Task 1

The task consists of analysis regarding window size influence on the effectiveness of TCP. We will calculate theoretical best window size, and then check how different sizes of the window affect our simulated TCP connection. Results will be represented in an excel graph.

Preset values

From task description:

- R1-R2 link delay: 70ms,
- access links delay: 10ms,
- R1-R2 link capacity: 10 Mbps,
- R1-R2 link buffer: 5 packets.

We have chosen init time to be 20 s, and total simulation time 120 s, so that all distinct elements in the “sawtooth” graph could be seen.

Theoretical formula for optimal window size

The formula we use is as follows:
*Opt. window size [bytes] = Bandwidth [bytes/s] * rtt [s]*

So the actual RTT is *RTT = 2 * (propagation time + 2 * delay) =*
*2 * (70 + 2 * 10) = 180 ms.*
One way trip is access link delay -> R1-R2 link delay -> access link delay. So these delays summed up and multiplied by two give us the time for the packet to be sent, and for the ACK to be received.
Bandwidth = 10^7 [bits/s] = 10^7 / 8 [bytes/s] = 1250000 [bytes/s]
*Opt. window size = 1250000 [bytes/s] * 0.18 [s] = 225000 [bytes]*

What we also need to do is divide this window size by the segment size, because in the simulation parameters we need to input TCP max window size as segments, not as bytes. Since we got predefined segment size at 1460 (which is TCP MSS - Maximum segment size of TCP packet), we get the following result:
Opt. window size = 225000 [bytes] / 1500 [bytes/segment] = 150 [segments]

Simulation results - observation

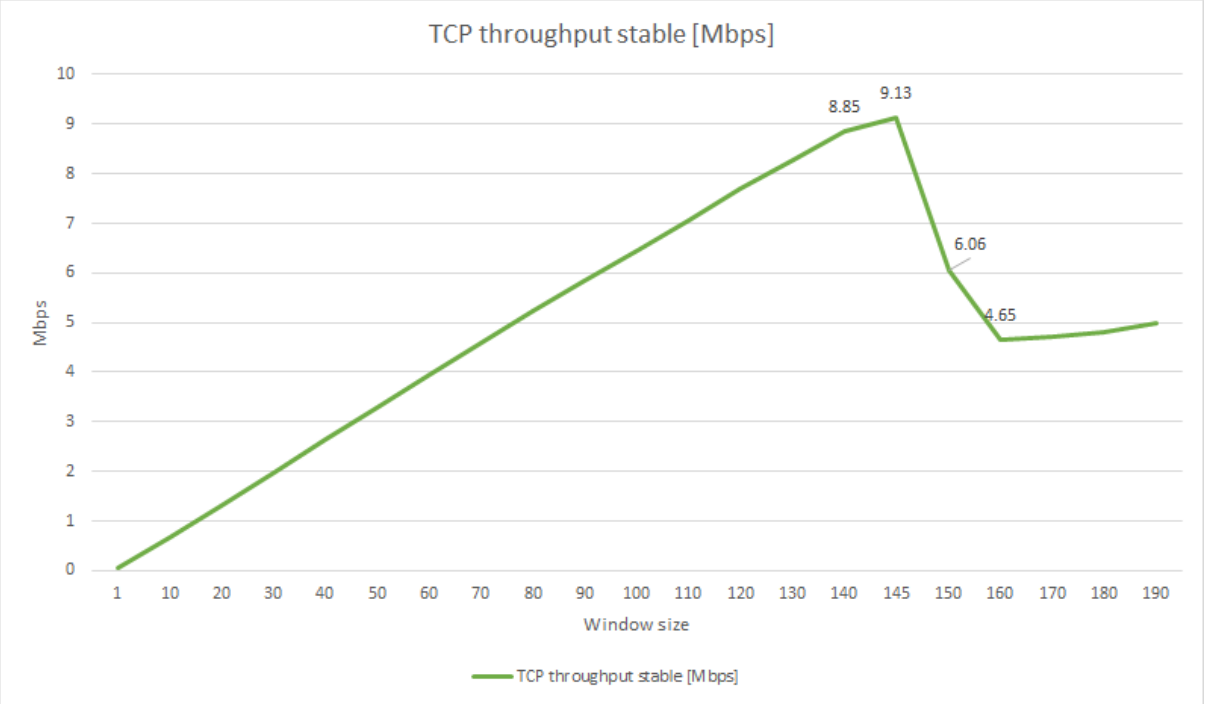
After running the test simulations for a bit, we noticed that the throughput varies despite the simulation input arguments being the same. If the result was deemed incorrect, we run the simulation again.
Also, per lab supervisor instructions, we increased the simulation time from 180 to 300 seconds, just to be extra sure graphs and simulations are correct.

