**Project 2 instructions**

**Changes required in your tcl file for project2**

At the first step of the project, you made and ran a .tcl file. In that file, you created 4 TCP agents. Change the Agent from default TCP to TCP/Linux. For instance, if you named your tcp agents tcp10, tcp11, tcp14 and tcp16, change them as follows. You only need to add /Linux to the definition of your TCP agents.

set tcp10 [new Agent/TCP/Linux]

set tcp11 [new Agent/TCP/Linux]

set tcp14 [new Agent/TCP/Linux]

set tcp16 [new Agent/TCP/Linux]

**Developing awk files to analyze the network’s trace file**

At the first step of the project you learned

1. Simulate a network in ns2 (tcl file), run it and generate a trace file (tr file).
2. Create an awk file to process the trace file and extract the amount of throughput at each 0.5 seconds time interval

At the second step, you first learn to create awk files to extract the following information from the trace file, and then analyze the results.

1. Number of received packets at the destination
2. Number of generated packets at source
3. Number of sent packets from source
4. Number of dropped packets at source
5. Number of dropped packets along the path
6. Average delay
7. Jitter
8. Average jitter

**Important note:**

In this project, we chose the packet size 1K, 2K or 3K Bytes.

The network simulator ns2 creates packets of maximum size 1K Bytes. Therefore, if we ask a CBR generator to create 3K data at each interval, it actually generates 3 packets of size 1K.

In the second step of the project, we will work with the number of packets, which implicitly represent the amount of data.

***Number of received packets at the destination***

Refer to the attached example2\_rcvd.awk file.

Run it using the trace file made by example2.tcl (given in the first step of the project).

awk -f example2\_rcvd.awk example2.tr

It creates an xls file. Draw the Scatter chart. It looks like the following chart.

A close up of a map

Description automatically generated

Now study the awk file. You see that the structure of this file is similar to throughput, and even the condition to check each record is also the same.

At the BEGIN section, we define three variables to store the number of received packets for each flow

rcvd\_packets\_flow0 = 0;

rcvd\_packets\_flow1 = 0;

rcvd\_packets\_flow2 = 0;

Then in the body section, we check whether we have passed the interval (0.5 seconds), and if yes, we print these variables into the xls file. Also we should reset these variables, so that we will calculate the number of received packets for each interval separately.

if ( timeInterval > 0.5) {

# Export number of received packets to xls file

printf("%f \t %f \t %f \t %f\n", time2, rcvd\_packets\_flow0, rcvd\_packets\_flow1,

rcvd\_packets\_flow2) > "example2\_receivedpkts.xls";

rcvd\_packets\_flow0 = 0;

rcvd\_packets\_flow1 = 0;

rcvd\_packets\_flow2 = 0;

time1 = $2;

}

**Note:**

Here we used printf command to export the results. Google printf to learn more about this command. In this example, we want to print four float numbers: time2, rcvd\_packets\_flow0, rcvd\_packets\_flow1, rcvd\_packets\_flow2. Therefore, we need to enter four %f at the first section of printf (f represents float), three \t to separate them and one \n to go to the next line. printf("%f \t %f \t %f \t %f\n", …

Then for each flow id, we should check whether the packet has arrived at the destination (the event is “r” and $4 is the destination). For instance, the following if statement checks whether the current record indicates that a packet belonged to flow id 0 has arrived at destination node n2. If yes, it increments the number of packets received at destination for flow id 0.

# if the packet arrived at destination node n2 belongs to flow id 0

if ($1 == "r" && $4 == 2 && $8 == 0) {

rcvd\_packets\_flow0++;

}

**Note:** $4 refers to the 4th column of the record. Refer to ns2-tutorial for the trace file format.

**Note:** It is important to check the flow id. There might be some other routing information exchanged among nodes that arrive on our destination. We do not want to count that unrelated information in our analysis. We only analyze the data packets belonged to the flows.

***Number of generated packets at source***

The format of the file will be similar to example2\_rcvd.awk file, which we discussed above. There are a few differences as we will discuss below.

