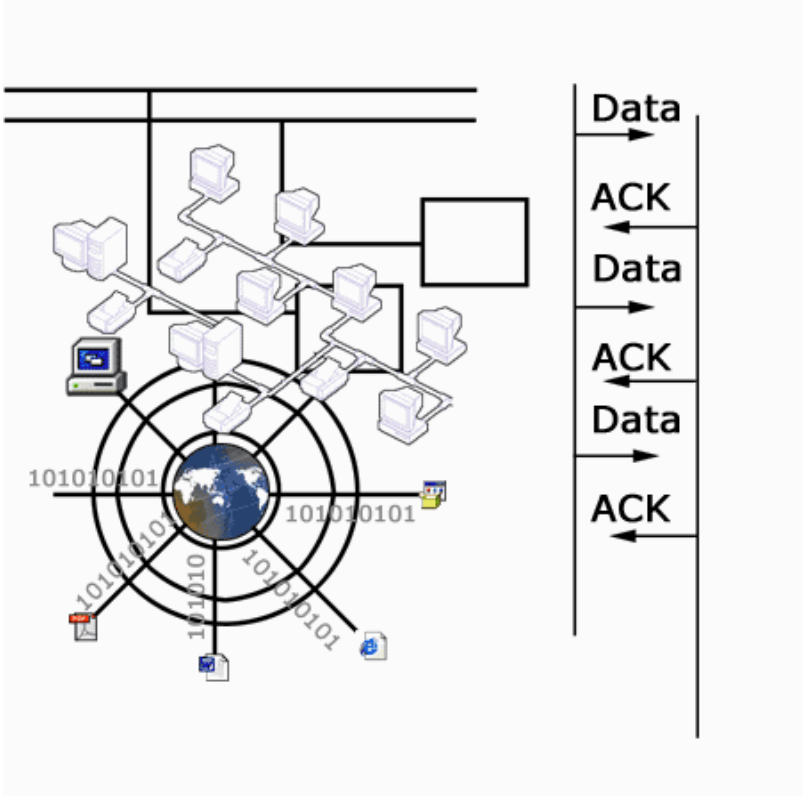
Can TCP be the transport protocol of the 21st century?

by Kostas Pentikousis



TCP is a transport layer network protocol that offers a reliable, connection-oriented, byte-stream service [19]. It is a full-duplex protocol, which means that each TCP connection supports a pair of byte-streams, one flowing in each direction. TCP supports *flow control*, which prevents the sender from overrunning the buffer capacity of the receiver. In addition, TCP implements *congestion control*, which prevents the sender from injecting too much traffic into the network. TCP is an *end-to-end* protocol. That is, TCP turns a host-to-host packet delivery service, provided by IP, into a process-to-process communication channel [19]. For a more detailed introduction to TCP, see [19] and [22].

Since 1981 when TCP was first introduced [20], the Internet has undergone major changes. From the early days when 1440 bits per second (bps) worth of bandwidth was astonishing, to the bandwidth liberation of the 1990s; from character-based network applications like e-mail and telnet, to the media-rich World Wide Web. And now, the new frontier: wireless handheld devices.

During these transitions, TCP evolved to meet the needs of these new technologies. The engineers who worked on improving TCP succeeded in designing a protocol that is robust, scalable, fault-tolerant, and well-performing. For this reason, TCP remains the predominant transport layer protocol of the Internet. Competitive technologies of the early 80s, such as Unix to Unix Copy Program (UUCP) and networks like BITNET [6], are obsolete now. In the 1990s, Asynchronous Transfer Mode (ATM) was introduced as the ultimate unifying network solution. ATM provided a number of features, such as Quality of Service (QoS), that were highly desirable and missing from TCP/IP protocol stack. Consequently, ATM received significant attention by the trade press, who heralded it as the "end of TCP." Many engineers rode this bandwagon, and devised a variety of ATM-related technologies, including: Wide Area ATM networks, "ATM to the desktop", and "native ATM" applications. However, for many different reasons, ATM did not manage to play a central role in the success story of the Internet. Today it is viewed as yet another network infrastructure on top of which a TCP/IP stack runs [19].

Although TCP has outlasted ATM, it faces new challenges with the rise of wireless and mobile computing. Can TCP remain the transport layer of choice for a wired-wireless world (i.e., an Internet that includes hosts that use wired and wireless connections)?

