L41: Lab 5
TCP Latency and Bandwidth

Lecturelet 5

Dr Robert Watson / Dr Graeme Jenkinson

2019-2020

L41: Lab 5 – TCP Latency and Bandwidth

- Lab 5 topic and questions
- TCP congestion control
- TCP Protocol Control Block (TCPCB)
- ARM DTrace limitations

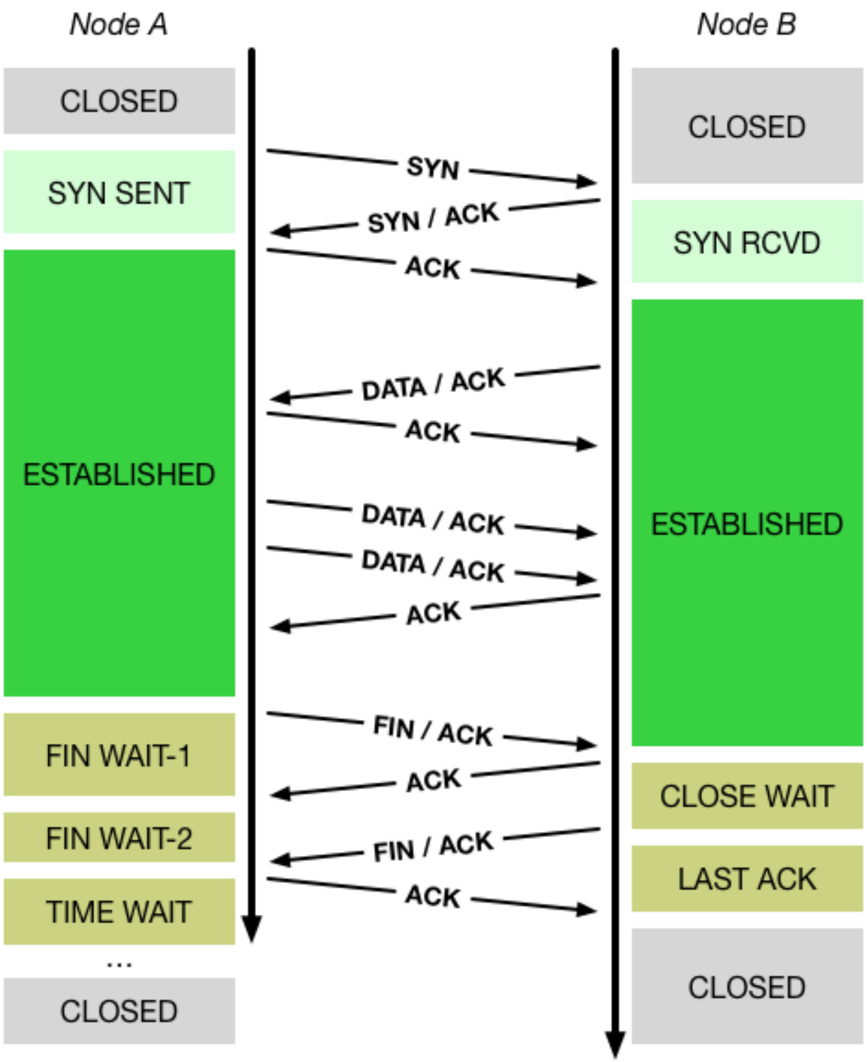
Lab 5 – TCP congestion control

- This lab explores the behavior the TCP implementation and the bandwidth it achieves as latency is varied
  - How does TCP congestion control affect bandwidth at different latencies?
  - What are the impacts of specific implementation choices and policies, such as socket-buffer auto-sizing
- As we are working over the loopback interface, we can instrument both ends of the TCP connection
  - Track packet-level headers on transmit and receive
  - Also track TCP-internal parameters such as whether TCP is in “slow start” or the steady state
- And, of course, we care about the arising probe effect

Experimental questions for the lab report

1. Plot DUMMYNET-imposed latency on the X axis and effective bandwidth on the Y axis, considering both the case where the socket-buffer size is set versus allowing it to be auto-resized.
   - Is the relationship between round-trip latency and bandwidth linear?
   - How does socket-buffer auto-resizing help, hurt, or fail to affect performance as latency varies?
2. Plot a time-bandwidth graph comparing the effects of setting the socket-buffer size versus allowing it to be auto-resized by the stack. Stack additional graphs showing the sender last received advertised window and congestion window on the same X axis.
   - How does socket-buffer auto-resizing affect overall performance, as explained in terms of the effect of window sizes?
3. Be sure to describe any simulation or probe effects.

