TCP in a Wireless World

Geoff Huston • *Telstra* • *gih@telstra.net*

Mobile wireless is one of the more challenging environments for the Internet protocols, and for TCP in particular. One approach to supporting the wireless environment is the so-called "walled garden." Here the transport protocol used within the mobile wireless environment is not TCP, but is instead a transport protocol that has been specifically adapted to mobile wireless. In this model, Internet applications interact with an application gateway to reach the wireless world, and the application gateway uses a wireless transport protocol and potentially a modified version of the application data to interact with the wireless device. The most common implementation of this approach is to extend a Web client into the mobile wireless device, using some form of proxy server at the boundary of the wireless network and the Internet. This is the approach adopted by the Wireless Access Protocol (WAP) Forum.

An alternative is to allow mobile wireless devices to function as any other Internet-connected device. This approach requires some form of end-to-end direct IP continuity and an associated end-to-end TCP functionality, where the TCP path straddles both wired and wireless segments. Ensuring the efficient operation of TCP in this environment becomes integral to the development of the environment itself: The problem is no longer one of adjusting TCP to match the requirements of the wireless environment, but one of providing seamless interworking between the wired and wireless worlds, as shown in Figure 1.

Managing Bit-Level Corruption

TCP was designed for wire-based carriage and the protocol design makes numerous assumptions that are typical of such an environment.[1,2] For example, it assumes that packet loss is the result of network congestion, that round-trip times (RTT) have some level of stability, that bandwidth is constant, and that session durations will justify the initial TCP handshake overhead. The wireless environment challenges these assumptions and others underlying TCP design. Wireless has significant bit-error rates (BER), often bursting to very high rates. Wireless links that use forward error-correcting (FEC) codes often have high latency. If the link-level protocol uses a stop and resend mechanism that automatically retransmits corrupted data, this wireless segment latency will also have high variability. Wireless links may also use adaptive coding techniques that adjust to the prevailing signal-to-noise ratio, in which case the bandwidth of the link will vary. If the wireless device is a handheld mobile device, it may be memory constrained. And finally, wireless environments typically support short-duration sessions.

The major factor for mobile wireless is the BER, where frame loss of up to 1 percent is not uncommon, and errors occur in bursts, rather than evenly spaced in the packet stream. These error conditions force the TCP sender initially to attempt fast retransmit of the missing segments; when this does not correct the condition, the sender will experience an ACK time-out, causing the sender to collapse its sending window and recommence in TCP's slow-start mode from the point of packet loss. The heart of this problem is TCP's assumption that packet loss is a symptom of network congestion rather than packet corruption. It is possible to determine the effects of this loss rate on TCP performance,[3] and the outcome of such performance models is that TCP is very sensitive to packet loss levels, and sustainable performance rapidly drops when packet drops exceed 1 percent.

Layer 2 ARQ

Data-link-level solutions to the high BER are available to designers, and FEC codes and automatic retransmission systems (ARQ) can be used on the wireless link. FEC introduces a relatively constant

 1089-7801/01/$10.00

coding delay and a bandwidth overhead into the path, but cannot correct all forms of bit-error corruption. ARQ uses a "stop and resend" control mechanism similar to TCP itself; this can cause TCP to integrate extended latencies into its RTT estimate, making it assume a far higher latency on the path than is the case, or, more likely, it may trigger a retransmission at the same time the ARQ is already retransmitting the same data.

If Layer 2 ARQ is not the best possible answer to addressing packet loss in mobile wireless systems, then what can be done at the TCP level? There are some basic ways to alleviate the worst aspects of packet corruption on TCP performance. For example, there is a Fast Retransmit and Fast Recovery mechanism that allows a single packet loss to be repaired fairly quickly. Selective acknowledgments, or SACKs, allow a sender to repair multiple segment losses per window within a single RTT, and can be useful where large windows are operated over long delay paths.

However, these mechanisms are probably inadequate to allow TCP to function efficiently over all forms of wireless systems. Particularly in the case of mobile wireless systems, packet corruption is sufficiently common that it will probably have to be addressed directly for TCP to work efficiently.

