The Network Stack (2)

Lecture 6, Part 1: TCP

Prof. Robert N. M. Watson

2026-2027

The Network Stack (2)

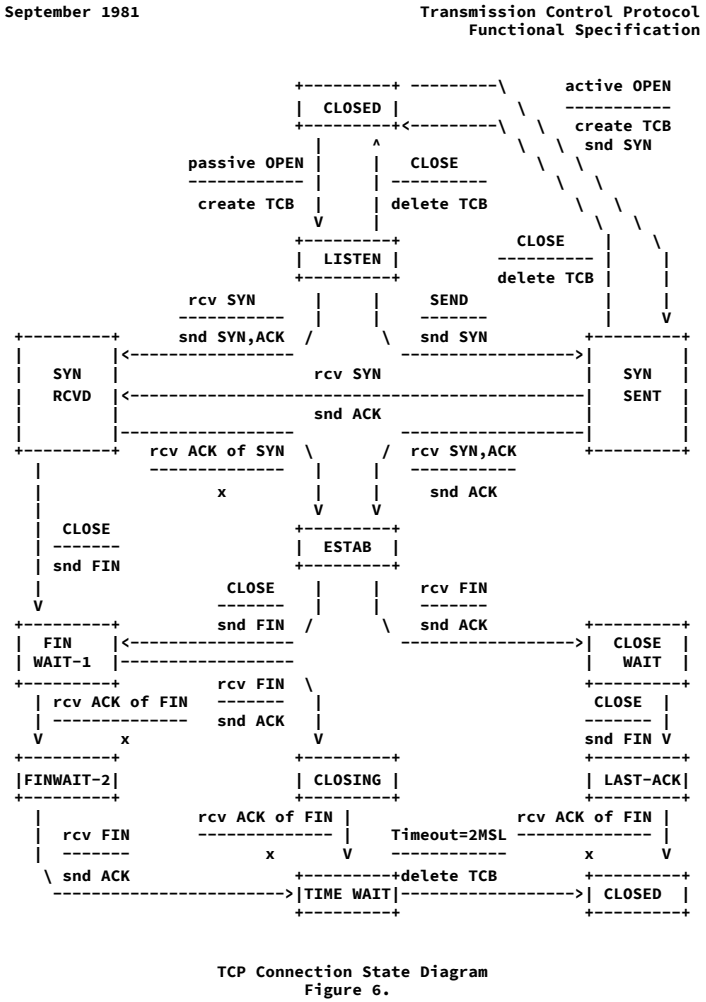
- The Transmission Control Protocol (TCP)
  - The TCP state machine
  - TCP congestion control
  - TCP implementations and performance
  - The evolving TCP stack
  - Lab 3 on TCP
- Wrapping up the Advanced Operating Systems lecture series