TCP connection throughput dependence on window size

Per task description, we will plot a graph showing stable TCP throughput for different values of the window size. To reduce the amount of simulations run, we will take values for window size every 10 segments, except for 140, 145, 150. I.e. window size being:

1, 10, 20, 30, …, 140, 145, 150, …, 190

This is the obtained plot. Note that throughput is rounded in format x.xx:



As it can be seen from the plot, the throughput is higher the bigger the window size is, until it starts falling right before 150, which is our theoretical window size value. The biggest stable Throughput is 9.15 Mbps at window size being 145.

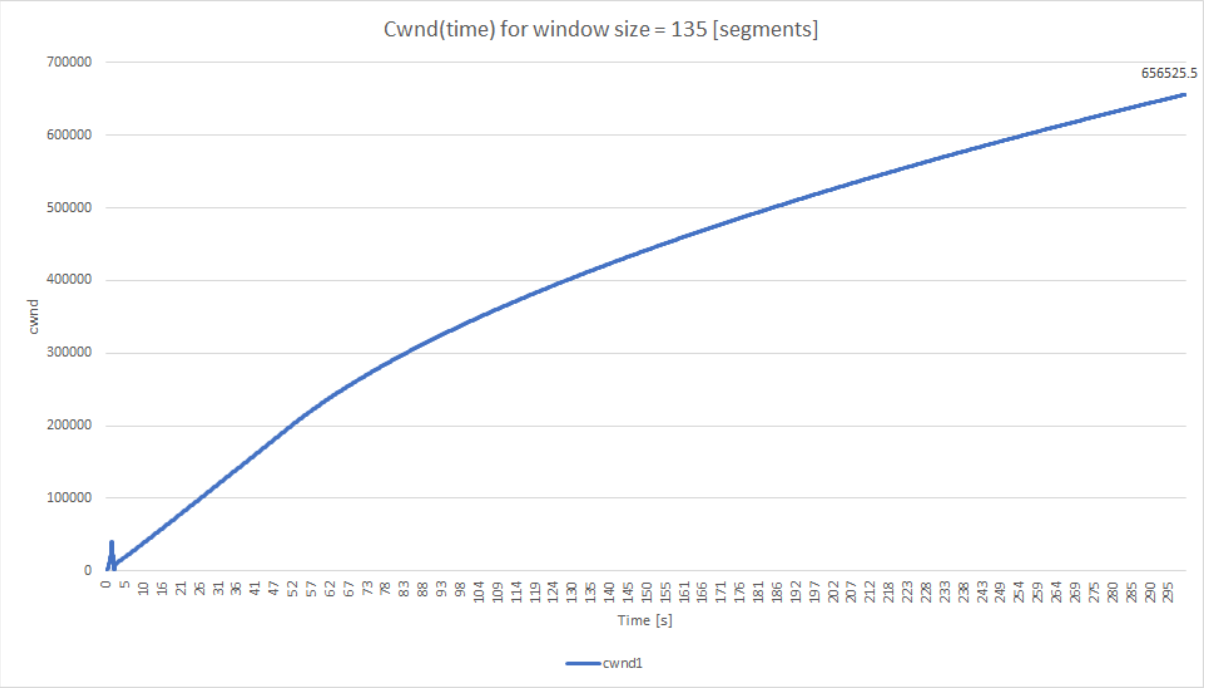
Note: Since the amount of simulations is rather big, we included the outputs of the simulations as a .log file from PuTTy terminal if they are needed.

Cwnd(time) plots for two different window sizes - analysis

Now we are going to show two Cwnd(time) plots. One is for a window size being smaller by 10% than the theoretical, the other one is for window size being larger than 10%. For window size = 135, console output:

```
Simulation time       300
Initialization time   20
Active sources        [1 0 0 1]
TCP Windows           [135 5000 5000]
Link delay            70ms
Link capacity         10Mb
Link buffer           5

TCP1 Average Throughput = 8.0932810666666661 [Mbps]
      Stable Throughput = 8.5518857142857243 [Mbps]
TCP2 Average Throughput = 0.0 [Mbps]
      Stable Throughput = 0.0 [Mbps]
TCP3 Average Throughput = 0.0 [Mbps]
      Stable Throughput = 0.0 [Mbps]
```

And the plot showing cwnd(time):

