

ECE435: Network Engineering – Homework 6

TCP

Due: Friday, 13 March 2026, 5:00pm

Submission Directions:

For this homework short answers will suffice.

To submit, create a document with your answers (text, pdf, libreoffice, MS Office if you must) and e-mail them to *vincent.weaver@maine.edu* by the homework deadline. Title your e-mail “ECE435 Homework 6” and be sure your name is included in the document.

Notes on tcpdump

For these examples I have been gathering network packet dumps using tcpdump. If you have a Linux machine you can gather similar traces by installing tcpdump and using a command such as:

```
sudo tcpdump -XX -i wlan0 tcp
```

to see the packets going by for TCP connections such as your HW#1 or HW#2 code. You can specify the interface with `-i`, the names of these vary with what version of Linux you have, use `ip addr` to see. `eth0` or `enol` are common for wired ethernet, `wlan0` or something like `wlp3s0` for wifi.

Questions:

1. Using the code from HW#1 we connect to a server and send this packet. The full packet looks like this:

```
0x0000:  0013 3b10 667f b827 eba8 3711 0800 4500  ..;.f..'..7...E.
0x0010:  0038 572a 4000 4006 69cc c0a8 0833 826f  .8W*@.@.i....3.o
0x0020:  2e7f bda5 0050 cdc4 6a49 3c7b 6ca5 8018  ....P..jI<{1...
0x0030:  00e5 79f4 0000 0101 080a 0104 3e58 34a8  ..y.....>X4.
0x0040:  7bc3 4745 540a                                {.GET.
```

But so far we only know about the parts of the packet starting with the TCP header which begins at offset 0x22:

```
0x0020:          bda5 0050 cdc4 6a49 3c7b 6ca5 8018  ....P..jI<{1...
0x0030:  00e5 79f4 0000 0101 080a 0104 3e58 34a8  ..y.....>X4.
0x0040:  7bc3 4745 540a                                {.GET.
```

Fill in the name of the field as well as decode the value. Decode to decimal if it makes sense, report what any flag values are, report units if appropriate. For help decoding the TCP header see the class notes or else RFC793. For the extended options RFC1323 might help too.

BEGIN TCP HEADER	Name of Field	Decoded Value
0x0022: bda5		
0x0024: 0050		
0x0026: cdc4 6a49		
0x002a: 3c7b 6ca5		
0x002e: 80		
0x002f: 18		
0x0030: 00e5		
0x0032: 79f4		
0x0034: 0000		
0x0036: 01		
0x0037: 01		
0x0038: 080a		
0x003a: 0104 3e58		
0x003e: 34a8 7bc3		
END TCP HEADER		
BEGIN DATA		
0x0042: 4745 540a		
END DATA		

2. In the frame from the previous question:

- (a) What type of server was being connected to (based on the port number)?
- (b) What percent of the total packet (all bytes received, not just TCP) was the actual data being sent (the payload)? (also, show your work if you want partial credit)

3. You use tcpdump to monitor our sockets program from HW#1. Tcpdump can decode many of the TCP fields for you, the output shown below is the decoded packet summary. Fields like “seq” and “win” are hopefully obvious. In the “Flags” field, S=SYN, P=PSH, F=FIN, etc. Also note tcpdump by default uses “relative SEQ/ACK counts” which means after the initial packet it will print the difference rather than the raw SEQ/ACK values. The first line of each packet summary contains the timestamp, and the source address/port and the destination address/port.

(a) You start the server, then connect with the client and you see the following three packets:

```
16:38:55.451412 IP macbook-air.52658 > pi3.31337:
Flags [S], seq 3649305635, win 29200,
options [mss 1460,sackOK,TS val 23595005 ecr 0,nop,wscale 7], length 0

16:38:55.451762 IP pi3.31337 > macbook-air.52658:
Flags [S.], seq 2280757527, ack 3649305636, win 28960,
options [mss 1460,sackOK,TS val 122287507 ecr 23595005,nop,wscale 7], length 0

16:38:55.451829 IP macbook-air.52658 > pi3.31337:
Flags [.], ack 1, win 229,
options [nop,nop,TS val 23595005 ecr 122287507], length 0
```

i. What just happened? How do you know?

ii. Why is the initial SEQ number a really large values like 3649305635 instead of starting at 0?

(b) Your “uppercase” server from HW#1 is running. You type "Hi" on the client and press enter and the following four packets are sent:

```
16:39:01.056250 IP macbook-air.52658 > pi3.31337:
Flags [P.], seq 1:4, ack 1, win 229,
options [nop,nop,TS val 23596406 ecr 122287507], length 3

16:39:01.056551 IP pi3.31337 > macbook-air.52658:
Flags [.], ack 4, win 227,
options [nop,nop,TS val 122288067 ecr 23596406], length 0

16:39:01.057167 IP pi3.31337 > macbook-air.52658:
Flags [P.], seq 1:4, ack 4, win 227,
options [nop,nop,TS val 122288067 ecr 23596406], length 3

16:39:01.057265 IP macbook-air.52658 > pi3.31337:
Flags [.], ack 4, win 229,
options [nop,nop,TS val 23596406 ecr 122288067], length 0
```

Describe briefly what is going on in each packet.