Lecture 6, Part 1

Lecture 6, Part 2

Lecture 6, Part 3

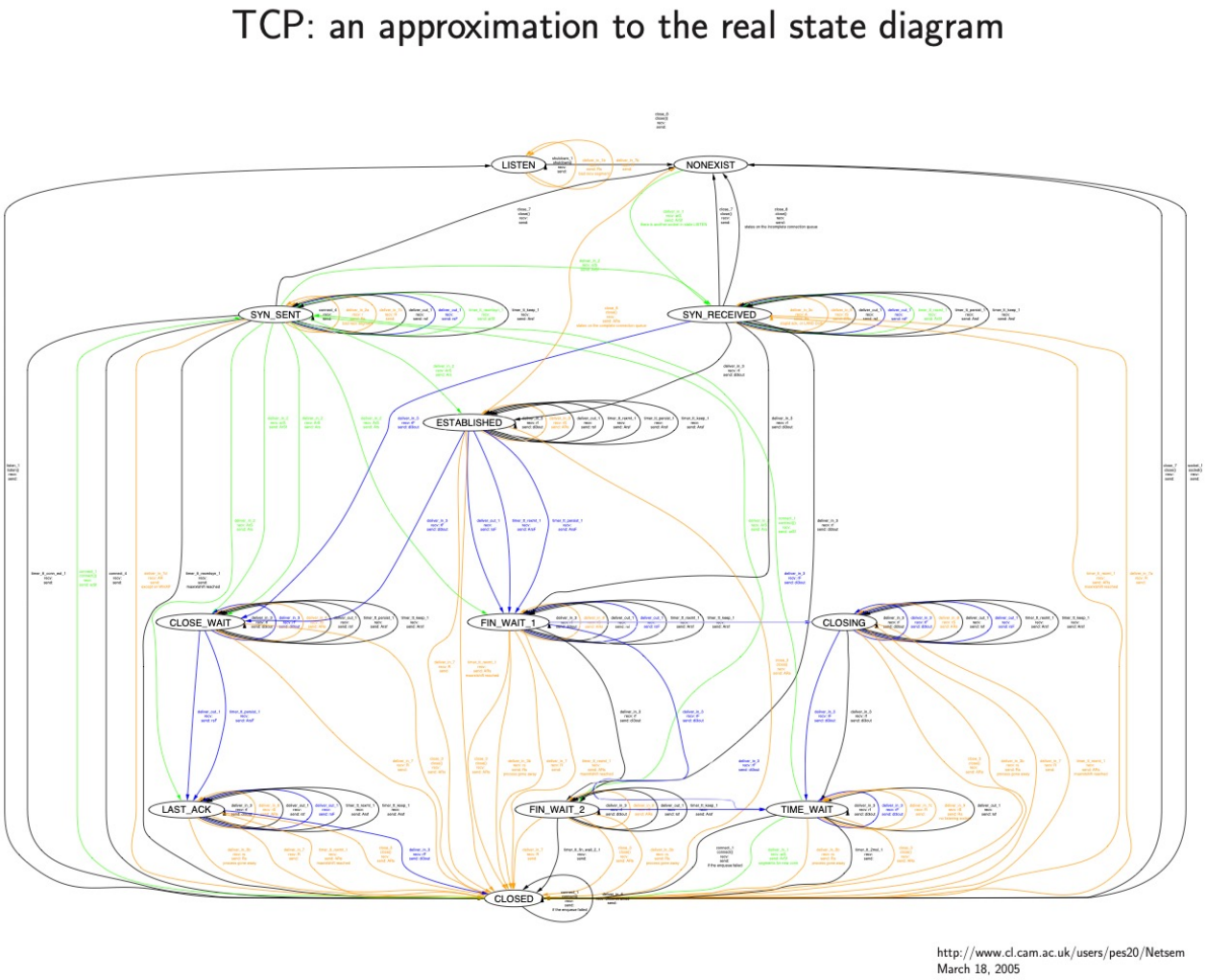
The Transmission Control Protocol (TCP)



TCP Connection State Diagram
Figure 6.

- V. Cerf, K. Dalal, and C. Sunshine, ***Transmission Control Protocol (version 1)***, INWG General Note #72, December 1974.
- In practice: J. Postel, Ed., ***Transmission Control Protocol: Protocol Specification***, RFC 793, September, 1981.

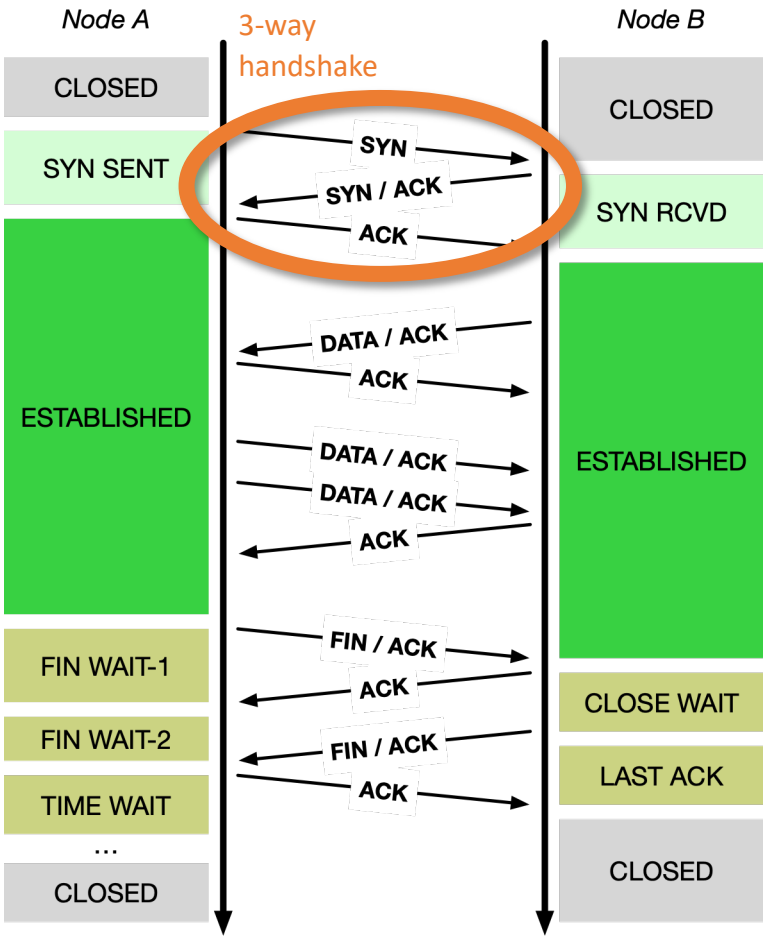
Compare to Bishop, et al (2005)





