## 9.1 Traditional TCP

This section highlights several mechanisms of the transmission control protocol (TCP) (Postel, 1981) that influence the efficiency of TCP in a mobile environment. A very detailed presentation of TCP is given in Stevens (1994).

### 9.1.1 Congestion control

A transport layer protocol such as TCP has been designed for fixed networks with fixed end-systems. Data transmission takes place using network adapters, fiber optics, copper wires, special hardware for routers etc. This hardware typically works without introducing transmission errors. If the software is mature enough, it will not drop packets or flip bits, so if a packet on its way from a sender to a receiver is lost in a fixed network, it is not because of hardware or software errors. The probable reason for a packet loss in a fixed network is a temporary overload some point in the transmission path, i.e., a state of congestion at a node.

Congestion may appear from time to time even in carefully designed networks. The packet buffers of a router are filled and the router cannot forward the packets fast enough because the sum of the input rates of packets destined for one output link is higher than the capacity of the output link. The only thing a router can do in this situation is to drop packets. A dropped packet is lost for the transmission, and the receiver notices a gap in the packet stream. Now the receiver does not directly tell the sender which packet is missing, but continues to acknowledge all in-sequence packets up to the missing one.

The sender notices the missing acknowledgement for the lost packet and assumes a packet loss due to congestion. Retransmitting the missing packet and continuing at full sending rate would now be unwise, as this might only increase the congestion. Although it is not guaranteed that all packets of the TCP connection take the same way through the network, this assumption holds for most of the packets. To mitigate congestion, TCP slows down the transmission rate dramatically. All other TCP connections experiencing the same congestion do exactly the same so the congestion is soon resolved. This cooperation of TCP connections in the internet is one of the main reasons for its survival as it is today. Using UDP is not a solution, because the throughput is higher compared to a TCP connection just at the beginning. As soon as everyone uses UDP, this advantage disappears. After that, congestion is standard and data transmission quality is unpredictable. Even under heavy load, TCP guarantees at least sharing of the bandwidth.

### 9.1.2 Slow start

TCP's reaction to a missing acknowledgement is quite drastic, but it is necessary to get rid of congestion quickly. The behavior TCP shows after the detection of congestion is called **slow start** (Kurose, 2003).

The sender always calculates a **congestion window** for a receiver. The start size of the congestion window is one segment (TCP packet). The sender sends one packet and waits for acknowledgement. If this acknowledgement arrives, the sender increases the congestion window by one, now sending two packets (congestion window = 2). After arrival of the two corresponding acknowledgements, the sender again adds 2 to the congestion window, one for each of the acknowledgements. Now the congestion window equals 4. This scheme doubles the congestion window every time the acknowledgements come back, which takes one round trip time (RTT). This is called the exponential growth of the congestion window in the slow start mechanism.

It is too dangerous to double the congestion window each time because the steps might become too large. The exponential growth stops at the **congestion threshold**. As soon as the congestion window reaches the congestion threshold, further increase of the transmission rate is only linear by adding 1 to the congestion window each time the acknowledgements come back.

Linear increase continues until a time-out at the sender occurs due to a missing acknowledgement, or until the sender detects a gap in transmitted data because of continuous acknowledgements for the same packet. In either case the sender sets the congestion threshold to half of the current congestion window. The congestion window itself is set to one segment and the sender starts sending a single segment. The exponential growth (as described above) starts once more up to the new congestion threshold, then the window grows in linear fashion.

### 9.1.3 Fast retransmit/fast recovery

Two things lead to a reduction of the congestion threshold. One is a sender receiving continuous acknowledgements for the same packet. This informs the sender of two things. One is that the receiver got all packets up to the acknowledged packet in sequence. In TCP, a receiver sends acknowledgements only if it receives any packets from the sender. Receiving acknowledgements from a receiver also shows that the receiver continuously receives something from the sender. The gap in the packet stream is not due to severe congestion, but a simple packet loss due to a transmission error. The sender can now retransmit the missing packet(s) before the timer expires. This behavior is called **fast retransmit** (Kurose, 2003).

