L41 - Lecture 6: The Network Stack (2)

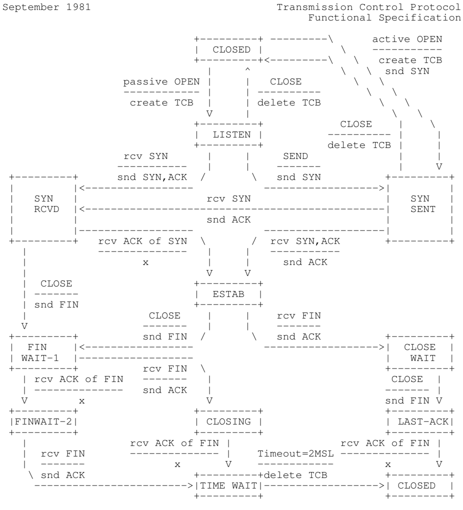
Dr Robert N. M. Watson

29 April 2015

Last time

- The Berkeley Sockets API
- Network-stack design principles
- Memory and packet flow in hardware/software
- Selected network-stack research

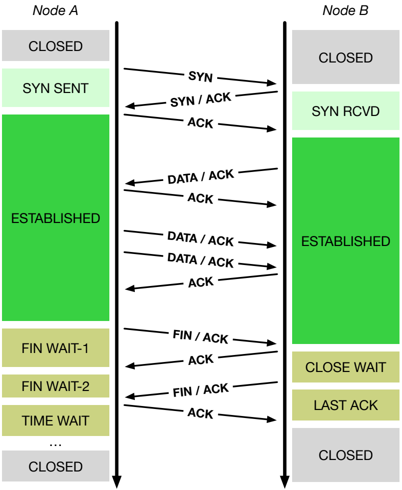
The Transmission Control Protocol (TCP)



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: Jon Postel, Ed, *Transmission Control Protocol: Protocol Specification*, RFC 793, September, 1981.

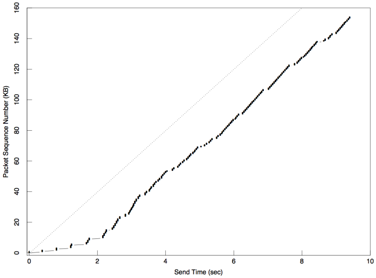
TCP goals and properties



- Reliable, ordered, byte-stream transport protocol over IP
- Three-way handshake: SYN / SYN-ACK / ACK (mostly!)
- Flow control via advertised window size in ACKs
- Congestion control via packet loss and ECN ('fairness')
- Network may delay, (reorder), drop, corrupt packets
- Sequence numbers ACK'd; data retransmitted on loss
- Round-Trip Time (RTT) measured to time out loss

TCP congestion control and avoidance

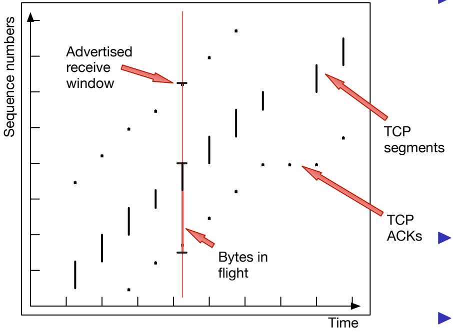
Figure 4: Startup behavior of TCP with Slow-start



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*, and *dynamic window sizing on congestion*
- ECN (RFC 3168), ABC (RFC 3465), Compound (Tan, et al, INFOCOM 2006), Cubic (Rhee and Xu, ACM OSR 2008)

TCP time/sequence graphs



- Extracted from bi-directional TCP packet traces
  - Sequence numbers in data segments, advertised window, acknowledgments
  - X: time
  - Y: sequence number
- Visualise receive windows, congestion behaviour, RTT, ...
- We can also extract this data using DTrace