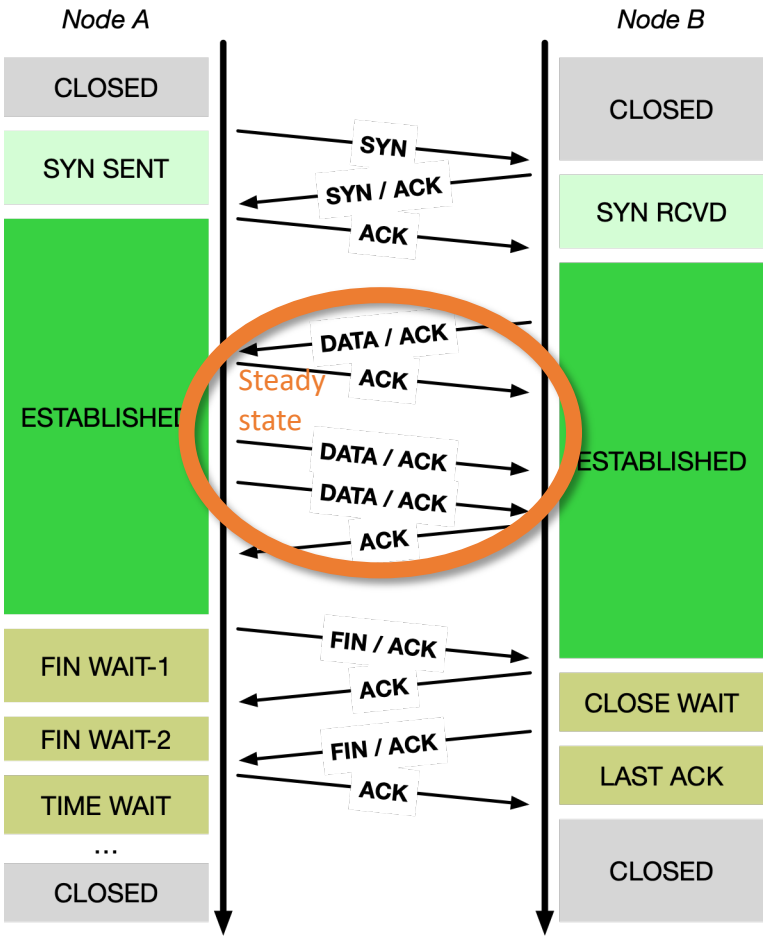
Steve Bishop, Matthew Fairbairn, Michael Norrish, Peter Sewell, Michael Smith, and Keith Wansbrough. **Rigorous Specification and Conformance Testing Techniques for Network Protocols, as Applied to TCP, UDP, and Sockets**. Proceedings of SIGCOMM 2005, ACM, 2005.