The Internet Protocol, which is the inter-networking protocol that TCP usually relies upon, underwent a major redesign in recent years. Its new design, IPv6, solves the problem of accommodating an ever-increasing number of hosts that connect to the Internet. The new standard allows for approximately $3x10^{38}$ IP-enabled devices. More information about IPv6 is provided at the official IPv6 web site at http://www.ipv6.org.

Mobile IP has been developed to provide "routing support...[to] IP nodes (hosts and routers) using either IPv4 or IPv6 to seamlessly 'roam' IP subnetworks and media types" [11]. This IP infrastructure is essential for providing to all connected mobile hosts the same functionality that desktops have been enjoying thus far. The natural choice of a transport layer protocol to work with Mobile IP is TCP. Yet TCP has not undergone any recent significant changes to meet the needs of mobile computing.

In the mid-90s a working group named TCP Next Generation (TCPng) was formed. Unfortunately, this group never published any RFCs or standards. Still, there have been considerable efforts to improve TCP, which is the focus of this article. For a comprehensive, but by no means exhaustive, list of proposed changes to TCP see http://www.aciri.org/floyd/tcp_small.html. In addition, [15] contains an overview of TCP improvement proposals.

Underlying communication infrastructure

Heterogeneity is an attribute that most characterizes the evolution of modern communications. TCP can be used in a variety of networks, including Ethernet, FDDI, and ATM. It can run over a variety of physical media, including wires, optical fibers, radio waves, or infrared signals. A TCP connection may, and usually does, work over all these different media/networks.

However, TCP was not designed with such heterogeneity in mind. Table 1 presents the Bit Error Rate (BER) as a function of media. The BER (or the probability that any given bit might get corrupted) is small over wired media. Consequentially, no Forward Error Correction (FEC) [12, 19] is used. FEC means that an error-correcting transmission encoding scheme is used instead of an error-detecting one. That is, redundant information is included in each packet that can be used by the receiver to determine *and* correct corrupted bits. FEC is usually employed in wireless transmissions because BERs are high.

Wired media	
Copper cable	10^{-6} to 10^{-7}
Fiber optic	10^{-12} to 10^{-14}

Wireless media	
Radio transmission without Error Correction	up to 10^{-1}
Radio transmission with Error Correction	10^{-6}

Table 1. Bit error rates for wired and wireless media [12, 19]

TCP has been used mainly for wired networks, so it has been highly tuned with certain assumptions in mind. For example, when a data segment is lost, in a wired network it is relatively safe to assume that this was most likely due to congestion (i.e. too many segments are contending for network resources) [2, 19]. But, if a wireless link or subnetwork is involved it could as well be because of bad reception at the location of the user.

It is not rare, and will certainly become increasingly more common in forthcoming years, that a given TCP connection will pass through networks with high latency/low bandwidth (e.g. a wireless WAN) and then through a low latency/high bandwidth (wired) network and then through a high bandwidth/high latency network (e.g. Long Fat Networks [15]). This complicates things even more since *TCP performance does not depend upon the transfer rate per se, but rather upon the product of the transfer rate and the round-trip delay* [10].

On the other hand, TCP survived the days of low bandwidth, high latency, and high error rates. So one might wonder why existing TCP versions are not able to cope efficiently with the evolving new environment. For example, TCP could also be used over a new medium where high BERs reign, bandwidth is limited, and latencies are obviously bigger due to the different propagation medium. For example, in wireless wide area networks the bandwidth can be as low as 19.2Kbps [21]. TCP worked well in environments under similar conditions before (e.g. dialup connections), so where is the problem?

(It's not) elementary, my dear Watson

Current TCP versions interpret a missing segment as congestion in the network, which implies that the sender should slow down its transmission rate because the network has reached its capacity. This interpretation is mostly true for wired-only networks. Unfortunately, if we also have wireless links in a communication path this could be wrong. Hence, TCP makes a wrong assumption, which usually leads to bad performance [4, 5, 7, 23]. Since we would like TCP to work well in wired-wireless environments, it is necessary that we find a strategy that decides in an intelligent manner what is happening, i.e. makes the right "guess" every time. In other words, TCP must work well under all underlying heterogeneous or homogeneous infrastructures.