The receipt of acknowledgements shows that there is no congestion to justify a slow start. The sender can continue with the current congestion window. The sender performs a **fast recovery** from the packet loss. This mechanism can improve the efficiency of TCP dramatically.

The other reason for activating slow start is a time-out due to a missing acknowledgement. TCP using fast retransmit/fast recovery interprets this congestion in the network and activates the slow start mechanism.

### 9.1.4 Implications on mobility

While slow start is one of the most useful mechanisms in fixed networks, it drastically decreases the efficiency of TCP if used together with mobile receivers or senders. The reason for this is the use of slow start under the wrong assumptions. From a missing acknowledgement, TCP concludes a congestion situation. While this may also happen in networks with mobile and wireless end-systems, it is not the main reason for packet loss.

Error rates on wireless links are orders of magnitude higher compared to fixed fiber or copper links. Packet loss is much more common and cannot always be compensated for by layer 2 retransmissions (ARQ) or error correction (FEC). Trying to retransmit on layer 2 could, for example, trigger TCP retransmission if it takes too long. Layer 2 now faces the problem of transmitting the same packet twice over a bad link. Detecting these duplicates on layer 2 is not an option, because more and more connections use end-to-end encryption, making it impossible to look at the packet.

Mobility itself can cause packet loss. There are many situations where a soft handover from one access point to another is not possible for a mobile end-system. For example, when using mobile IP, there could still be some packets in transit to the old foreign agent while the mobile node moves to the new foreign agent. The old foreign agent may not be able to forward those packets to the new foreign agent or even buffer the packets if disconnection of the mobile node takes too long. This packet loss has nothing to do with wireless access but is caused by the problems of rerouting traffic.

The TCP mechanism detecting missing acknowledgements via time-outs and concluding packet loss due to congestion cannot distinguish between the different causes. This is a fundamental design problem in TCP: An error control mechanism (missing acknowledgement due to a transmission error) is misused for congestion control (missing acknowledgement due to network overload). In both cases packets are lost (either due to invalid checksums or to dropping in routers). However, the reasons are completely different. TCP cannot distinguish between these two different reasons. Explicit congestion notification (ECN) mechanisms are currently discussed and some recommendations have been already given (RFC 3168, Ramakrishnan, 2001). However, RFC 3155 (Dawkins, 2001b) states that ECN cannot be used as surrogate for explicit transmission error notification. Standard TCP reacts with slow start if acknowledgements are missing, which does not help in the case of transmission errors over wireless links and which does not really help during handover. This behavior results in a severe performance degradation of an unchanged TCP if used together with wireless links or mobile nodes.

However, one cannot change TCP completely just to support mobile users or wireless links. The same arguments that were given to keep IP unchanged also apply to TCP. The installed base of computers using TCP is too large to be changed and, more important, mechanisms such as slow start keep the internet

operable. Every enhancement to TCP, therefore, has to remain compatible with the standard TCP and must not jeopardize the cautious behavior of TCP in case of congestion. The following sections present some classical solutions before discussing current TCP tuning recommendations.

## 9.2 Classical TCP improvements

Together with the introduction of WLANs in the mid-nineties several research projects were started with the goal to increase TCP's performance in wireless and mobile environments.

### 9.2.1 Indirect TCP

Two competing insights led to the development of indirect TCP (I-TCP) (Bakre, 1995). One is that TCP performs poorly together with wireless links; the other is that TCP within the fixed network cannot be changed. I-TCP segments a TCP connection into a fixed part and a wireless part. Figure 9.1 shows an example with a mobile host connected via a wireless link and an access point to the 'wired' internet where the correspondent host resides. The correspondent node could also use wireless access. The following would then also be applied to the access link of the correspondent host.