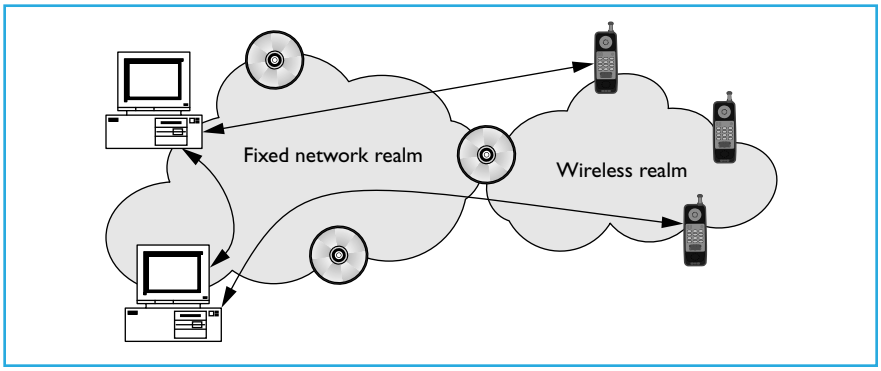

Figure 1. Linking the wired and wireless worlds.

Forward ACK with Rate Halving

One approach is to decouple TCP congestion control mechanisms from data recovery actions. The intent is to allow new data to be sent during recovery to sustain TCP ACK clocking. This approach is termed forward acknowledgments with rate halving, or FACK,[4] where one packet is sent for every two ACKs received while TCP is recovering from lost packets. This algorithm effectively reduces the sending rate by one-half within one RTT interval, but does not freeze the sender to wait the draining of one-half the congestion window's data from the network before proceeding to send further data. This allows the sender to continue to pass new data into the network during recovery, while preventing the sender from bursting a large volume of retransmission into the network.

FACK is particularly effective for long-delay networks, where the fast-recovery algorithm causes the sender to cease sending for up to one RTT interval, thereby losing the accuracy of the implicit ACK clock for the session. FACK lets the sender continue sending packets into the network during this period, in an effort to maintain an accurate view of the ACK clock. FACK also provides a capability to set the number of SACK blocks that specify a missing segment before resending the segment, giving the sender greater control over sensitivity to packet reordering. This approach requires the sender's TCP to use the FACK algorithm for recovery and, for optimal performance, the receiver's TCP to use SACK options.

Link-Level Signaling

Corrupted TCP segments are often detected at the data-link level and discarded by the link-level drivers. This discard cannot be used to reliably generate an error message to the packet sender, given that the packet's IP header may itself be corrupted; for the same reason, the discard signal cannot be reliably passed to the receiver. However, even though the information is unreliable, this signaling from the link level to the transport level is useful because, at the transport level, it lets the sender know that the packet loss was not due to network congestion and that there is no need to take corrective action in terms of TCP congestion behavior.

One approach to provide this signaling calls for the link-level device to forward a "corruption experienced" Internet control message protocol (ICMP) packet when discarding a corrupted packet.[4] The ICMP packet is sent forward to the receiver, who then has the task of converting this message and the associated lost-packet information into a signal to the sender that the duplicate ACKs are the result of corruption, not network congestion. This signal from the receiver to the sender can be embedded in a TCP header option. The sending TCP session will maintain a "corruption experienced" state for two RTT intervals, retransmitting the lost packets without halving the congestion window size.

If corruption occurred in the packet header, the sender's address may not be reliable. The ICMP approach addresses this problem by having the router keep a cache of recent packet destinations. Then when a failed IP header checksum indicates that the IP header information is unreliable, the router will forward the ICMP message to all destinations in the cache. The potential weakness in this approach is that network congestion may occur at the same time as packet corruption, in which case the sender will not react to the congestion and will continue to send into the congestion point for two more RTT intervals.

This approach is not without deployment concerns. The wireless routers and the receiver's link-level drivers must be modified to generate the ICMP corruption experienced messages. The receiver's IP stack must be modified to take

signals from the IP ICMP processor and from the link-level driver and then convert them to TCP corruption loss signals within the TCP header of the duplicate ACKs. Finally, the TCP processor at the sender must be modified to undertake corruption-experienced packet-loss recovery. Nevertheless, explicit corruption signaling is a very promising approach to the performance issues with TCP over wireless.

