Lab 05: TCP Congestion Control

Due date: **October 22, 2018**

Points: **500**

Questions: **40**

|  |  |
| --- | --- |
| Checklist (points in each section are only awarded if all bullet points are met)  Feel free to make use of this checklist as your own sanity check. | |
| Complete [140 pts] | All questions are answered. Each complete question is worth 5 points |
| Correct [350 pts] | Responses demonstrate that you have thought about your answer and that your answer is unique to you as an individual. Each correct question is worth 12.5 points |
| Name [10 pts] | Your name and NAU email are on the assignment |

The purpose of this lab is to help you examine the impact of the TCP congestion control algorithm on things like TCP window values and performance metrics, particularly in the face of network congestion. This lab is critical for ensuring you have a fundamental understanding of one of the most widely used protocols in the Internet AND it introduces you to something of an experimental design set up for networks experiments.

# Part 0: Identification

|  |  |
| --- | --- |
| Name: | Austin Kelly |
| NAU Email: | ak678    @ nau.edu |

# Part 1: Setting up Mininet

In this lab, we will be using Mininet, a virtual network that allows you to run networked applications realistically on your own computer.

Specifically, you will need to write a custom Mininet configuration file that uses the network topology below. Use the following topology with 10Mbits on all links.

h1 h2

\ /

\ /

sw1-------------sw2

/ \

/ \

h3 h4

For a template on how to create a custom topology, use the following:

<http://mininet.org/walkthrough/#custom-topologies>

I will be assuming you title your topology file as topo\_sw2\_host4.py

You can test your topology and assign 10Mbps to each link using the following command on your VM:

sudo mn --custom topo\_sw2\_host4.py --topo mytopo --link tc,bw=10 --test pingall

The test was successful if 0% of packets were dropped.

Now start Mininet session with your custom topology:

sudo mn --custom ~/mininet/custom/topo\_sw2\_host4.py --topo sw2host4 --link tc,bw=10

And start xterm for each host:

mininet> xterm h1 h2 h3 h4

# Part 2: TCP Reno and throughput

From a terminal into your VM, select Reno as your TCP congestion control algorithm

sudo sysctl net.ipv4.tcp\_congestion\_control=reno

Start Wireshark and start a capture on interface h2-eth1.

Start iperf3 servers on the xterms for h2.

Get ready to start iperf3 clients in the xterms for h1. Configure the iperf3 client to run for 60 seconds (use the *man* command if you don’t know how to do this: man iperf3). Start the iperf3 clients as simultaneously as possible.

When both clients have finished, stop capturing on Wireshark. Now, use Wireshark to filter the TCP conversation by right clicking on one of the TCP packets that have been generated and selecting “Conversation Filter” > “ F5 TCP” and save your capture session as lab05\_reno.pcapng.

Run the following in the terminal

sudo tshark –r lab05\_reno.pcapng –t ad –T fields –E separator=, -e frame.time -e tcp.analysis.bytes\_in\_flight > lab05\_reno\_alone.csv

The bytes-in-flight refers to the number of bytes a TCP sender actually sent over the line. Remember, this is determined by cwnd and rwnd.

Use RStudio (or another data analysis and graphing tool) to create a line graph of packet size over time with lab05.csv. Label the x-axis as “Time (s)” and the y-axis as “Bytes in flight.”

Q0. **Paste an image of the graph below:**

# Q1. What is the mean bytes-in-flight across the entire capture length?

Q2. What is the median bytes-in-flight across the entire capture length?

Q3. What is the standard deviation bytes-in-flight across the entire capture length?

Q4. What kind of trend do you notice with the bytes-in-flight over time?

Q5. How many times does a congestion event happen, based on your data?

Q6. Are these congestion events sensed by timeouts or triple duplicate ACKs? How can you tell?

Q7. Based on the experiment setup, what is the cause of congestion?

# Part 3: TCP Reno and UDP

Start Wireshark and start a capture on interface h2-eth1.

Start iperf3 servers on the xterms for h2 and h4.

Get ready to start iperf3 clients in the xterms for h1 (client of h2) and h3 (client of h4). Configure the iperf3 clients to run for 60 seconds (use the *man* command if you don’t know how to do this: man iperf3). **The iperf3 client on h3 should use UDP rather than TCP**. Start the iperf3 clients as simultaneously as possible.

