Part II - Advanced Operating Systems: Lab 3 - TCP

Dr Robert N. M. Watson

2020-2021

This is Part II Lab 3. If you are an L41 student, please see the other lab variant.

Your lab assignment will analyse the TCP implementation state machine, comparing it to the protocol state machine as specified. You will also explore (and explain) behavioural artifacts in the state machine that you discover when running the IPC benchmark in `tcp` mode while varying latency using the DUMMYNET network simulation tool.

Students may also wish to investigate latency and bandwidth interactions explored in the corresponding L41 assignment, but that is not required to complete the Part II lab assignment, and that analysis should not be included in your submission.

Hypotheses

In this lab, you will test and explore two hypotheses:

1. *Network stacks implement the TCP state machine as specified.*
2. *Transitions through the TCP state machine are insensitive to latency.*

Be sure to consider any apparent probe, tracing, or simulation effects that may arise in your experimentation.

1 Background: Graphviz and Python-Graphviz

Graphviz is a widely used open-source tool for plotting graphs[1]. Using a simple textfile input, a set of nodes, edges, labels, and also layout directives, can be used to generate graphical output. We have installed Python-Graphviz on your RPi4 board, and the Lab 3, Part II example Jupyter notebook includes a simple demonstration of its use[2]. We will ask you to use Graphviz in plotting TCP state diagrams.

2 Approach

You will configure the IPC benchmark to use the `tcp` socket type, run only in `2thread` mode, leave the total I/O size as 16MiB, and use the default socket-buffer settings (i.e., do not specify `-s`). Simulated network latency will be varied using DUMMYNET. State-machine transitions will be tracked using DTrace, and plotted using Graphviz.

[1]`http://www.graphviz.org/documentation/`

[2]`https://graphviz.readthedocs.io/en/stable/manual.html`

3 DTrace probes and tcpcb fields

You should begin by instrumenting the `fbt::tcp_state_change:entry` probe described in the main lab document. This function is called when the state associated with a `struct tcpcb` (TCP control block) is changed. When inspecting a `tcpcb`, the `t_state` field contains the current TCP connection state. The D `tcp_state_string` array can be indexed using `t_state` to return the text name of the state (e.g., ESTABLISHED), which is substantially more human friendly than a numeric state.

4 Experimental questions: Latency and the TCP state machine

1. With no synthetic latency introduced, plot an effective (i.e., as measured) TCP state-transition diagram for the two directions of a single TCP connection: states will be nodes, and transitions will be edges. Where state transitions diverge between the two directions, be sure to label edges indicating 'client' vs. 'server'.
2. Extend the diagram to indicate, for each edge, the TCP header flags of the received packet triggering the transition, or the local system call (or other event – e.g., timer) that triggers the transition.
3. Compare the graphs you have drawn with the TCP state diagram in RFC 793, describing and explaining any differences.
4. Using DUMMYNET, explore the effects of simulated latency at 5ms intervals between 0ms and 40ms. What observations can we make about state-machine transitions as latency increases, and why do any diferences arise?
5. Describe any apparent simulation or probe effects that may be affecting the fidelity of the state machines you have measured and plotted.