. Garcia-Luna-Aceves, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media", in *Proceedings of IEEE ICNP '99*, Toronto, October 1999.
[46] K. Pentikousis, *Error Modeling for TCP Performance Evaluation*. MS Thesis, Department of Computer Science, SUNY at Stony Brook, May 2000.
[47] C. Perkins, *Mobile IP Design Principles and Practices*, Addison-Wesley, 1998
[48] L. L. Peterson and B. S. Davie, *Computer Networks, 2nd edition*. Morgan-Kaufmann, 2000.
[49] Pittsburgh Supercomputing Center, *Experimental TCP Selective Acknowledgment Implementations.* Available at http://www.psc.edu/networking/all_sack.html.
[50] J. B. Postel. *Transmission Control Protocol*. RFC 793, September 1981.
[51] A. K. Salkintzis, **"**A Survey of Mobile Data Networks", in *IEEE Communication Surveys*, vol. 2, no. 3, third quarter 1999.
[52] H. Saltzer, D. P. Reed, and D. Clark, "End-to-end arguments in system design", in *ACM Transactions on Computing Systems*, vol. 2, no. 4, 1984.
[53] S. Savage, N. Cardwell, and T. Anderson. "The Case for Informed Transport Protocols", in *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, Rio Rico, AZ, March 1999.
[54] Shepard, T., and Partridge, C., *When TCP Starts Up With Four Packets Into Only Three Buffers*. RFC 2415, Experimental, September 1998.
[55] Siemens Corporation, *UMTS by Siemens -- Lexicon*, October 2000. Available at http://www.siemens.nl/umts/lexicon.
[56] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: a reliable transport protocol for wireless wide-area networks", in *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 1999, Seattle, WA USA.
[57] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.
[58] S. Tabbane, *Handbook of Mobile Radio Networks*, Artech House Mobile Communications Library, Norwood, MA, 2000.
[59] Third Generation Partnership Project (www.3gpp.org), October 2000.
[60] J. Tisal, *GSM Cellular Radio Telephony*, John Wiley & Sons, West Sussex, England, 1998.
[61] Transmeta Mobile Linux (www.transmeta.com), November 2000.
[62] Transmeta, *The Technology Behind Crusoe™ Processors*, May 2000. Available at http://www.transmeta.com/crusoe/download/pdf/crusoetechwp.pdf.
[63] V. Tsaoussidis, H. Badr, K. Pentikousis, X. Ge. "Energy/Throughput Tradeoffs of TCP Error Control Strategies", in *Proceedings of IEEE Symposium on Computers and Communications*, France, July 2000.

[64] V. Tsaoussidis, H. Badr, R. Verma, "Wave and Wait Protocol (WWP): Low Energy, High Throughput for Mobile IP-Devices", in *Proceedings of the 8th IEEE Conference on Networks,* September 2000.

[65] V. Tsaoussidis, H. Badr, "TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains", in *Proceedings of the 8th IEEE Conference on Network Protocols,* Japan, November 2000.

[66] UMTS Forum (www.umts-forum.org), October 2000.

[67] N. H. Vaidya, M. Mehta, C. Perkins, G. Montenegro, "Delayed Duplicate Acknowledgements: A TCP-unaware Approach to Improve Performance of TCP over Wireless" Technical Report 99-003, Computer Science Dept., Texas A&M University, February1999.

[68] B. Walke, *Mobile Radio Networks,* John Wiley & Sons, 1999.

[69] Wireless Application Protocol Forum Ltd., *Official Wireless Application Protocol: The Complete Standard with Searchable CD-ROM*, John Wiley & Sons, 1999.

[70] Wireless Application Protocol Forum (www.wapforum.org), October 2000.

[71] Wireless Data University (www.wirelessu.com), October 2000.

[72] Wireless week magazine, *Wireless Industry Terms,* October 2000. Available at http://www.wirelessweek.com/industry/terms.htm.

[73] G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley, 1995.

[74] M. Zorzi, R.R. Rao, "The effect of correlated errors on the performance of TCP", in *IEEE Communications Letters***,** vol. 1, pp. 127-129, Sep. 1997.

[75] M. Zorzi, R.R. Rao, "The energy efficiency of TCP", in *Proceedings of MoMUC '99*, San Diego, CA, 1999.