BSD/FreeBSD TCP implementation evolution

1983 - 4.2 BSD: BSD sockets, TCP/IP implementation
1986 - 4.3 BSD: VJ/Karels congestion control

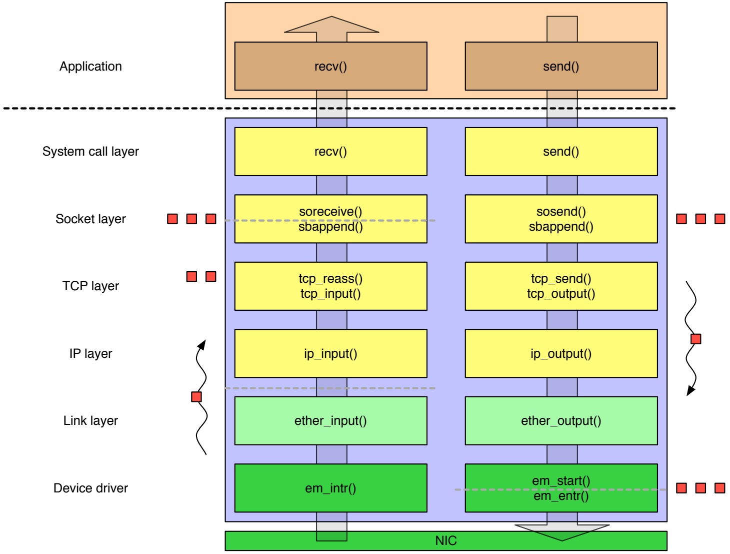
1999 - FreeBSD 3.1: `sendfile(2)`
2000 - FreeBSD 4.2: TCP accept filters
2001 - FreeBSD 4.4: TCP ISN randomisation
2002 - FreeBSD 4.5: TCP SYN cache/cookies

2003 - FreeBSD 5.0–5.1: IPv6, TCP TIMEWAIT state reduction
2004 - FreeBSD 5.2–5.3: TCP host cache, SACK, fine-grained locking

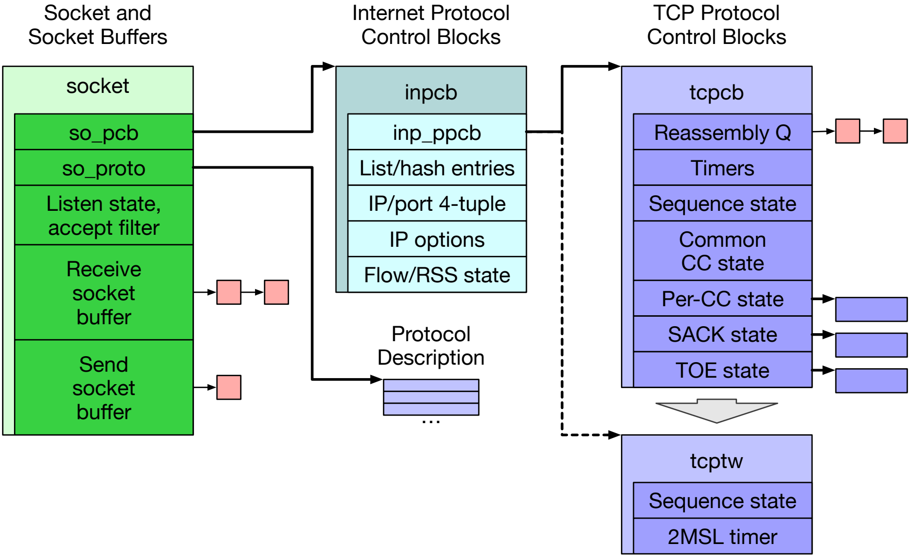
2008 - FreeBSD 6.3: TCP LRO, TSO
2008 - FreeBSD 7.0: T/TCP removed, socket-buffer autosizing
2009 - FreeBSD 7.1: read-write locking, full TCP offload
2009 - FreeBSD 8.0: TCP ECN
2012 - FreeBSD 9.0: pluggable congestion control, connection groups

- Which changes have protocol-visible effects vs. only code?