Standard TCP is used between the fixed computer and the access point. No computer in the internet recognizes any changes to TCP. Instead of the mobile host, the access point now terminates the standard TCP connection, acting as a proxy. This means that the access point is now seen as the mobile host for the fixed host and as the fixed host for the mobile host. Between the access point and the mobile host, a special TCP, adapted to wireless links, is used. However, changing TCP for the wireless link is not a requirement. Even an unchanged TCP can benefit from the much shorter round trip time, starting retransmission much faster. A good place for segmenting the connection between mobile host and correspondent host is at the foreign agent of mobile IP (see chapter 8). The foreign agent controls the mobility of the mobile host anyway and can also hand over the connection to the next foreign agent when the mobile host

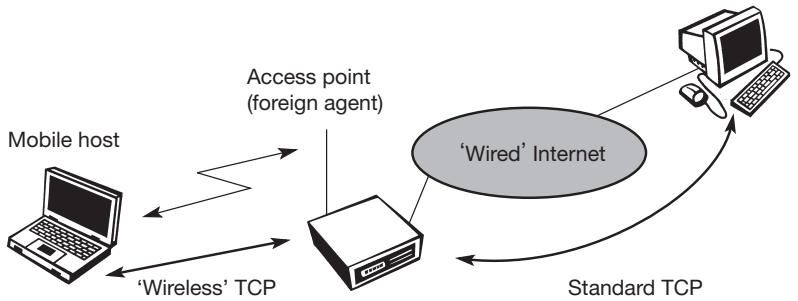


**Figure 9.1**
Indirect TCP segments a TCP connection into two parts

However, the simplicity of the scheme also results in some **disadvantages:**

- Snooping TCP does not isolate the behavior of the wireless link as well as I-TCP. Assume, for example, that it takes some time until the foreign agent can successfully retransmit a packet from its buffer due to problems on the wireless link (congestion, interference). Although the time-out in the foreign agent may be much shorter than the one of the correspondent host, after a while the time-out in the correspondent host triggers a retransmission. The problems on the wireless link are now also visible for the correspondent host and not fully isolated. The quality of the isolation, which snooping TCP offers, strongly depends on the quality of the wireless link, time-out values, and further traffic characteristics. It is problematic that the wireless link exhibits very high delays compared to the wired link due to error correction on layer 2 (factor 10 and more higher). This is similar to I-TCP. If this is the case, the timers in the foreign agent and the correspondent host are almost equal and the approach is almost ineffective.
- Using negative acknowledgements between the foreign agent and the mobile host assumes additional mechanisms on the mobile host. This approach is no longer transparent for arbitrary mobile hosts.
- All efforts for snooping and buffering data may be useless if certain encryption schemes are applied end-to-end between the correspondent host and mobile host. Using IP encapsulation security payload (RFC 2406, (Kent, 1998b)) the TCP protocol header will be encrypted – snooping on the sequence numbers will no longer work. Retransmitting data from the foreign agent may not work because many security schemes prevent replay attacks – retransmitting data from the foreign agent may be misinterpreted as replay. Encrypting end-to-end is the way many applications work so it is not clear how this scheme could be used in the future. If encryption is used above the transport layer (e.g., SSL/TLS) snooping TCP can be used.

### 9.2.3 Mobile TCP

Dropping packets due to a handover or higher bit error rates is not the only phenomenon of wireless links and mobility – the occurrence of lengthy and/or frequent disconnections is another problem. Quite often mobile users cannot connect at all. One example is islands of wireless LANs inside buildings but no coverage of the whole campus. What happens to standard TCP in the case of disconnection?

A TCP sender tries to retransmit data controlled by a retransmission timer that doubles with each unsuccessful retransmission attempt, up to a maximum of one minute (the initial value depends on the round trip time). This means that the sender tries to retransmit an unacknowledged packet every minute and will give up after 12 retransmissions. What happens if connectivity is back ear-

lier than this? No data is successfully transmitted for a period of one minute! The retransmission time-out is still valid and the sender has to wait. The sender also goes into slow-start because it assumes congestion.

What happens in the case of I-TCP if the mobile is disconnected? The proxy has to buffer more and more data, so the longer the period of disconnection, the more buffer is needed. If a handover follows the disconnection, which is typical, even more state has to be transferred to the new proxy. The snooping approach also suffers from being disconnected. The mobile will not be able to send ACKs so, snooping cannot help in this situation.

