

# Advanced Operating Systems: Lab 3 - TCP

Lecturelet 3

Prof. Robert N. M. Watson

2022-2023

# Lab 3 objectives

- Further develop tracing, analysis, presentation skills around network-stack protocols and implementation
- Explore the TCP protocol **and** implementation, tracing and analysing wire-level behaviours and internal state
  - Quite different from purely packet-centric analysis
- Experiment with the interactions between TCP and variable network latency; explore:
  - TCP state-machine behaviour (**Part II assignment**)
  - TCP congestion-control behaviour (**L41 assignment**)
- Gather, analyse, and present data for the third lab submission.

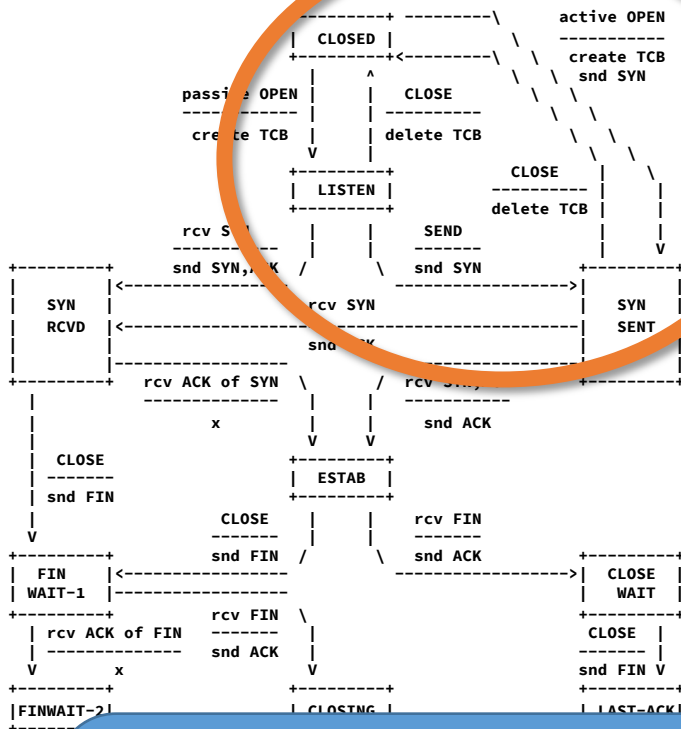
# New documents

- Advanced Operating Systems: Lab 3 – TCP – General Information (**read this first**)
- Advanced Operating Systems: Lab 3 – TCP – Part II Assignment
- Advanced Operating Systems: Lab 3 – TCP – L41 Assignment
  
- For Part II, there is a new lab notebook, which shows you how to use the GraphViz plotting package, which will ask you to use in rendering TCP state machines.
- For L41, there is no new lab notebook – you can just reuse the Lab 2 notebook as your starting point.

# Lecture 6: The Transmission Control Protocol (TCP)

September 1981

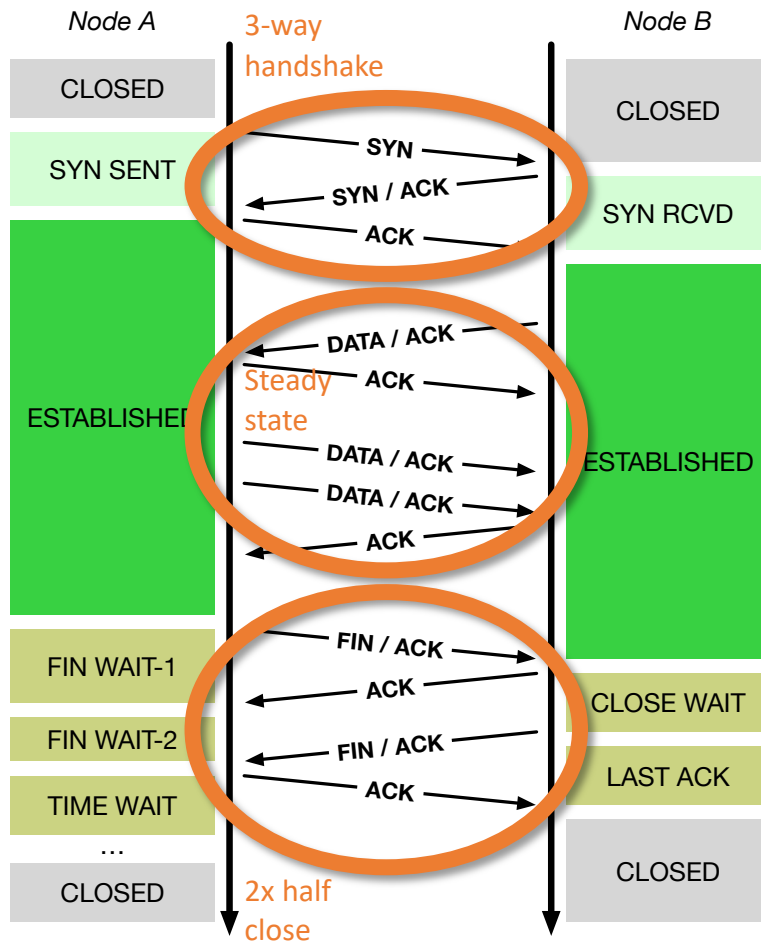
Transmission Control Protocol  
Functional Specification



- 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.