Lect. 5: Local send/receive paths in the network stack



Data structures - sockets, control blocks



Denial of Service (DoS) - state minimisation

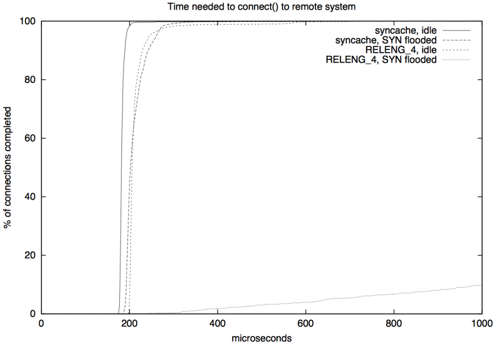
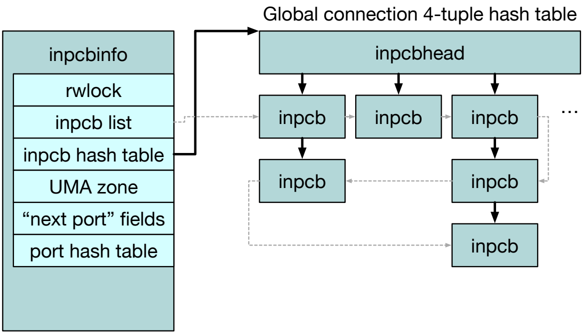


Figure 3: Time needed to connect() to remote system.

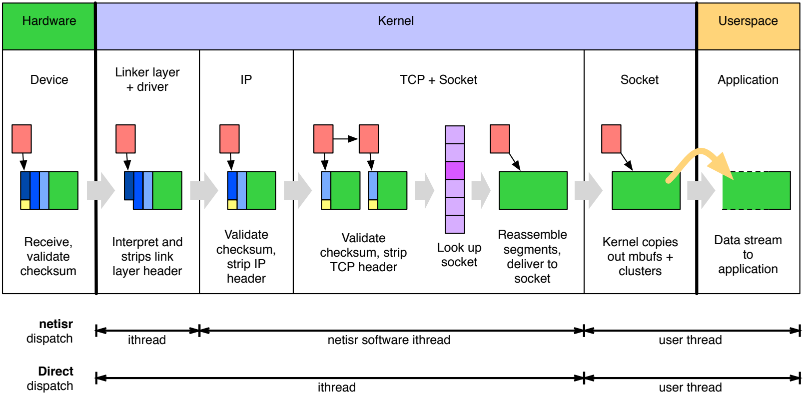
- Yahoo!, Amazon, CNN taken out by SYN floods in February 2000
- D. Borman: TCP SYN cache - minimise state for new connection
- D. Bernstein: SYN cookies - eliminate state entirely – at a cost
- J. Lemon: TCP TIMEWAIT reduction - minimise state during close
- J. Lemon: TCP TIMEWAIT recycle - release state early under load

TCP-connection lookup tables



- Global list of connections for monitoring (e.g., `netstat`)
- Connections are installed in a global hash table for lookup
- Separate (similar) hash table for port-number allocations
- Tables protected by global read-write lock as reads dominant
  - New packets more frequent than new connections

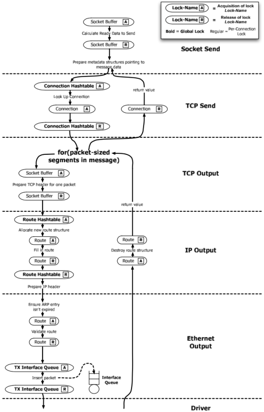
Lect. 5 - Work dispatch: input path



- Deferred dispatch - *ithread -> netisr thread -> user thread*
- Now: direct dispatch - *ithread -> user thread*
  - Pros: reduced latency, better cache locality, drop overload early
  - Cons: reduced parallelism and work placement opportunities