Another important goal is energy efficiency. Current implementations of TCP do not have to worry about the energy efficiency of the protocol; all hosts have an unlimited power supply in TCP's world. This is not the case with laptop and handheld PCs, personal digital assistants (PDAs), and Wireless Application Protocol compliant phones, which rely on the limited power that a battery provides. A

number of parameters, like the total connection time and the TCP-induced byte overhead, the potential coupling with Data Link Layer (LL) protocols, and so on have to be studied. For example, some Data Link Layer protocols implement a reliable transmission scheme (e.g. ARQ [19]). At the same time, TCP also provides an end-to-end reliable transmission. This overlap, which attempts to ensure that the transmission is reliable, has been shown to lead to degraded overall performance [7]. The main idea here is to avoid doing things twice, both at the Data Link Layer and at the Transport Layer.

Application	Hosts would like access to services such as WWW, e-mail, FTP, messaging, etc.
Transport	TCP variants, I-TCP, UDP, experimental protocols
Network	All hosts should support IP. (IPv6 and Mobile IP if necessary)
Data Link	TCP unaware (TULIP), TCP-aware (Snoop)
Physical	FEC

Table 2. Different approaches in the architecture of a wired-wireless Internet.

Table 2 presents the different approaches researchers have followed in order to solve the aforementioned problems. Some suggest that the Data Link Layer (LL) should be responsible for providing a homogeneous network environment. Others, favor a Transport Layer solution. In the following paragraphs we summarize the different approaches along with their advantages and disadvantages.

Data Link Layer proposals

Proposals in this category have a very strong argument. The Data Link Layer (LL) has more control over the Physical Layer than any other layer. Since the problem lies at the Physical Layer (high link error rates in a wireless environment), a LL protocol can sense a problem faster. Therefore, we should solve the problem at the LL and provide a good communication channel - similar to a wired one - to the layers above, in particular to TCP. This enables us to keep current TCP implementations as they are. The way to do this is to ensure reliable delivery of packets at all times, for example using an Automatic Repeat reQuest (ARQ) scheme. However, studies have shown that this could lead to degraded performance [7]. This is because TCP also attempts to ensure reliable transmission, thus leading to a lot of duplicated effort. The authors in [5] suggest that if such a solution were accepted, it has to be tightly coupled with TCP. Nevertheless, the developers of TULIP (Transport Unaware Link Improvement Protocol) show that over half-duplex radio channels an "uncoupled" solution is indeed possible, and can yield better performance [17].

TCP-aware LL proposals

The most prominent proposal in this category is the *snoop* protocol [5]. Snoop lies in an intermediate

position between the LL proposals and the split TCP connection proposals. It utilizes the ability of an LL protocol to respond rapidly and to use available information to keep TCP "happy" with the existing network connection. For example, snoop will favor local LL retransmissions instead of TCP retransmissions. This is achievable because a snoop agent will notice duplicate acknowledgments (dupacks) traveling from the receiver to the sender and will suppress them, locally re-transmitting the (potentially) lost segment instead. Without snoop, the receipt of a certain number of dupacks would have caused a TCP segment retransmission [2].

Split TCP connections

Split TCP connection proposals, such as Indirect TCP (I-TCP) [3], were first proposed in the mid-90s. The idea behind I-TCP is based on the fact that a TCP connection can be established over two completely different classes of subnetworks namely, wired and wireless. Therefore, each TCP connection could be split into two connections at the point where the two subnetworks meet. For example, suppose we have a mobile user who is browsing a (conventional) web site using his laptop. The TCP connection will be split into two: one between the mobile host and the base station and one between the base station and the web server. The advantage is that we can utilize the best transport protocol for each type of network. This is very similar to using an HTTP proxy server. Splitting the TCP connection is essentially the technique used under the Wireless Application Protocol (WAP) architecture. The WAP Forum official web site at http://www.wapforum.com provides more information on the WAP protocol stack.

By splitting TCP connections we lose the end-to-end semantics of TCP. Consequently, we do not have a process-to-process communication channel. Intermediate nodes act as proxies, inspecting and modifying every single segment. In addition, the performance may also be degraded by splitting a particular connection several times. Handoffs are also not handled as efficiently, and crashes in the base station result in TCP connection termination [15].