Duplicate example2\_rcvd.awk and change the new copy’s name to example2\_gen.awk. Then make the following changes in the new file.

the BEGIN{} section, replace the initializations pf rcvd\_packets\_flow variables with gen\_packets\_flow as follows.

gen\_packets\_flow0 = 0;

gen\_packets\_flow1 = 0;

gen\_packets\_flow2 = 0;

Then in the Body of awk file, check whether we have passed the interval and if yes, enter the following lines to export the number of sent packets from source during the last interval to xls file, and also reset the variables.

# Export cumulative number of generated packets at source to xls file

printf("%f \t %f \t %f \t %f\n", time2, gen\_packets\_flow0, gen\_packets\_flow1,

gen\_packets\_flow2) > "example2\_genpkts.xls";

gen\_packets\_flow0 = 0;

gen\_packets\_flow1 = 0;

gen\_packets\_flow2 = 0;

And also enter the following lines after the interval checking to increment the variables if a packet belonged to each flow id has been entered to an outgoing queue (check $1 == "+") at source node n0 (check $3 == 0).

Note that once source node n0 generated a packet, it inserts the packet into the proper outgoing link queue ($1 == "+"). Now depending on the number of current packets at the queue and the queue size, the packet might be sent to the next hop (event "-" will occur) or dropped (event "d" will occur) at that node. We will work on "-" and "d" events later on.

# if a packet belongs to flow id 0 was generated at source node n0

if ($1 == "+" && $3 == 0 && $8 == 0) {

gen\_packets\_flow0++;

}

# if a packet belongs to flow id 1 was generated at source node n0

if ($1 == "+" && $3 == 0 && $8 == 1) {

gen\_packets\_flow1++;

}

# if a packet belongs to flow id 2 was generated at source node n0

if ($1 == "+" && $3 == 0 && $8 == 2) {

gen\_packets\_flow2++;

}

You may also want to correct the comments to indicate that example2\_gen.awk finds the number of generated packets at source (not the received packets at destination as example2\_rcvd.awk).

***Number of sent packets at source***

Again, we should initialize required variables at the BEGIN section. You could name them as sent\_packets\_flow

Then in the Body of awk file, check whether we have passed the interval and if yes, add the required printf to export the number of sent packets at source during the last interval to an xls file (sent.xls), and also reset the variables.

Then, add the required lines after the interval checking to increment the variables if a packet belonged to each flow id has been sent. For instance, the following command checks whether a packet belonged to flow id 0 ($8 == 0) was sent ($1 == "-") from node n0 ($3 == 0). If yes, increments sent\_packets\_flow0 by 1. Not that you should do this for all required flows.

# if a packet belongs to flow id 0 was sent from node n0

if ($1 == "-" && $3 == 0 && $8 == 0) {

sent\_packets\_flow0++;

}

***Number of dropped packets at source***

Again, we should initialize required variables at the BEGIN section. You could name them as droppedatsource\_packets\_flow

Then in the Body of awk file, check whether we have passed the interval and if yes, add the required printf to export the number of dropped packets at source during the last interval to an xls file (droppedatsource.xls), and also reset the variables.

Then, add the required lines after the interval checking to increment the variables if a packet belonged to each flow id has been dropped. For instance, the following command checks whether a packet belonged to flow id 0 ($8 == 0) was dropped ($1 == "d") at the outgoing queue of node n0 ($3 == 0). If yes, increments droppedatsource\_packets\_flow0 by 1. Not that you should do this for all required flows.

# if a packet belongs to flow id 0 was dropped at node n0

if ($1 == "d" && $3 == 0 && $8 == 0) {

droppedatsource\_packets\_flow0++;

}

***Number of dropped packets at some node***

If you want to know how many packets belonged to a given flow id have been dropped at some particular node other than the destination, you could use similar awk file like the above.

1. You need to change the node number $3. For instance, If you want to check how many packets of flow id 0 were dropped at node n2, use the following command. Note that you should initialize, export to xls and reset the droppedatnode2\_packets\_flow variables as well. Not that you should do this for all required flows.
2. If the traffic has been generated by a TCP agent, you should add one more condition in your if statement to check whether the type of traffic is tcp. The reason is TCP creates some acknowledge packets and we want to separate the data packets from them. So if your agent is a TCP, add && $5 == "tcp" to the following if statement.