Managing Payloads

High BERs are not the only problem facing TCP over wireless systems. Mobile wireless systems are typically small handsets or personal digital assistants, and the application transactions are often modified to reduce the amount of data transferred, given the limited amount of data that can be displayed on the device.

In this case, the ratio between payload and IP and TCP headers becomes an issue, and header compression must be considered. Header compression techniques typically strip out those header fields that do not vary on a packet-by-packet basis, or that vary by amounts that can be derived from other parts of the header; the delta values of those fields that are varying are then transmitted.[5,6]

Although such compression schemes can be highly efficient in operation, they also require the receiver to have successfully received and decompressed a packet before decompressing the next packet in the TCP stream. In the face of high BERs, these schemes introduce additional latencies into the data transfer, and multiple packet drops are difficult to detect and signal via SACK in this case.

Managing Link Outages

A more subtle aspect of mobile wireless is temporary link outages. For example, a mobile user may enter an area of no signal coverage for a period of time and attempt to resume the data stream when signal is obtained again. In the same way that there is no accepted way for a link-level driver to inform TCP of packet loss due to corruption, there is no way it can inform TCP of a link-level outage. In the face of such outages, TCP will assume network-level congestion and, in the absence of duplicate ACKs, will trigger retransmission timers. TCP will then attempt to restart the session in slow-start mode, commencing with the first dropped packet. Each attempt to send the packet will cause TCP to extend its retransmission timer using an exponential backoff, so that successive probes are less and less frequent. Because the link level cannot inform the sender on the resumption of the link, TCP may wait some considerable time before responding to link restoration.

To enable the link level to inform TCP that the connection has been resumed, the link level could retain a packet from each TCP stream that attempted to use the link. When the link becomes operational, the link-level driver would immediately transmit these packets on the link. The receiver could then generate a response that would trigger the sender into transmission within an RTT interval. Only a single packet per active TCP stream is necessary to trigger this response, so that the link level does not need to hold an extensive buffer of undeliverable packets during a link outage. Of course, if the routing level repaired the link outage in the meantime, the delivery of an out-of-order TCP packet would normally be discarded by the sender.

Conclusion

Is TCP suitable for the mobile wireless environment? It seems possible with some changes in its operation, specifically relating to the signaling of link-level states into the TCP session and the use of advanced congestion control and corruption signaling within the TCP session. Although it is difficult to imagine changing every deployed TCP stack within the Internet to achieve this added functionality, there does exist a middle ground between the "walled garden" approach and open IP. In this middle ground, the wireless systems would have access to "middleware," such as Web proxies and mail agents. These proxies would use a set of TCP options when communicating with mobile wireless clients that would make the application operate as efficiently as possible, while still permitting the mobile device transparent access to the Internet for other transactions.

Acknowledgments

This column is based on part of a longer article titled "The Future of TCP," published in *The Internet Protocol Journal*, vol. 3, no. 3, Sept. 2000; the complete article is available online at http://www.cisco.com/warp/public/759/ipj_3-3/ipj_3-3_futureTCP.html.

References

1. J. Postel, "Transmission Control Protocol," IETF RFC 793, Sept. 1981; available online at http://www.ietf.org/rfc/rfc793.txt.
2. G. Huston, "TCP Performance," *Internet Protocol Journal*, vol. 3, no. 2, Cisco Systems, June 2000; available online at http://www.cisco.com/warp/public/759/ipj_3-2_tcp.html.
3. P. Karn et al., "Advice for Internet Subnet Designers," work in progress, July 2000.
4. M. Allman, ed., "Ongoing TCP Research Related to Satellites," RFC 2760, Feb. 2000; available online at http://www.ietf.org/rfc/rfc2760.
5. V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links," IETF RFC 1144, Feb. 1990; available online at http://www.ietf.org/rfc/rfc1144.txt.
6. S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links," IETF RFC 2508, Feb. 1999; available online at http://www.ietf.org/rfc/rfc2508.txt.

Geoff Huston is Chief Scientist in the Internet area for Telstra. He is also a member of the Internet Architecture Board and secretary of the Internet Society Board of Trustees. He is author of *The ISP Survival Guide* and *Internet Performance Survival Guide: QoS Strategies for Multiservice Networks*, and coauthor with Paul Ferguson of *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, all published by John Wiley and Sons.

Readers may contact him via email at gih@telstra.net.