TCP principles and properties



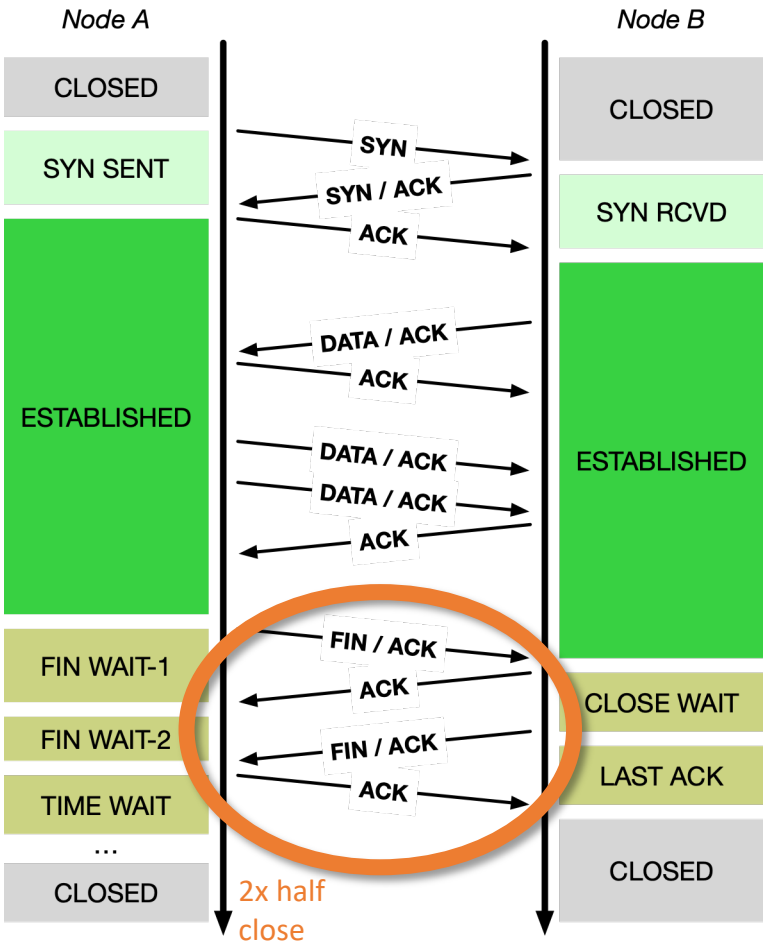
- Assumptions: Network may delay, (reorder), drop, corrupt IP packets
- TCP implements reliable, ordered, stream transport protocol over IP
- Three-way handshake: SYN / SYN-ACK / ACK (mostly!)
- Steady state
  - Sequence numbers ACK'd
  - Round-Trip Time (RTT) measured to time out loss
  - Data retransmitted on loss
  - Flow control via advertised window size in ACKs
  - Congestion control ('fairness') detects congestion via loss (and, recently, via delay: BBR)
- NB: "Half close" allows communications in one direction to end while the other continues

TCP principles and properties



- Assumptions: Network may delay, (reorder), drop, corrupt IP packets
- TCP implements reliable, ordered, stream transport protocol over IP
- Three-way handshake: SYN / SYN-ACK / ACK (mostly!)
- Steady state
  - Sequence numbers ACK'd
  - Round-Trip Time (RTT) measured to time out loss
  - Data retransmitted on loss
  - Flow control via advertised window size in ACKs
  - Congestion control ('fairness') detects congestion via loss (and, recently, via delay: BBR)
- NB: "Half close" allows communications in one direction to end while the other continues

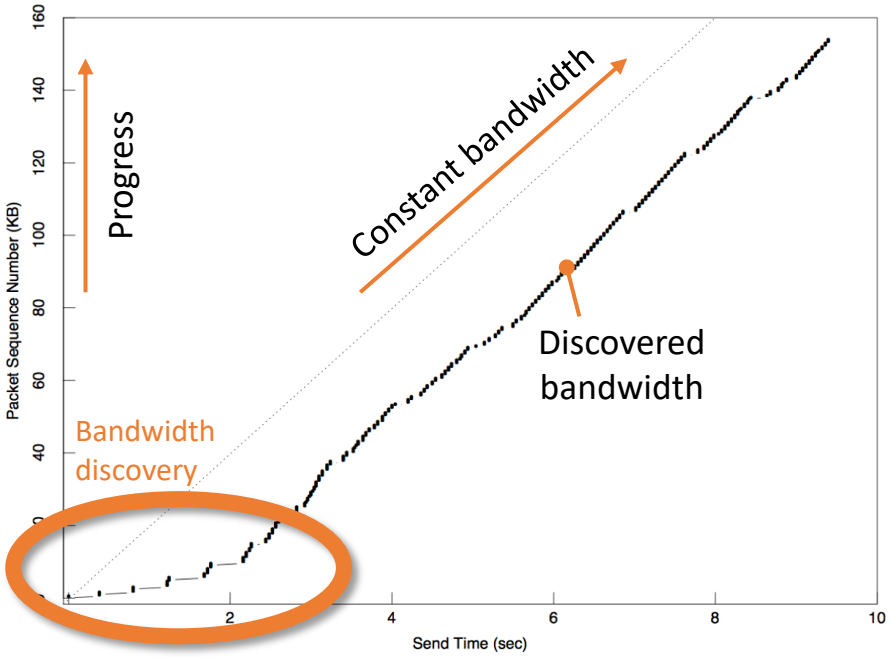
TCP principles and properties



- Assumptions: Network may delay, (reorder), drop, corrupt IP packets
- TCP implements reliable, ordered, stream transport protocol over IP
- Three-way handshake: SYN / SYN-ACK / ACK (mostly!)
- Steady state
  - Sequence numbers ACK'd
  - Round-Trip Time (RTT) measured to time out loss
  - Data retransmitted on loss
  - Flow control via advertised window size in ACKs
  - Congestion control ('fairness') detects congestion via loss (and, recently, via delay: BBR)
- NB: "Half close" allows communications in one direction to end while the other continues