# if a packet belongs to flow id 0 was dropped at node n5

if ($1 == "d" && $3 == 2 && $8 == 0) {

droppedatnode2\_packets\_flow0++;

}

***Number of dropped packets along the path***

You can use the following command. Note that you should initialize, export to xls and reset the dropped\_packets\_flow variables as well. Also if the agent is tcp, check the type of packet by adding && $5 == "tcp" to the following if statement.

# if a packet belongs to flow id 0 was dropped at any node along the path

if ($1 == "d" && $8 == 0) {

dropped\_packets\_flow0++;

}

***Average end-to-end delay***

Refer to the attached example2avgdelay\_flow0.awk file for example2. Use the structure of this file to write an awk file for the given topology at project. Do that for the four given flow ids. We need this chart to analyze the network. Note that the given file finds the average delays for one flow. Add the other required flows in a similar way.

***Jitter***

Refer to the attached example2jitter\_flow0.awk file for example2. Use the structure of this file to write an awk file for the given topology at project. Note that the given file finds the jitter for one flow.

***Average jitter (Extra credit)***

I assume at this point you have enough knowledge and skills on awk text processor to create an awk file that exports the average jitter for the four flows.

**Save charts as pictures:**

Once you created the charts in Excel files, save them as png files and add the pictures to your submission folder. The following link is a good reference:

<https://support.office.com/en-us/article/save-a-chart-as-a-picture-254bbf9a-1ce1-459f-914a-4902e8ca9217>

**Network analysis**

Now that you have created all the awk files and pictures of charts for the given topology of the project, we can start analyzing the network. The expected result for sent packets is as follows.

A close up of a map

Description automatically generated

You observe that, as expected, TCP agents start at 1 second and UDP agents start at 2 seconds. Then UDP agents transmit almost the same but TCP agents send a controlled amount of data.

**Note that UDP generates as many packets as created by CBR. However, TCP limits the number of generated packets to control the amount of traffic it sends to network to avoid congestion.**

**UDP analysis**

**1.** You observe that the number of packets that UDP agents send from the source at each interval fluctuate around an average, which is usually below 20.

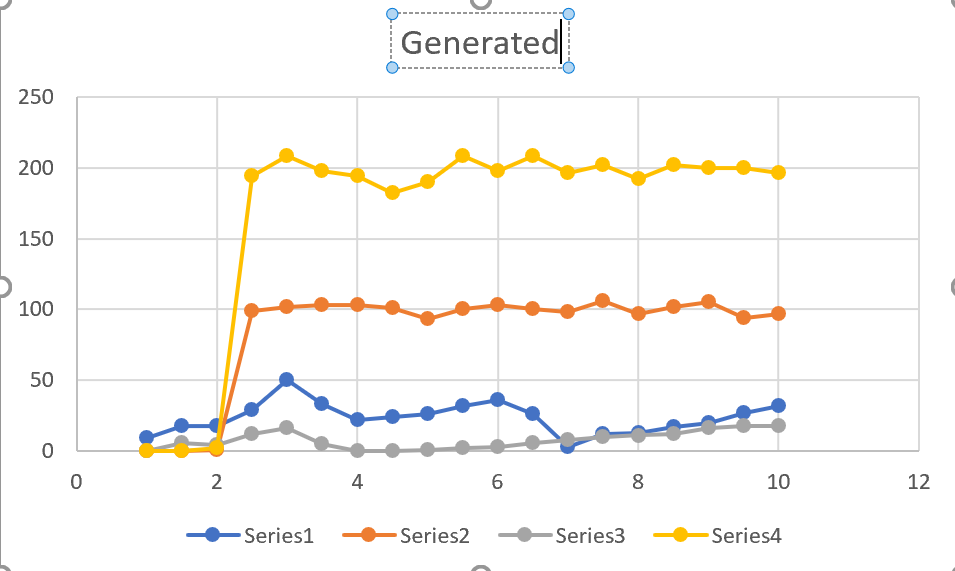
Why it does not go above 20 packets? What happens to the rest of packets generated at that interval?

UDP is stateless and doesn’t perform congestion control. Congestion control helps to control packets that can get lost in transit due to congestion. The protocol will continually send packets to destination and as a result, this is why we see a fluctuation. The average is around 20 packets per second because according to UDP protocol, each time a drop occurs the UDP protocol attempts to return to the previous send rate (20 packets / sec). On the flip side, TCP will cut back traffic in response to congestion. The number of packets won’t go beyond 20 packets because the first link (n9-n1) has queue limit of 20 packets, which becomes the bottle neck for the rest of the links

**2.** Refer to the generated packets chart UDP flows. Do you observe a meaningful difference between the number of packets each of the two flows have generated? What is the reason of this gap?