TCP modification proposals

The seamless, highly modular, and highly independent nature of the Transport Layer makes it an ideal place to remedy problems introduced by mobility. Moreover, most current internets provide a *best-effort* service: the underlying network infrastructure can drop, reorder, and duplicate segments, introduce variations in delay (jitter), and limit messages to some finite size. At the same time, Application Layer protocols need guaranteed in-order message delivery, expect support for arbitrarily large messages, and require network services availability for multiple processes. Hence, the "end-to-end argument" [19] favors the usage of an efficient transport protocol. TCP is probably the best candidate. TCP makes maintaining compatibility with the rest of the Internet easier; a large amount of research has been invested in it; and years of experience with TCP could be helpful. Furthermore, with IPv6 ready for the brave new world, TCP is its natural transport protocol. After all, the Internet has been so successful due to the robustness that TCP has shown.

As noted earlier, TCP interprets a missing segment as congestion, and therefore slows down its transmission rate. TCP modification proposals attempt to make TCP more aggressive, sustaining the transmission rate for some period of time. Depending on the nature, distribution, and persistence of the drops, the three most common TCP versions, i.e. Tahoe, Reno [2] and New Reno [9], seem to outperform each other [8, 23]. None of these three common TCP version have been shown to be superior across all testing configurations. In addition, TCP Reno with snoop has been shown to outperform the standard TCP Reno. Is it worthwhile to search for better TCP versions?

The numerous TCP modification proposals imply that there is merit in the quest for a next generation version of TCP. They each attempt to do the following:

- Add smarts to TCP Congestion Control algorithms like Selective Acknowledgment TCP (SACK TCP) [13], Forward Acknowledge TCP (FACK TCP) [14], and TCP Probing [24]. In the latter, the intention is to try to determine whether the missing segment was due to relatively persistent error conditions (e.g. congestion, prolonged wireless burst errors) or purely transient ones (e.g. wireless random errors). In the former case, the sender should back off and slow down his transmission rate; in the latter he should continue at the same rate.

- Isolate the forward path (where data flows) from the reverse path (where ACKs flow), thereby providing better performance under many scenarios, including asymmetric links. One such example is TCP Santa Cruz [18], which also proposes the decoupling of the growth of the congestion window from the number of returned ACKs.

- Improve the performance at the beginning of a TCP connection by increasing the initial size of the congestion window, and changing the algorithm by which the congestion window grows during Slow Start [2, 15].

- Explicitly point out when congestion is building up, e.g. using Random Early Detection and Explicit Congestion Notification [19], and therefore signal the sender to slow down. Implicitly, this could mean that any other drop must be due to random losses.

Experimental Transport Layer protocols

It is quite clear that TCP introduces a lot of overhead, which is always undesirable in principle, but sometimes highly necessary in practice. A number of researchers have proposed experimental, highly tuned Transport Layer protocols more suited than TCP is for wired-wireless networks. For example, the Wireless Transmission Control Protocol (WTCP) [21] was designed for Wide Area Networks with limited and variable throughput, and high and variable latencies. New, highly-tuned transport protocols can also be combined with the split TCP connection proposals to deliver a much-improved performance. Moreover, new Transport Layer protocols can be designed with energy efficiency in mind, and therefore are better solutions for portable devices. The WAP Forum has followed this approach in the current

version of the standards they have presented. WAP-based phones and services are already a reality in many countries around the globe. But word is already out that the next generation of standards will incorporate TCP. So, why bother?

First of all, we would like all connected hosts, regardless of size, mobility, and computational power, to be equal. We would like mobile devices to interact with existing hosts and use common services like the Web and e-mail directly. In terms of reliability, using an experimental protocol on a wide scale cannot be as robust as using TCP. All the newly proposed protocols must show their robustness and ability to scale.

If a new Transport Layer protocol had to be used at the mobile nodes, then in order to permit them to interact with the rest of the Internet some kind of proxy service is required. However, device-specific characteristics, like minimal display size, limited memory, and input restrictions, dictate that conventional services must first be "adjusted" before they are provided to, for example, a PDA user. A proxy service can perform this "adjustment" at the Application Layer, and additionally, it could use another Transport Layer protocol to deliver the modified content to the end user. On the other hand, the use of a proxy limits the number of Internet services enjoyed by the mobile user to the ones that are supported by the proxy, and introduces a single point of failure in a network.