(c) After typing "bye" and receiving the result BYE you see the following set of packets:

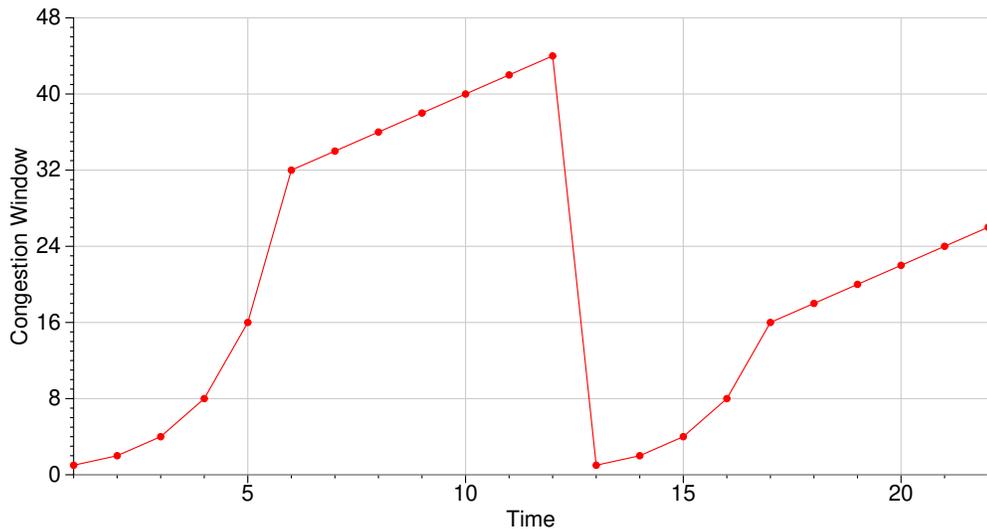
```
16:39:16.168569 IP macbook-air.52658 > pi3.31337:
Flags [F.], seq 18, ack 18, win 229,
options [nop,nop,TS val 23600184 ecr 122289578], length 0

16:39:16.168921 IP pi3.31337 > macbook-air.52658:
Flags [F.], seq 18, ack 19, win 227,
options [nop,nop,TS val 122289578 ecr 23600184], length 0

16:39:16.168974 IP macbook-air.52658 > pi3.31337:
Flags [.], ack 19, win 229,
options [nop,nop,TS val 23600184 ecr 122289578], length 0
```

What is going on with these packets?

4. Congestion



When a TCP connection encounters network congestion the behavior of the congestion window might look like the graph above. Once congestion is encountered the congestion window / slow-start threshold gets lowered.

What is one way that the TCP stack running on your machine can determine that packets are being lost over the network in a way that implies congestion?

5. Security

(a) Machine status with `ss`

You notice your computer is running slowly and the activity LED on your router is blinking rapidly. You are running Linux, and you know that running either `netstat` or `ss -a` will tell you at the socket level what's going on.

The `ss -a` command gives a report similar to the following (the `u_str` and `u_dgm` entries are removed; those are UNIX domain sockets which are local-to-the system equivalents of `tcp/udp` used for interprocess communication).

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
tcp	CLOSE-WAIT	1	0	192.168.8.38:34466	74.125.202.108:imaps
tcp	CLOSE-WAIT	1	0	192.168.8.38:35610	216.58.192.194:https
tcp	CLOSE-WAIT	1	0	192.168.8.38:36884	216.58.192.196:https
tcp	ESTAB	0	0	192.168.8.38:35496	74.125.202.109:imaps
tcp	ESTAB	0	0	192.168.8.38:36888	130.111.218.16:ssh
tcp	ESTAB	0	0	192.168.8.38:37284	23.45.134.221:https
tcp	ESTAB	0	0	192.168.8.38:42308	100.16.227.202:ssh
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53074
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53075
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53076
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53086

... 200 more just like this

tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53150
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53151
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53152
tcp	SYN-RCV	0	0	192.168.8.38:ssh	192.168.8.51:53153
tcp	TIME-WAIT	0	0	192.168.8.38:52020	172.217.9.66:https
tcp	TIME-WAIT	0	0	192.168.8.38:52242	172.217.9.34:https
tcp	TIME-WAIT	0	0	192.168.8.38:56936	216.58.192.196:https
tcp	TIME-WAIT	0	0	192.168.8.38:57386	216.58.216.106:https
udp	UNCONN	0	0	:::789	:::
udp	UNCONN	0	0	:::ipp	:::
udp	UNCONN	0	0	:::mdns	:::
udp	UNCONN	0	0	:::netbios-dgm	:::
udp	UNCONN	0	0	:::netbios-ns	:::
udp	UNCONN	0	0	:::sunrpc	:::
tcp	LISTEN	0	128	:::ssh	:::
tcp	LISTEN	0	128	:::sunrpc	:::
tcp	LISTEN	0	20	127.0.0.1:smtp	:::
tcp	LISTEN	0	50	:::microsoft-ds	:::
tcp	LISTEN	0	50	:::netbios-ssn	:::
tcp	LISTEN	0	5	127.0.0.1:ipp	:::

- i. Should you worry about the sockets in `tcp/CLOSE-WAIT`? Why or why not?
- ii. Should you worry about the sockets in `tcp/ESTAB`? Why or why not?
- iii. Should you worry about the huge number of sockets in `tcp/SYN-RCV`? Why or why not?
- iv. Should you worry about the sockets in `tcp/TIME-WAIT`? Why or why not?
- v. Should you worry about the sockets in `udp/UNCONN`? Why or why not?
- vi. Should you worry about the sockets in `tcp/LISTEN`? Why or why not?

- (b) Using `tcpdump` you also notice a lot of packets going to a different machine you have, but if you run `ss -a` it does not show all of those SYN-RECV connections. You view the system log `dmesg` and it says:

```
[1832748.308978] TCP: request_sock_TCP: Possible SYN flooding on port 22.  
                Sending cookies.  Check SNMP counters.  
[1832752.416011] DCCP: Activated CCID 2 (TCP-like)  
[1832752.416014] DCCP: Activated CCID 3 (TCP-Friendly Rate Control)  
[1832752.422698] sctp: Hash tables configured (bind 256/256)
```

What probably happened here?