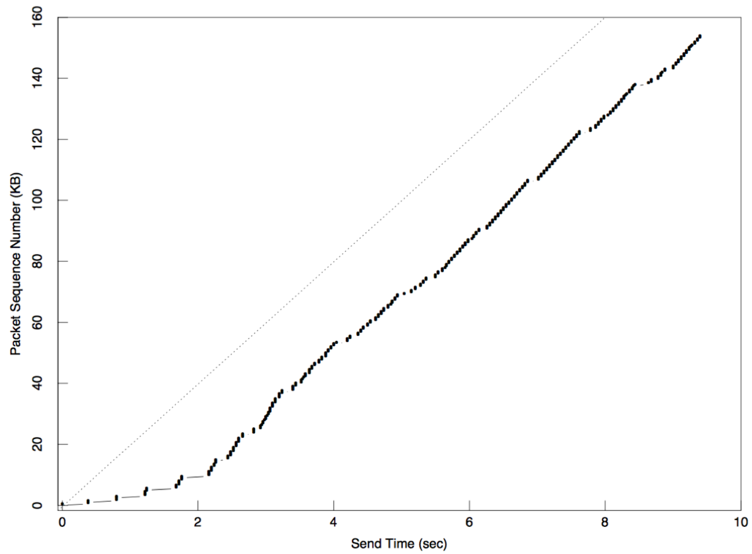
Lecture 6: TCP goals and properties



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

Lecture 6: TCP congestion control and avoidance

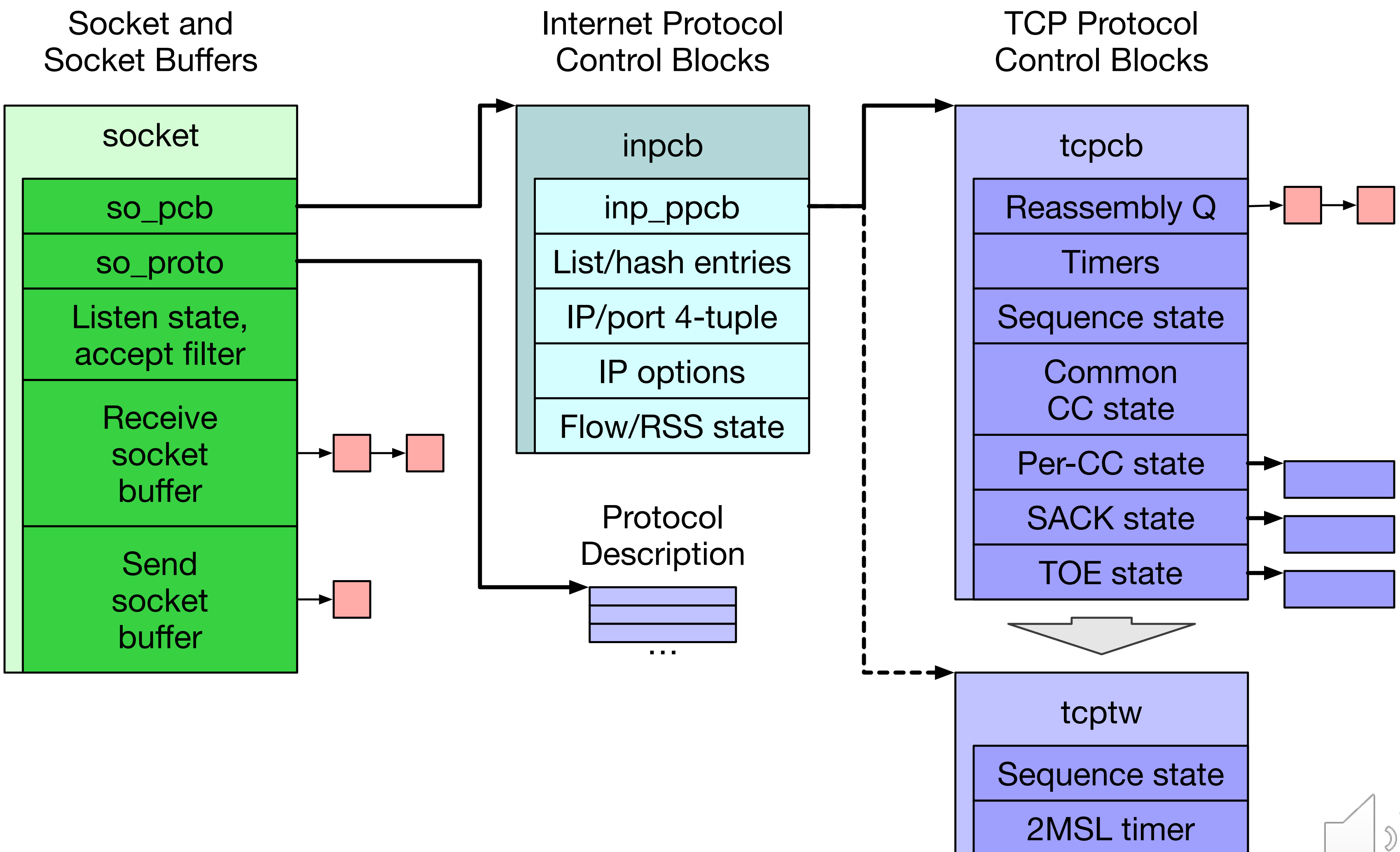
Figure 4: Startup behavior of TCP with Slow-start