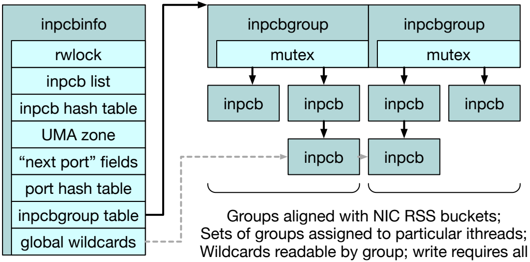
An Evaluation of Network Stack Parallelization Strategies in Modern Operating Systems

Paul Willmann, Scott Rixner, and Alan L. Cox, USENIX ATC, 2006.



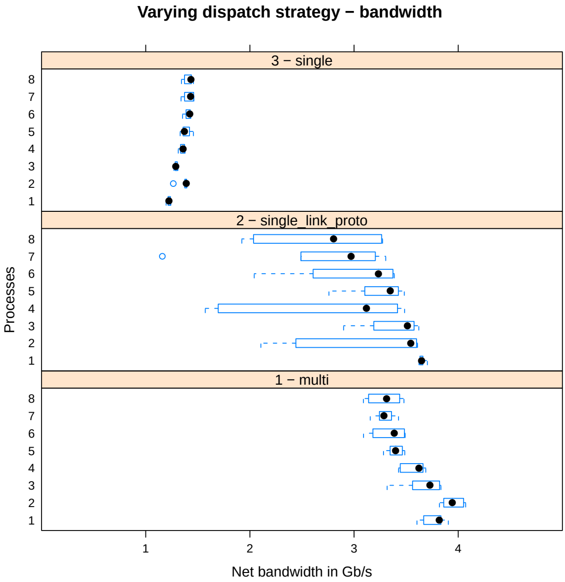
- Network bandwidth growth > CPU frequency growth
- Locking overhead (space, contention) substantial – getting ‘speedup’ is hard
- Evaluate different strategies for TCP processing parallelisation
- Message-based Parallelism
- Connection-based Parallelism (threads)
- Connection-based Parallelism (locks)
- Coalescing locks for connections by hashing 4-tuples has substantial benefit in overhead and parallelism

FreeBSD connection groups, RSS



- Connection groups blend MsgP and ConnP-L models
  - PCBs assigned to group based on 4-tuple hash
  - Lookup requires group lock, not global lock
  - Global lock retained for 4-tuple reservation (e.g., setup, teardown)
- Problem: have to look at TCP headers (cache lines) to place work!
- Microsoft: NIC Receive-Side Scaling (RSS)
  - Multi-queue NICs deliver packets to queue using hash
  - Align connection groups with RSS buckets / interrupt routing

Performance: dispatch model and locking



- 2010-vintage 4-core x86 multicore
- TCP LRO disabled
- Single queue: 1 ithread
- Single queue: 8 worker threads (1 per core)
- Multi queue: 8 queues, 8 ithreads

From architectural to micro-architectural optimisation

- Counting instructions -> counting cache misses
- Lock contention -> cache-line contention
- Locking -> identifying parallelism opportunities
- Work ordering, classification, and distribution
- NIC offload of further protocol layers, crypto
- Vertically integrate distribution and affinity
- DMA/cache interactions

Labs 4 + 5: TCP

- Build from abstract to more concrete understanding of TCP
- Use tools such as `tcpdump` and `DUMMYNET`
- Explore effects of latency on TCP performance

Lab 4 - TCP state machine and latency

- Measure the TCP state machine in practice
- Explore TCP latency vs. bandwidth (DUMMYNET)
- At what transfer size are different latencies masked?

Lab 5 - TCP congestion control

- Draw time–sequence-number diagrams
- Annotate diagrams with scheduler events
- Annotate diagrams with timer events
- Effects of latency on slow-start rampup