Mobile transport layer

Most applications rely on a transport layer, such as TCP (transmission control protocol) or UDP (user datagram protocol) in the case of the internet. Two functions of the transport layer in the internet are checksumming over user data and multiplexing/ demultiplexing of data from/to applications. While the network layer only addresses a host, ports in UDP or TCP allow dedicated applications to be addressed. While UDP is connectionless and does not give certain guarantees about reliable data delivery, TCP is much more complex and, needs special mechanisms to be useful in mobile environments. Mobility support in IP (such as mobile IP) is already enough for UDP to work.

TCP has built-in mechanisms to behave in a ‘network friendly’ manner. If, for example, TCP encounters packet loss, it assumes network internal congestion and slows down the transmission rate.

UDP does not behave in a network friendly manner, i.e., does not pull back in case of congestion and continues to send packets into an already congested network. {an overview of mechanisms within TCP that play an important role when using TCP for mobility. Several classical solutions are presented in sections 9.2.1 to 9.2.7;}

## Classical TCP improvements

### 9.2.1 Indirect TCP

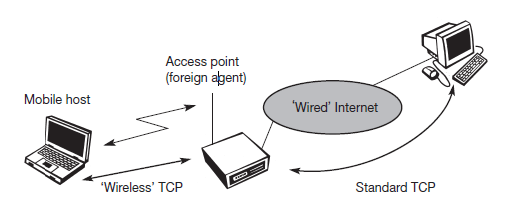


Figure 9.1: Indirect TCP segments a TCP connection into two parts

Two competing insights led to the development of indirect TCP (I-TCP):

* 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.

## There are several advantages with I-TCP:

● I-TCP does not require any changes in the TCP protocol as used by the hosts in the fixed network or other hosts in a wireless network that do not use this optimization. All current optimizations for TCP still work between the foreign agent and the correspondent host.

● Due to the strict partitioning into two connections, transmission errors on the wireless link, i.e., **lost packets, cannot propagate into the fixed network**. Without partitioning, retransmission of lost packets would take place between mobile host and correspondent host across the whole network. Now only packets in sequence, without gaps leave the foreign agent.

● It is always dangerous to introduce new mechanisms into a huge network such as the internet without knowing exactly how they will behave. However, new mechanisms are needed to improve TCP performance (e.g., disabling slow start under certain circumstances), but with I-TCP only between the mobile host and the foreign agent. Different solutions can be tested or used at the same time without jeopardizing the stability of the internet. Furthermore, optimizing of these new mechanisms is quite simple because they only cover one single hop.

● The authors assume that the short delay between the mobile host and foreign agent could be determined and was independent of other traffic streams. An optimized TCP could use precise time-outs to guarantee retransmission as fast as possible. Even standard TCP could benefit from the short round trip time, so recovering faster from packet loss. Delay is much higher in a typical wide area wireless network than in wired networks due to FEC and MAC. GSM has a delay of up to 100 ms circuit switched, 200 ms and more packet switched (depending on packet size and current traffic). This is even higher than the delay on transatlantic links.

● Partitioning into two connections also allows the use of a different transport layer protocol between the foreign agent and the mobile host or the use of compressed headers etc. The foreign agent can now act as a gateway to translate between the different protocols.

## I-TCP also comes with some disadvantages:

● The loss of the end-to-end semantics of TCP might cause problems if the foreign agent partitioning the TCP connection crashes. If a sender receives an acknowledgement, it assumes that the receiver got the packet. Receiving an acknowledgement now only means (for the mobile host and a correspondent host) that the foreign agent received the packet. The correspondent node does not know anything about the partitioning, so a crashing access node may also crash applications running on the correspondent node assuming reliable end-to-end delivery.

● In practical use, increased handover latency may be much more problematic. All packets sent by the correspondent host are buffered by the foreign agent besides forwarding them to the mobile host (if the TCP connection is split at the foreign agent). The foreign agent removes a packet from the buffer as soon as the appropriate acknowledgement arrives. If the mobile host now performs a handover to another foreign agent, it takes a while before the old foreign agent can forward the buffered data to the new foreign agent. During this time more packets may arrive. All these packets have to be forwarded to the new foreign agent first, before it can start forwarding the new packets redirected to it. ● The foreign agent must be a trusted entity because the TCP connections end at this point. If users apply end-to-end encryption, e.g., according to RFC 2401 (Kent, 1998a), the foreign agent has to be integrated into all security mechanisms.

### 9.2.2 Snooping TCP

One of the drawbacks of I-TCP is the segmentation of the single TCP connection into two TCP connections. This loses the original end-to-end TCP semantic. The following TCP enhancement works completely transparently and leaves the TCP end-to-end connection intact.



Figure 9.3:Snooping TCP as a transparent TCP extension

In this approach, the foreign agent buffers all packets with destination mobile host and additionally ‘snoops’ the packet flow in both directions to recognize acknowledgements (Balakrishnan, 1995), (Brewer, 1998). The reason for buffering packets toward the mobile node is to enable the foreign agent to perform a local retransmission in case of packet loss on the wireless link. The foreign agent buffers every packet until it receives an acknowledgement from the mobile host. If the foreign agent does not receive an acknowledgement from the mobile host within a certain amount of time, either the packet or the acknowledgement has been lost. Alternatively, the foreign agent could receive a duplicate ACK which also shows the loss of a packet. Now the foreign agent retransmits the packet directly from the buffer, performing a much faster retransmission compared to the correspondent host. The time out for acknowledgements can be much shorter, because it reflects only the delay of one hop plus processing time.

To remain transparent, the foreign agent must not acknowledge data to the correspondent host. This would make the correspondent host believe that the mobile host had received the data and would violate the end-to-end semantic in case of a foreign agent failure. However, the foreign agent can filter the duplicate acknowledgements to avoid unnecessary retransmissions of data from the correspondent host. If the foreign agent now crashes, the time-out of the correspondent host still works and triggers a retransmission. The foreign agent may discard duplicates of packets already retransmitted locally and acknowledged by the mobile host. This avoids unnecessary traffic on the wireless link. Data transfer from the mobile host with destination correspondent host works as follows. The foreign agent snoops into the packet stream to detect gaps in the sequence numbers of TCP. As soon as the foreign agent detects a missing packet, it returns a negative acknowledgement (NACK) to the mobile host. The mobile host can now retransmit the missing packet immediately. Reordering of packets is done automatically at the correspondent host by TCP. Extending the functions of a foreign agent with a ‘snooping’ TCP has several advantages:

● The end-to-end TCP semantic is preserved. No matter at what time the foreign agent crashes (if this is the location of the buffering and snooping mechanisms), neither the correspondent host nor the mobile host have an inconsistent view of the TCP connection as is possible with I-TCP. The approach automatically falls back to standard TCP if the enhancements stop working.

● The correspondent host does not need to be changed; most of the enhancements are in the foreign agent. Supporting only the packet stream from the correspondent host to the mobile host does not even require changes in the mobile host.

● It does not need a handover of state as soon as the mobile host moves to another foreign agent. Assume there might still be data in the buffer not transferred to the next foreign agent. All that happens is a time-out at the correspondent host and retransmission of the packets, possibly already to the new care-of address.

● It does not matter if the next foreign agent uses the enhancement or not. If not, the approach automatically falls back to the standard solution. This is one of the problems of I-TCP, since the old foreign agent may have already signaled the correct receipt of data via acknowledgements to the correspondent host and now has to transfer these packets to the mobile host via the new foreign agent.

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 ITCP. 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 ITCP. 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

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 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 ITCP (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 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.

● 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.