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), BBR (Cardwell et al, ACM Queue 2016)

Lecture 6: Data structures – sockets, control blocks



tcpcb sender-side data-structure fields

- In this lab, there are two parties with **tcpcb**s as we run:
  - The 'client' is receiving data
  - The 'server' is sending data **← Instrument CC send state here**
- For the purposes of classical TCP congestion control, only the sender retains congestion-control state
- Described in more detail in the lab assignment:

snd_wnd	Last received advertised flow-control window.
snd_cwnd	Current calculated congestion-control window.
snd_ssthresh	Current slow-start threshold:

if (snd_cwnd <= snd_ssthresh), then TCP is in slowstart; otherwise, it is in congestion avoidance

- Instrument **tcp_do_segment** using DTrace to inspect TCP header fields and **tcpcb** state for **only the server**
  - Inspect port number to decide which way the packet is going
- NB: Flush the TCP host cache between benchmark runs

ARM DTrace limitations (1/2)

In previous years, we had suggested that the TCP segment length can be computed as follows:

```
tdatalen = ntohs(((struct ip *)args[0]->m_data)->ip_len) - ((((struct ip *)args[0]->m_data)->ip_hl << 2) + (args[1]->th_off << 2));
```

However, a bug in ARM DTrace resulted in the indexing of the `m_data` field dereferencing NULL

The TCP segment length should instead by measured directly from the `ip_length` field in the struct `ipinfo_t` structure (accessible via `args[2]` in the `tcp:::send` probe)

ARM DTrace limitations (2/2)

- FreeBSD's DTrace implementation restricts the creation of trace buffer sizes that exceed a fixed percentage of the available kernel memory
- Unfortunately, for small memory boards such as the BBB this is overly restrictive and prevents the allocation of trace buffers greater than 3MB:

```
#pragma D option bufsize=3M
#pragma D option bufresize=manual
```

- When running the benchmark, it is acceptable to limit your experiment to a total buffer sizes that does not result in drops (exceeding space in the trace buffer)

This lab session

- Ensure that you are able to properly extract both TCP header and `tcpcb` fields from the `tcp_do_segment` FBT probe.
- Generate the data for a time–bandwidth graph.
- Ask us if you have any questions or need help.