And the winner is...

It has not yet been determined if it is better to solve the aforementioned problems at the Transport Layer or instead at the layers below it. If the former is preferable, is it better to have TCP modified or should we instead proceed with the introduction of totally new transport protocols? In addition, it is also not clear which of the above proposals will be included in the next standard specification of TCP. However, as this is a very active research area in networks, it is reasonable to expect some sort of resolution soon. As a final word, given that IP has already evolved, and that most of the current proposals assume an IP datagram service, TCP seems to have an advantage over the other solutions.

References

1

Allman, M., On the Generation and Use of TCP Acknowledgments. In *ACM Computer Communications Review*, 28(5), October 1998.

2

Allman, Paxson, et al. *TCP Congestion Control*, RFC 2581, April 1999.

3

A. Bakre and B. R. Badrinath. "I-TCP: Indirect TCP for Mobile Host". Technical DCS-TR314, Rutgers University, October 1994.

4

---. "Handoff and system support for Indirect TCP/IP. In *Proceedings of the Second USENIX*

Symposium on Mobile and Location-Independent Computing, April 1995.

Balakrishnan, Seshan, et al. "Improving TCP/IP Performance Over Wireless Networks". In *Proceedings of ACM MobiCom*, November 1995.

CREN. *CREN History and Future*, http://www.cren.net/cren/cren-hist-fut.html, May 2000.

DeSimone, Chuah, et al. "Throughput Performance of Transport-Layer Protocols over Wireless LANs". In *Proceedings of Globecom '93*, December 1993.

K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP". *Computer Communication Review*, Volume 26 No. 3, pp. 36-46, July 1996

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

Jacobson, Braden, et al. *TCP Extensions for High Performance*, RFC 1323, May 1992.

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

William C.Y. Lee, *Mobile Communications Design Fundamentals, 2nd edition*, John Wiley and Sons, 1993.

Mathis, Mahdavi, et al. *TCP Selective Acknowledgment Options* RFC 2018, April 1996.

M. Mathis and J. Mahdavi. "Forward acknowledgment: Refining TCP congestion control." In *Proceedings of the ACM SIGCOMM*, August 1996.

Montenegro, Dawkins, et al. *Long Thin Networks*, RFC 2757, January 2000.

Nguyen, Giao, et al. "A Trace-Based Approach for Modeling Wireless Channel Behavior," In *Proceedings of the Winter Simulation Conference*, Coronado, CA, December 1996.

C. Parsa and J.J. Garcia-Luna-Aceves, "Improving TCP Performance over Wireless Networks at the Link Layer", *Mobile Networks and Applications*, 1999.

---, "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media", In *Proceedings of IEEE ICNP '99*, Toronto, October 1999.

L. L. Peterson and B. S. Davie. *Computer Networks, 2nd edition*, Morgan-Kaufmann, 2000.

Postel, J. B. *Transmission Control Protocol*. RFC 793, September 1981.

21

Sinha, Venkitaraman, et al. "WTCP: a reliable transport protocol for wireless wide-area networks", in *Proceedings of the fifth annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 1999, Seattle, WA USA.

22

Tanenbaum, A. *Computer Networks, 3rd edition*, Prentice Hall, 1996.

23

Tsaoussidis, Badr, et al. "Energy/Throughput Tradeoffs of TCP Error Control Strategies", In *Proceedings of IEEE Symposium on Computers and Communications*, IEEE Computer Society Press, France, July 2000.

24

V. Tsaoussidis, H. Badr, "Efficient Energy/Throughput Tradeoffs of Reliable Transport Protocols using Probing Mechanisms", in *Proceedings of PDPTA 2000*, CSREA Press, Las Vegas, June 2000.

Biography

Kostas Pentikousis (ACM S 2000, IEEE S 2000) is a PhD student in Computer Science at the State University of New York at Stony Brook. His research interests are in the areas of computer networks, including transport and application layer protocols, wireless communications, and network management. He received a B.Sc. (Honors) in Computer Science from Aristotle University, Greece (1996) and a M. Sc. in Computer Science from SUNY Stony Brook (2000). He can be reached at kostas@cs.sunysb.edu.