The **M-TCP (mobile TCP)**[1] approach has the same goals as I-TCP and snooping TCP: to prevent the sender window from shrinking if bit errors or disconnection but not congestion cause current problems. M-TCP wants to improve overall throughput, to lower the delay, to maintain end-to-end semantics of TCP, and to provide a more efficient handover. Additionally, M-TCP is especially adapted to the problems arising from lengthy or frequent disconnections (Brown, 1997).

M-TCP splits the TCP connection into two parts as I-TCP does. An unmodified TCP is used on the standard host-**supervisory host (SH)** connection, while an optimized TCP is used on the SH-MH connection. The supervisory host is responsible for exchanging data between both parts similar to the proxy in I-TCP (see Figure 9.1). The M-TCP approach assumes a relatively low bit error rate on the wireless link. Therefore, it does not perform caching/retransmission of data via the SH. If a packet is lost on the wireless link, it has to be retransmitted by the original sender. This maintains the TCP end-to-end semantics.

The SH monitors all packets sent to the MH and ACKs returned from the MH. If the SH does not receive an ACK for some time, it assumes that the MH is disconnected. It then chokes the sender by setting the sender's window size to 0. Setting the window size to 0 forces the sender to go into **persistent mode**, i.e., the state of the sender will not change no matter how long the receiver is disconnected. This means that the sender will not try to retransmit data. As soon as the SH (either the old SH or a new SH) detects connectivity again, it reopens the window of the sender to the old value. The sender can continue sending at full speed. This mechanism does not require changes to the sender's TCP.

The wireless side uses an adapted TCP that can recover from packet loss much faster. This modified TCP does not use slow start, thus, M-TCP needs a **bandwidth manager** to implement fair sharing over the wireless link.

The **advantages** of M-TCP are the following:

- It maintains the TCP end-to-end semantics. The SH does not send any ACK itself but forwards the ACKs from the MH.
- If the MH is disconnected, it avoids useless retransmissions, slow starts or breaking connections by simply shrinking the sender's window to 0.

1 The reader should be aware that mobile TCP does not have the same status as mobile IP, which is an internet RFC.

- Since it does not buffer data in the SH as I-TCP does, it is not necessary to forward buffers to a new SH. Lost packets will be automatically retransmitted to the new SH.

The lack of buffers and changing TCP on the wireless part also has some **disadvantages:**

- As the SH does not act as proxy as in I-TCP, packet loss on the wireless link due to bit errors is propagated to the sender. M-TCP assumes low bit error rates, which is not always a valid assumption.
- A modified TCP on the wireless link not only requires modifications to the MH protocol software but also new network elements like the bandwidth manager.

### 9.2.4 Fast retransmit/fast recovery

As described in section 9.1.4, moving to a new foreign agent can cause packet loss or time out at mobile hosts or corresponding hosts. TCP concludes congestion and goes into slow start, although there is no congestion. Section 9.1.3 showed the mechanisms of fast recovery/fast retransmit a host can use after receiving duplicate acknowledgements, thus concluding a packet loss without congestion.

The idea presented by Caceres (1995) is to artificially force the fast retransmit behavior on the mobile host and correspondent host side. As soon as the mobile host registers at a new foreign agent using mobile IP, it starts sending duplicated acknowledgements to correspondent hosts. The proposal is to send three duplicates. This forces the corresponding host to go into fast retransmit mode and not to start slow start, i.e., the correspondent host continues to send with the same rate it did before the mobile host moved to another foreign agent.

As the mobile host may also go into slow start after moving to a new foreign agent, this approach additionally puts the mobile host into fast retransmit. The mobile host retransmits all unacknowledged packets using the current congestion window size without going into slow start.

The **advantage** of this approach is its simplicity. Only minor changes in the mobile host's software already result in a performance increase. No foreign agent or correspondent host has to be changed.

The main **disadvantage** of this scheme is the insufficient isolation of packet losses. Forcing fast retransmission increases the efficiency, but retransmitted packets still have to cross the whole network between correspondent host and mobile host. If the handover from one foreign agent to another takes a longer time, the correspondent host will have already started retransmission. The approach focuses on loss due to handover: packet loss due to problems on the wireless link is not considered. This approach requires more cooperation between the mobile IP and TCP layer making it harder to change one without influencing the other.