Flow 9->15 generates more packets compared to 9->12 because when the queue is full, more packets aren’t allowed to be sent until queue is freed and newly generated packets can be pushed into queue.

Node 5 will get congested because traffic is forwarded through it to the other 4 destination nodes so we are bound to see that flow 9->12 generates less packets than 9->15.

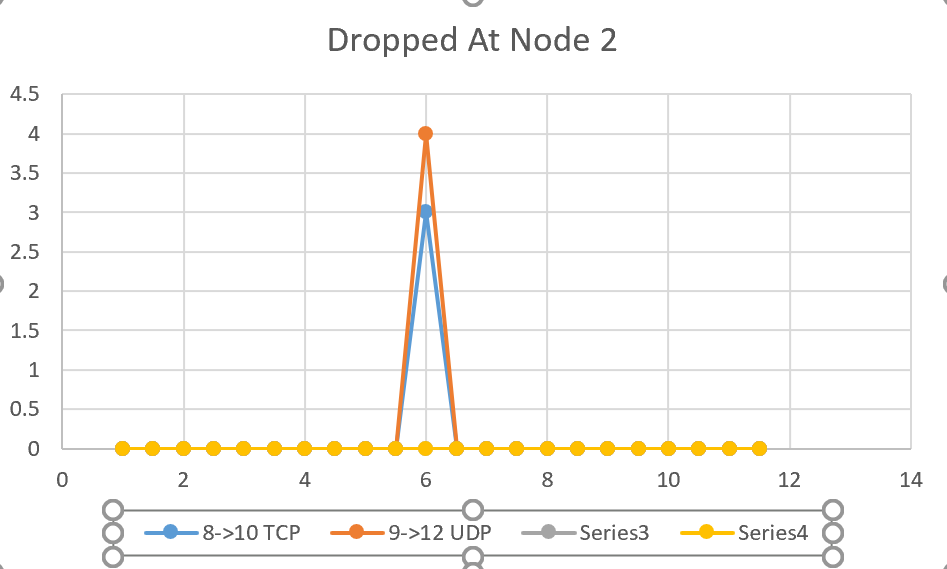
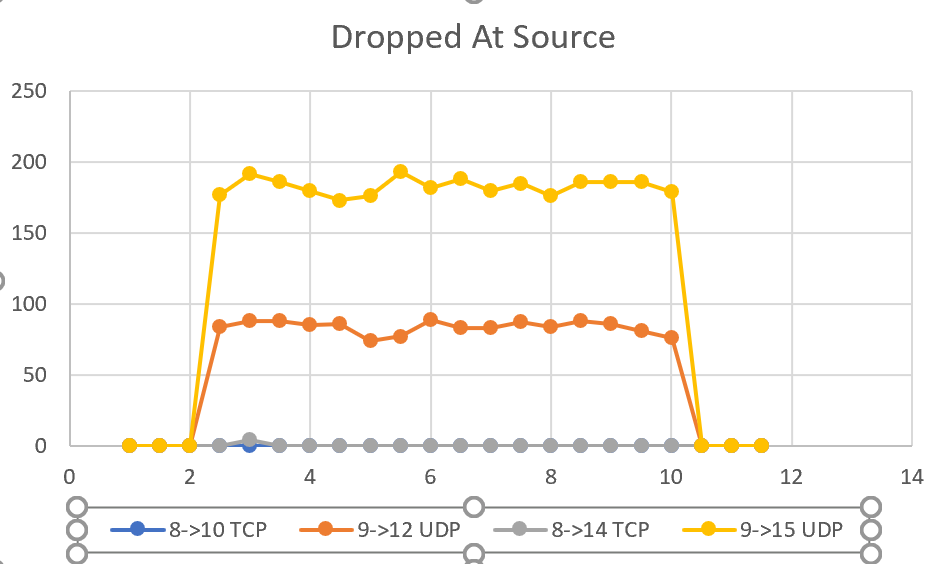


**3.** Refer to the generated packets chart and dropped\_at\_source charts for UDP flows. Is generated\_packets = sent\_packets + dropped\_packets\_at\_source?