TCP congestion control and avoidance

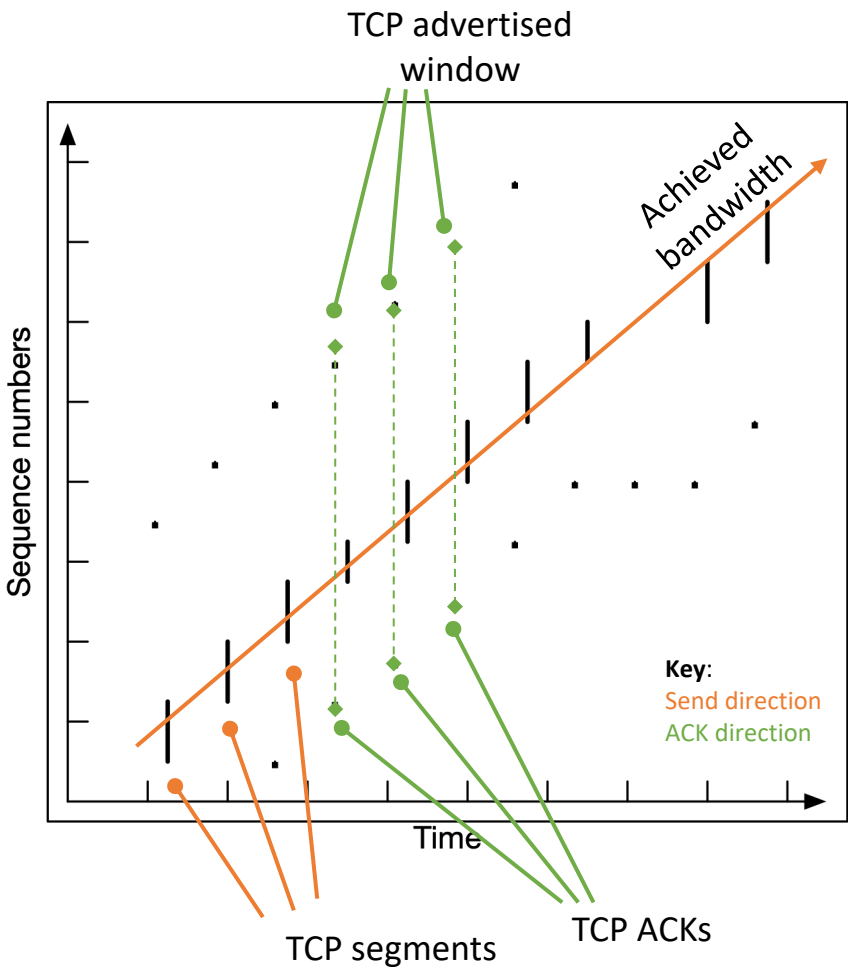
Figure 4: Startup behavior of TCP with Slow-start



Same conditions as the previous figure (same time of day, same Suns, same network path, same buffer and window sizes), except the machines were running the 4.3+TCP with slow-start. No bandwidth is wasted on retransmits but two seconds is spent on the slow-start so the effective bandwidth of this part of the trace is 16 KBps — two times better than figure 3. (This is slightly misleading: Unlike the previous figure, the slope of the trace is 20 KBps and the effect of the 2 second offset decreases as the trace lengthens. E.g., if this trace had run a minute, the effective bandwidth would have been 19 KBps. The effective bandwidth without slow-start stays at 7 KBps no matter how long the trace.)

- 1986 Internet CC collapse
  - 32Kbps → **40bps**
- Van Jacobson, SIGCOMM 1988
  - Don't send more data than the network can handle!
  - **Conservation of packets** via ACK clocking
  - Exponential retransmit timer, slow start, aggressive receiver ACK, dynamic window sizing on congestion, and (later) ABC
- ECN (RFC 3168), ABC (RFC 3465), Compound (Tan, et al, INFOCOM 2006), Cubic (Rhee and Xu, ACM OSR 2008), BBR (Cardwell, ACM Queue 2016)

TCP time/sequence graphs (Van Jacobson)



- Extracted from TCP packet traces (e.g., via `tcpdump`)
- Visualize windows, congestion response, buffering, RTT, etc:
  - X: Time
  - Y: Sequence number
- We can extract this data from the network stack directly using DTrace
  - Allows correlation/plotting with respect to other variables / events
  - E.g., TCP and socket-buffer state
- TCP time/sequence diagrams have since been extended to represent additional information
  - E.g., SACK (selective acknowledgement) blocks