**Note:** Every TCP connection has two TCBS, one at each endpoint – each of which transits independently through the state machine. When we use loopback connections in our lab assignment, there will be two open sockets, one for each endpoint, and hence two TCP control blocks (tcpcbs). The two endpoints have inverted 4-tuples, so can be identified (with suitable care).

# Lecture 6: 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 in the IPC benchmark

```
ipc-benchmark [-Bgjqsv] [-b buffersize] [-i pipe|local|tcp] [-n iterations]  
[-p tcp_port] [-P arch|dcache|instr|tlbmem] [-t totalsize] mode
```

Modes (pick one - default 1thread):

```
1thread      IPC within a single thread  
2thread     IPC between two threads in one process  
2proc       IPC between two threads in two different processes  
describe    Describe the hardware, OS, and benchmark configurations
```

Optional flags:

```
-B           Run in bare mode: no preparatory activities  
-g           Enable getrusage(2) collection  
-i pipe|local|tcp Select pipe, local sockets, or TCP (default: pipe)  
-j           Output as JSON  
-p tcp_port   Set TCP port number (default: 10141)  
-P arch|dcache|instr|tlbmem Enable hardware performance counters  
-q           Just run the benchmark, don't print stuff out  
-s           Set send/receive socket-buffer sizes to buffersize  
-v           Provide a verbose benchmark description  
-b buffersize Specify the buffer size (default: 131072)  
-n iterations Specify the number of times to run (default: 1)  
-t totalsize Specify the total I/O size (default: 16777216)
```

- `-i tcp`                    **Set IPC type to TCP**
- `-p 10141`                 **Set TCP port number**

# Loopback networking, IPFW, DUMMYNET

- Loopback network interface
  - Synthetic local network interface: packets “loop back” when sent
  - Interface name lo0
  - Assigned IPv4 address 127.0.0.1
  - **Set the MTU to 1500 bytes**
- IPFW – IP firewall by Rizzo, et al.
  - Numbered rules classify packets and perform actions
  - Actions include accept, reject, and inject into DUMMYNET
  - **Set up IPFW to match port 10141 and inject into DUMMYNET**
- DUMMYNET – Link simulation tool by Rizzo, et al.
  - Impose simulated network conditions (e.g., latency) on “pipes”
  - **Configure DUMMYNET pipes as required for the assignment**

# Some TCP-relevant DTrace probes

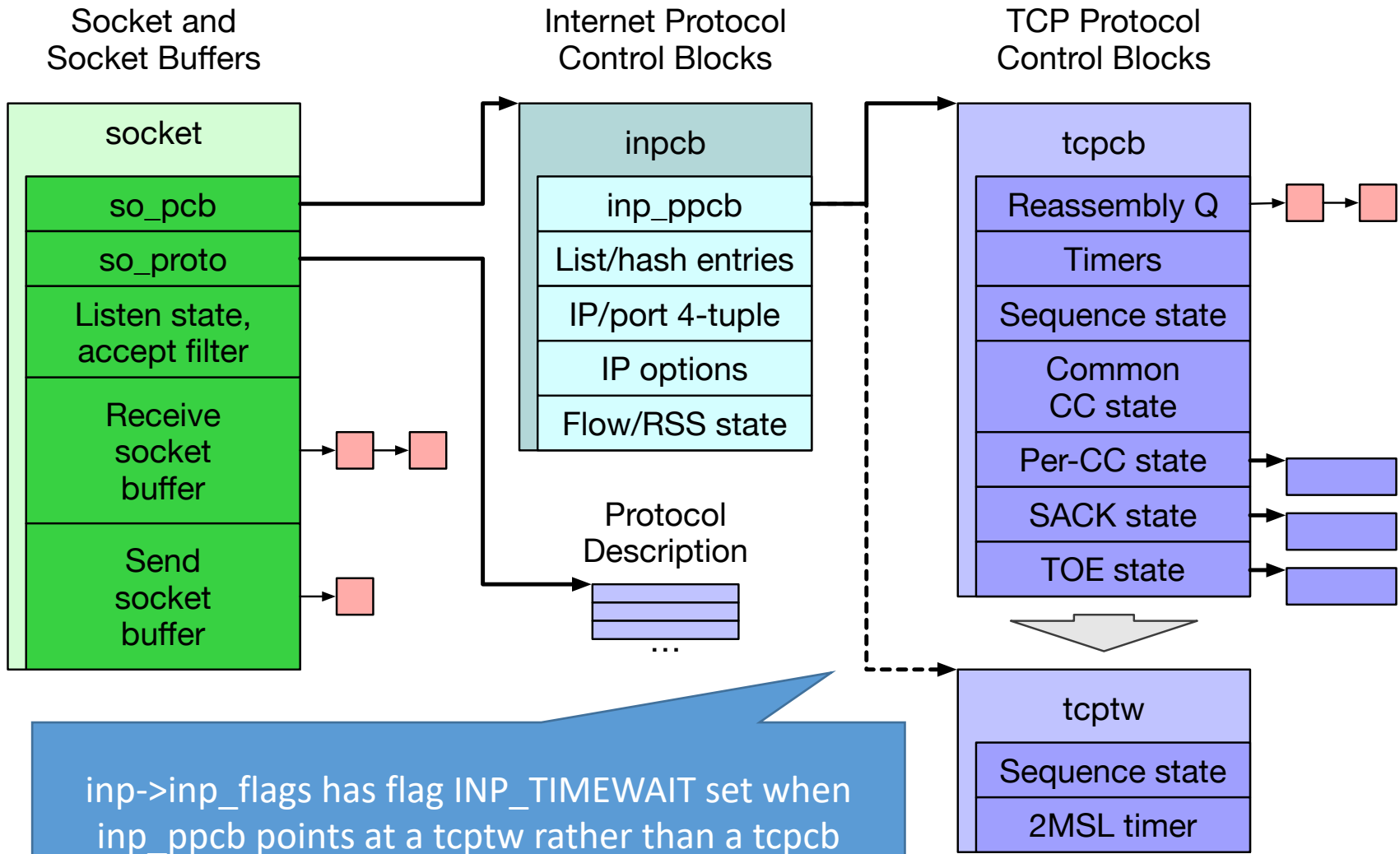
- Described in more detail in the lab assignment:

