49202 Communication Protocols

Transport Layer Example Problems

Daniel R. Franklin

Faculty of Engineering & IT

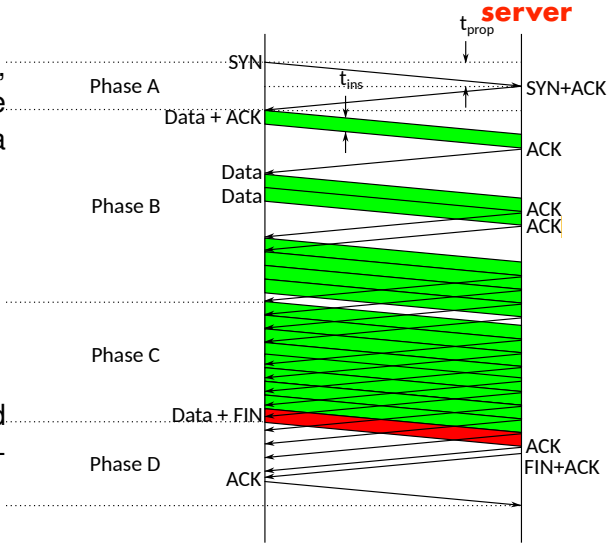
March 28, 2023

Example: TCP timing

With regards to the following diagram, calculate the durations of each phase of transmission for this typical TCP data transfer:

- t_{ins} = insertion time of a data segment (blue)
- t_{prop} = propagation time of a data segment from one end of the link to the other end.

Ignore insertion times of SYN, ACK and empty FIN segments, as well as processing times at each end.



Solution

- t_{prop} is the time taken for the first bit of a packet to travel from the sender to the receiver; we assume it is symmetric (receiver to sender = sender to receiver)
- Since insertion time (time taken to transmit) for the 3-way handshake segments is negligible, we see that **Phase A** takes $2t_{prop}$ to complete

Solution

- Phase B, the slow-start phase, starts with one data segment insertion (transmission), which takes t_{ins} units of time. This is then sent.
- The **last** bit of this transmission takes a further t_{prop} units of time to reach the receiver, at which point an ACK is sent back, arriving a further t_{prop} later. `CongestionWindow` is now increased to two segments.
- So, we transmit two more segments, before stopping since the `CongestionWindow` is exhausted.

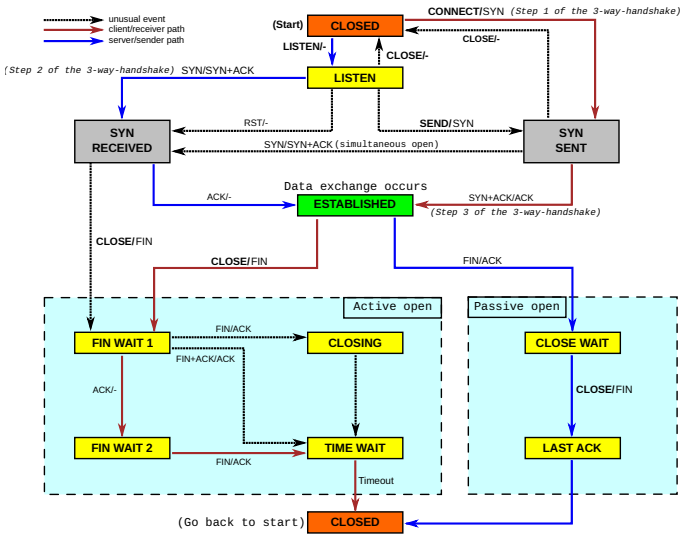
Solution

- We can send another frame when an ACK arrive for the first of those two segments, which occurs $2t_{prop}$ after the first segment arrives at the receiver (`CongestionWindow` increases by one for each received ACK). When the second ACK arrive, our `CongestionWindow` is now increased again by one. We may send a total of four segments this time.
- Tranmsission may resume when the ACK for the first of those four segments arrives (another $2t_{prop}$ later). But *this* time, with `CongestionWindow` reaching 8 segments, we are still sending when ACKs start returning - so we can transmit *continuously*. This marks the end of the slow start phase.
- So **Phase B** takes a total of $N_{RTT}(t_{ins} + 2t_{prop})$, where N_{RTT} is the number of round trip times needed to reach steady state

Solution

- **Phase C** is the phase where we are still transmitting at the time that ACKs start to return, so we can transmit continuously until all of our data segments are sent. Including the final red segment, this takes $9t\_ins$ (in this example).
- Finally, teardown is initiated when the last data segment, which has the FIN flag set, is transmitted. This is acknowledged $2t_{prop}$ units of time later; we the send the final ACK which arrives at the receiver after one more t_{prop}; so **Phase D** lasts $3t_{prop}$ units of time.

Example 2



- The normal state transition path taken by a *client* when the *server* terminates the session is FIN WAIT 1 followed by CLOSING (true/false?)
- The normal state transition path taken by a *server* when the *client* terminates the session is CLOSE WAIT followed by LAST ACK (true/false?)

Solution

- A client terminates the connection by sending a ClOSE/FIN segment and entering the FIN WAIT 1 state. When it receives the ACK from the server, it enters the FIN WAIT 2 state, where it waits for the server to send its FIN segment. When it receives this, it sends an ACK, before entering TIME WAIT state and finally CLOSED. So, the answer here is FALSE.
- The client would jump to the CLOSING state if the FIN is piggybacked on the ACK (one segment from the server has both flags set). This is not the normal process (but it could happen)
- The client would jump directly to TIME WAIT if the server has also initiated a termination at the same time (or overlapping in time; i.e. the FIN is received before the ACK to the client-initiated teardown). Having seen a FIN, we have to simply ACK and move to TIME WAIT followed by CLOSED
- If the server terminates the connection, things are a bit simpler - the packet exchange is similar (FIN-ACK/FIN-ACK) to the above, but there is only one possible path, via CLOSE WAIT and LAST ACK states.

Example 3

- A host with a large (unlimited) amount of data ready to send to the other end of a TCP connection currently has a `CongestionWindow` of 8192 bytes. The last acknowledgement from the other end of the connection included an `AdvertisedWindow` of 4096 bytes. If the MSS of this network is 1460 bytes, and no further acknowledgements arrive what is the size of the next three TCP segments that are transmitted?

Solution

- We can send no more than the smaller of `CongestionWindow` and `AdvertisedWindow`, which is 4096 bytes (until more ACKs arrive...)
- 4096 / 1460 = 2 remainder 1176 bytes
- Therefore, we send two segments of 1460 bytes and one of 1176 bytes.