TCP Behaviour and Congestion Control

Networked Systems (H) 2019-2020 – Laboratory Exercise 4
Dr Colin Perkins, School of Computing Science, University of Glasgow

1 Introduction

The laboratory exercises for Networked Systems (H) will introduce you to network programming in C on Unix systems using the Berkeley Sockets API, and help you understand the operation and structure of the network. The exercises will build on your knowledge of C programming from the Systems Programming (H) course, and introduce network programming in C. Other exercises will illustrate key points in the operation of the network.

The laboratory exercises are intended to complement the material covered in the lectures. Some expand on the lectures to give you broader experience in a particular subject. Others exercises cover material, such as network programming in C, that's better taught by doing than by lecturing.

There are a mixture of formative and summative exercises. The formative exercises will give you practice in programming networked systems in C, and allow you to gain further experience in C programming. They are not assessed. The two summative exercises will focus on secure communication and on understanding the behaviour of the TCP protocol. They are assessed and together are worth a total of 20% of the marks for this course.

This laboratory exercise reviews TCP behaviour and congestion control and supports the second summative exercise which is now available. **That exercise is assessed and worth 10% of the marks for this course**.

2 Preliminaries

Retrieve the file `lab04.zip` from Moodle or from the course website, and unzip it to produce the file `lab04.pcap`. Copy that file to one of the `stlinux` machines.

The `tcpdump` command captures the contents of the packets that comprise a TCP connection and records them into a packet capture (.pcap) file. The `lab04.pcap` file is an example of such a recorded TCP connection. The `tcpdump` command can also be used to inspect the contents of a packet capture. Using one of the `stlinux` machines, run the following command to display the contents of the `lab04.pcap` file:

```
tcpdump -r lab04.pcap
```

The resulting output will be around 2,000 lines long, with the first few lines looking like the following:

```
reading from file lab04.pcap, link-type EN10MB (Ethernet)
13:15:48.900817 IP 192.168.0.66.56735 > yali.mythic-beasts.com.http: Flags [S],
                seq 1852594190, win 65535, options [mss 1460,nop,wscale 6,
                nop,nop,TS val 729676479 ecr 0,sackOK,eol], length 0
13:15:48.941961 IP yali.mythic-beasts.com.http > 192.168.0.66.56735: Flags [S.],
                seq 1658809771, ack 1852594191, win 28960, options [mss 1400,sackOK,
                TS val 2521817020 ecr 729676479,nop,wscale 7], length 0
13:15:48.942010 IP 192.168.0.66.56735 > yali.mythic-beasts.com.http: Flags [.],
                ack 1, win 2060, options [nop,nop,TS val 729676520 ecr
                2521817020], length 0
```

These have been wrapped to fit this page: the actual output will be longer lines, each starting with a timestamp (with the exception of the header line at the start). To make this easier to read, you can send the output of the `tcpdump` command to a pager ("`tcpdump -r lab04.pcap | more`") or redirect it to a file ("`tcpdump -r lab04.pcap > filename`").

Review the manual page for the `tcpdump` command. This can be accessed using the command `man tcpdump` on the `stlinux` machines, or online at `https://www.tcpdump.org`. Ask one of the lab demonstrators if you are unsure how to read the output of `tcpdump`.

You will see that there are many options that can be passed to `tcpdump` to change how it captures and displays .pcap files. For example, the command "`tcpdump -v -r lab04.pcap`" will display more detail about the packet headers and contents, while the command "`tcpdump -X -r lab04.pcap`" will display the full contents of the packets captured.

3 TCP Connection Handshake

The first part of this exercise involves using `tcpdump` to review the `lab04.pcap` file to understand the initial three-way handshake that starts a TCP connection. Run `tcpdump` and look at the records for the first three packets. Observe that:

- The first packet has the `SYN` bit set in the TCP header ("`Flags [S]`"), and includes an initial sequence number, a congestion window, and some TCP options. It has zero length (i.e., it does not carry any data).
- The second packet also has the `SYN` bit set in the TCP header ("`Flags [S]`"), and includes the initial sequence number used for the server-to-client direction, a congestion window, some TCP options, and is of length zero. This packet also includes an acknowledgement for the first packet (i.e., it is a `SYN-ACK` packet).
- The third packet does not have any flags set in the TCP header ("`Flags [.]`") and merely acknowledges the second packet. Note that, by default, `tcpdump` displays the actual sequence number for the first packet sent in each direction, but displays all later sequence numbers as relative to that. This means that it will display the acknowledgement number as 1, since a packet with the `SYN` flag set is assumed to take one sequence number in TCP. You can add the `-S` option ("`tcpdump -S -r lab04.pcap`") to display the actual sequence numbers recorded in each packet, which will show the third packet sending an acknowledgement one higher than the sequence number sent in the second packet.

These first three lines, after the header, of the `tcpdump` output show the TCP three-way handshake used to establish the connection. The remainder of the output shows the data transfer and connection shutdown. Study the output of `tcpdump` to be sure you understand what it is showing.

4 TCP Data Transfer

Run the following command:

```
tcpdump -ttttt -r lab04.pcap > packets.dat
```

This formats the packets in the `tcpdump` file with timestamps displayed relative to the start of the connection, and saves the result in the file `packets.dat`. Edit the `packets.dat` file to:

1. remove packets sent from the client to the server, keeping only packets sent in the server-to-client direction;
2. remove packets sent in the initial and closing handshakes (i.e., remove packets in the `SYN`/`SYN-ACK`/`ACK` handshake at the start of the file, and in the corresponding `FIN`/`FIN-ACK`/`ACK` handshake at the end of the file); and
3. remove packets that do not contain data (i.e., remove any lines that contain `ack` but not `seq`).

The result should be a file containing one line for each packet sent from the server to client that contains data. You may edit the file manually, or you can write/use a program to make the changes.

Process the file to extract the timestamp and the last sequence number from the sequence number range from each line. Save the processed results in the file `lab04-timeseq.dat`. For example, if a line contains the text:

```
00:00:00.093456 IP yali.mythic-beasts.com.http > 192.168.0.66.56735: Flags [.], seq 1:1389, ack 122...
```

then the corresponding line in the `lab04-timeseq.dat` file would be:

```
00:00:00.093456 1389
```

Prepare a plot of the sequence number vs. timestamp for the `lab04-timeseq.dat` file. The x-axis of this plot will represent timestamp, in seconds since the start of the connection; the y-axis will represent sequence numbers. For each line in the `lab04-timeseq.dat` file, place a mark on the appropriate point on the graph, joining the points with a line. You may use any graph plotting tool you want to prepare this graph. Then, prepare a second plot, this time including only packets received in the first 200ms of the connection. Inspect the two plots to understand the data transfer behaviour of the TCP connection. Discuss with the lab demonstrators, to confirm your understanding of what these plots are showing.

- + -