<code>fbt::synccache_add:entry</code>	TCP segment installs new SYN-cache entry
<code>fbt::synccache_expand:entry</code>	TCP segment converts SYN-cache entry to full connection
<code>fbt::tcp_do_segment:entry</code>	TCP segment received post-SYN cache
<code>fbt::tcp_state_change:entry</code>	TCP state transition

- We are using implementation-specific probes (FBT) rather than portable TCP provider probes in order to:
  - avoid the 5-argument limit to FreeBSD/arm64 DTrace; and
  - provide easier access to internal data structures
- Do not limit yourself to only these probes!



# Lecture 6: Data structures – sockets, control blocks



inp->inp\_flags has flag INP\_TIMEWAIT set when inp\_ppcb points at a tcptw rather than a tcpcb

## Part II: The TCP state machine

How does the TCP implementation state machine differ from the TCP protocol specification? How does latency affect transition through the state machine?

- Plot an effective (measured) TCP state-transition diagram for both directions of a flow
- Label the state-transition diagram with causes – TCP headers, system calls, timer, etc.
- Compare the diagram with RFC 793
- What observations can we make about state-machine transitions as latency increases?
- Describe any apparent simulation or probe effects

## Part II: `tcpcb` sender-side data-structure fields

- In this lab, two parties have **`tcpcbs`** as we run:
  - The 'client' is receiving data
  - The 'server' is sending data

} Instrument state transitions in both
- Described in more detail in the lab assignment:
  - **`t_state`** Current TCP state in a `tcpcb`
- Note that during connection setup and teardown, there may not be a `tcpcb` present

# L41: Latency and TCP congestion control

- This lab explores how latency and TCP congestion control interact to affect achieved bandwidth:
  - How do slow start and congestion control interact?
- 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

# L41: `tcpcb` sender-side data-structure fields

- In this lab, there are two parties with `tcpcbs` 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

# L41: Lab 3 hypothesis

1. Longer round-trip times extend the period over which TCP slow start takes place, but TCP is able to achieve equivalent throughput through rapid identification of, and adaptation to, available bandwidth.

# L41: Experimental questions for the lab report

1. Explore how varying latency effects TCP bandwidth.
  - Vary latency, measure bandwidth.
  - Plot a **latency-bandwidth graph** over many connections:
    - X axis: DUMMYNET-imposed latency.
    - Y axis: Distribution of achieved bandwidths over connections.
2. Measure each of the following dependent variable distributions for each latency point, and plot them using stacked plots aligned to the same X axis as the bandwidth plot:
  - time to (first) exiting slow start
  - last advertised and congestion windows
  - time spent in {kernel, user, idle}
3. Using this data, and other data you may gather using DTrace, to explain why TCP performance varies as it does as latency varies.
4. Confirm or reject the hypothesis – and explain why.

# Get in touch if you need a hand

- Attend the in-person Lab 3 session
  - If you can't, contact me to book a 1:1 supervision session
- After that:
  - You can reach us on Slack – we try to reply quickly
  - We are happy to arrange 1:1 supervision sessions during the assignment period as you work through the lab
  - Or drop me email directly