If not, where did the rest of packets dropped and when? Did you expect that? Why?

Add a chart that shows when those packets dropped at that node.

No, generated\_packets does not = sent\_packets + dropped\_packets\_at\_source. This is because link 2-3 was down for 1 second which meant that traffic has to be re-routed to link 2->4 instead.

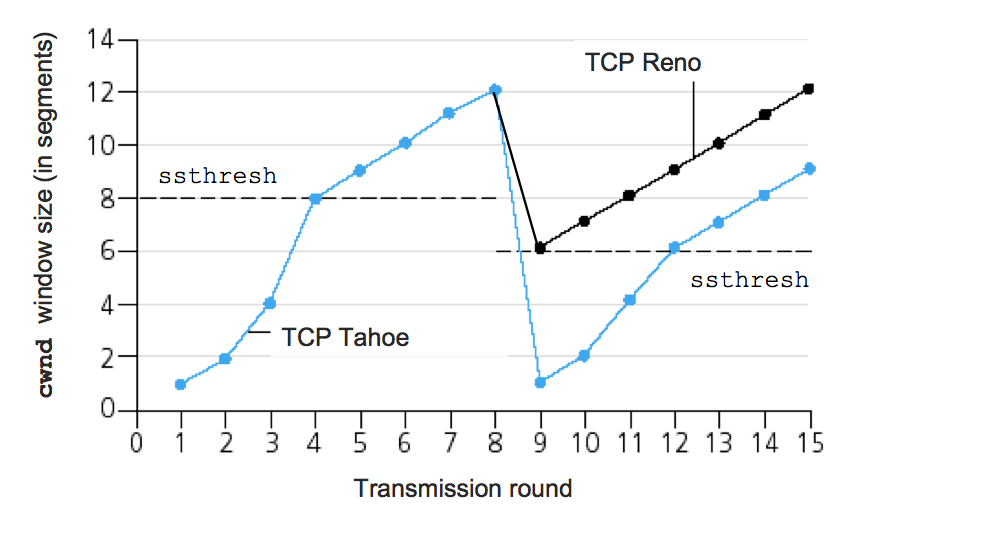


4.Refer to the average\_delay charts for UDP flows. It seems the delay is almost the same all time. But there is a big change on one of the flows. When does that happen? What is the reason of this change? (Refer to the network configuration to check what change we made at that time in the simulation). Why this change does not affect the other UDP flow?

The big difference is at six seconds, when the link from node 2 and node 3 shut down. Then, they resume again at 7 seconds which means a blocked path for one of the flows and the other flow is unchanged.

**TCP analysis**

You observe that the number of packets that TCP agents send from the source at each interval follow a pattern. There are two main TCP algorithms: Tahoe and Reno. The following picture shows the difference. They both use a window named congestion window and send as many packets as their windows allow. Each time the TCP agent receives an ack message from the destination, which confirms the packet has reached the destination correctly, the TCP agent grows its window by one. When the window size reaches some threshold, the TCP agent enters the congestion avoidance phase and grows its window slower. If, for any reason, the TCP agent does not receive an ack for a sent packet, it assumes a congestion has happened in the network resulted in a packet loss. Thus, it immediately cuts its window size to avoid sending too much data to the already congested network. TCP Reno cuts its window by half and TCP Tahoe cuts to size 1. Here in this project we used TCP Linux, which resets the window size by 0.



**5.** Is there any difference between generated traffic and sent traffic for TCP flows? What is the reason? Refer to other charts and the above discussions to answer this question.

The two TCP flow patterns appear to be similar between generated and sent traffic because TCP protocol knows to only generate the amount of traffic that matches the amount it is capable of sending.

**6.** When does the flow to node 10 cut its window? Did you expect that? Why? You also observe that at the same time, the average delay to node 10 falls to 0. What is the reason?

To complete our analysis, you also observe that the TCP flow to node 14 gets delayed at time 4 and TCP agent at source cannot hold that many packets at its current window size and it cuts its window. Thus, some packets of this flow get dropped at that time.

Me and Seren observed that the flow to node 14 gets to the window decreased to size zero around time 4.