When both clients have finished, stop capturing on Wireshark.

When both clients have finished, stop capturing on Wireshark. Now, use Wireshark to filter the TCP conversation by right clicking on one of the TCP packets that have been generated and selecting “Conversation Filter” > “ F5 TCP” and save your capture session as lab05\_reno\_udp.pcapng.

Run the following in the terminal

sudo tshark –r lab05\_reno\_udp.pcapng –t ad –T fields –E separator=, -e frame.time -e tcp.analysis.bytes\_in\_flight > lab05\_reno\_udp.csv

The bytes-in-flight refers to the number of bytes a TCP sender actually sent over the line. Remember, this is determined by cwnd and rwnd.

Use RStudio (or another data analysis and graphing tool) to create a line graph of packet size over time with lab05.csv. Label the x-axis as “Time (s)” and the y-axis as “Bytes in flight.”

Q8. **Paste an image of the graph below:**

# Q9. What is the mean bytes-in-flight across the entire capture length?

Q10. What is the median bytes-in-flight across the entire capture length?

Q11. What is the standard deviation bytes-in-flight across the entire capture length?

Q12. What kind of trend do you notice with the bytes-in-flight over time?

Q13. How many times does a congestion event happen, based on your data?

Q14. Are these congestion events sensed by timeouts or triple duplicate ACKs? How can you tell?

Q15. Based on the experiment setup, what is the cause of congestion

Q16. At what time was congestion first detected? Write the timestamp associated.

Q17. How long did it take to recover from the first congestion event? How can you tell?

Q18. Describe any differences in the number and types of congestion events that occur in Part 2 and Part 3.

# Part 4: TCP Reno and another TCP Reno stream

Start Wireshark and start a capture on interface h2-eth1.

Start Wireshark and start a capture on interface h4-eth1.

Start iperf3 servers on the xterms for h2 and h4.

Get ready to start iperf3 clients in the xterms for h1 (client of h2) and h3 (client of h4). Configure the iperf3 clients to run for 60 seconds (use the *man* command if you don’t know how to do this: man iperf3). **The iperf3 client on h3 should use TCP**. Start the iperf3 client on h1 and then about 10 seconds later, start the iperf3 client on h3.

When both clients have finished, stop capturing on Wireshark on both h2 and h4.

Now, use Wireshark to filter the TCP conversation by right clicking on one of the TCP packets that have been generated and selecting “Conversation Filter” > “ F5 TCP” and save your capture sessions as lab05\_reno\_h1.pcapng and lab05\_reno\_h3.pcapng.

Run the following in the terminal on h2:

sudo tshark –r lab05\_reno\_h1.pcapng –t ad –T fields –E separator=, -e frame.time -e tcp.analysis.bytes\_in\_flight > lab05\_reno\_h1.csv

Run the following in the terminal on h4:

sudo tshark –r lab05\_reno\_h3.pcapng –t ad –T fields –E separator=, -e frame.time -e tcp.analysis.bytes\_in\_flight > lab05\_reno\_h3.csv

The bytes-in-flight refers to the number of bytes a TCP sender actually sent over the line. Remember, this is determined by cwnd and rwnd.

Use RStudio (or another data analysis and graphing tool) to create a line graph of packet size over time with with two lines; one line corresponds to data from lab05\_reno\_h1.csv and one line corresponds to data from lab05\_reno\_h3.csv. Label the x-axis as “Time (s)” and the y-axis as “Bytes in flight.” Include a legend that labels each of the two lines.

Q19. **Paste an image of the graph below:**

# Q20. What is the mean bytes-in-flight for the h1 stream?

Q21. What is the mean bytes-in-flight for the h3 stream?

Q22. What is the median bytes-in-flight for the h1 stream?

Q23. What is the median bytes-in-flight for the h3 stream?

Q24. What is the standard deviation bytes-in-flight for the h1 stream?

Q25. What is the standard deviation bytes-in-flight for the h3 stream?

Q26. What kind of trend do you notice with the bytes-in-flight over time for both streams in relationship to each other?

Q27. Describe differences you observe between the flow you observed in Q8 and the flows you observe in Q19.