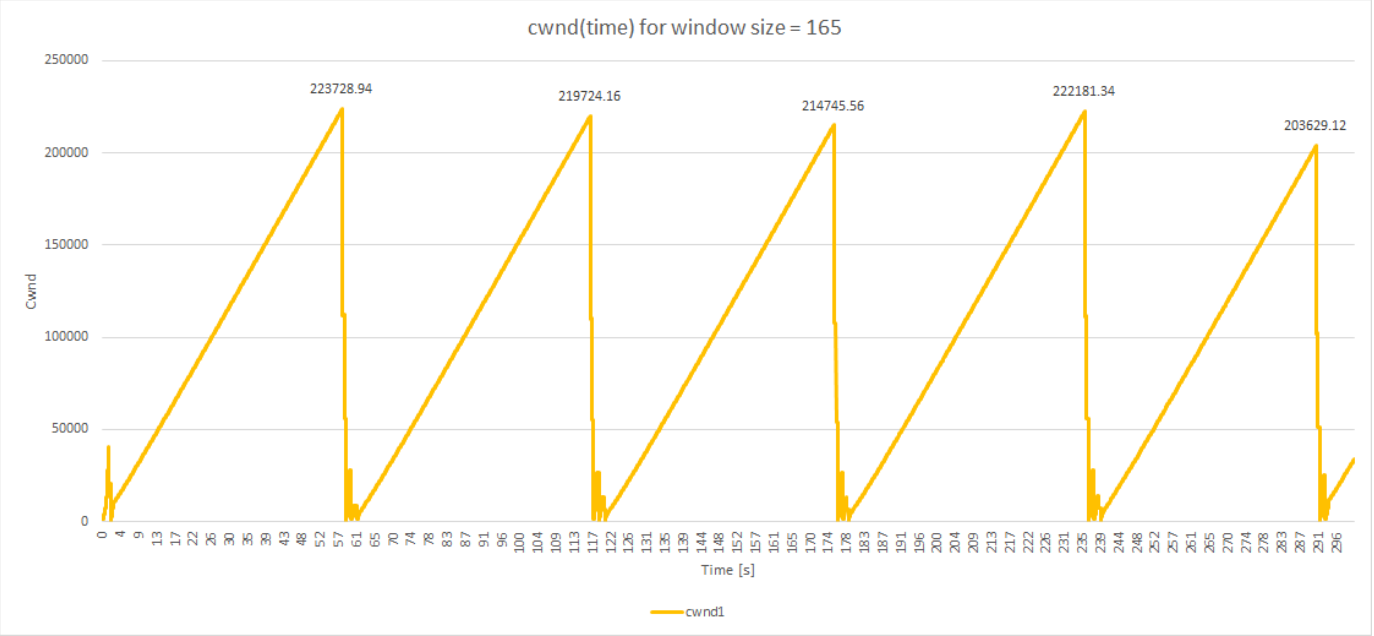


Max value of cwnd1 is shown in top-right, and it is 656525.5 bytes. The max value in the plot is bigger than the theoretical value. Furthermore, no sawtooth type of line can be seen in this plot.
Since cwnd doesn't drop by half at any time, it means no packet loss occurs (there is no congestion). This results in rather high bandwidth efficiency (85%)
This is because when the window size of 135 is reached, it doesn't increase further. So if for this window size we don't have ANY packet loss, then the cwnd will increase indefinitely. The packets will continue to transmit without packet loss.
Lack of probing for bigger window size is beneficial if we know the maximal window size for which no packet loss occurs. This allows for more stable transmission, that results in bigger throughput

Now for window size = 165. Console output:

```
Simulation time       300
Initialization time   20
Active sources        [1 0 0 1]
TCP Windows           [165 5000 5000]
Link delay            70ms
Link capacity         10Mb
Link buffer           5

TCP1 Average Throughput = 4.5048010666666665 [Mbps]
     Stable Throughput = 4.7070857142857205 [Mbps]
TCP2 Average Throughput = 0.0 [Mbps]
     Stable Throughput = 0.0 [Mbps]
TCP3 Average Throughput = 0.0 [Mbps]
     Stable Throughput = 0.0 [Mbps]
```

And the plot:



Compared to the last plot, we can see sawtooths and the cwnd doesn't exceed the defined optimal TCP window size (225000 bytes). The cwnd drops when it reaches roughly below above mentioned optimal window size. Note that window size is in segments in the config (and plot title), but in bytes on the plot.

Unfortunately, this isn't a typical cwnd sawtooth type of graph. When the above mentioned maximal window size is reached, the cwnd value should drop by half and then continue to grow up. But the behavior in the plot is not as described. It is due to the fact, that the packet loss is too big after the optimal window size is reached

We can roughly see on the plot that cwnd drops by half, but then it drops by half again, and again… until it reaches very low values. To be precise, it drops by half two times, and the third time cwnd is dropped, it drops into very low value rather by half like previously.

After that it takes some time until cwnd regains proper behavior and starts to grow.

Task 1 - final conclusions

Our calculated window size is not 100% correct. The best window size seems to be around 145. Our calculations are still quite a good estimation for the window size.
It is better to pick some value a bit lower than optimal window size and then increase window size until throughput drops, so we learn the actual best window size, i.e. window size for which the throughput is the highest.
If we take a window size too big, packet loss will occur, reducing the overall throughput. Assuming we know the optimal window size, we can get very good throughput value (good bandwidth usage).

Task 2

In this task, we are going to focus on the influence of the buffer size on the TCP efficiency (% use of throughput). We are going to show the throughput as a function of the buffer size. Furthermore, we will show following plots for lowest buffer size and for the most optimal buffer size found:

- cwnd(time) graph
- rtt(time) graph
- TCP momentary throughput(time)
    - Throughput for above being calculated for 1 s periods
- TCP momentary throughput estimation using cwnd and rtt

Preset values

From task description:

- R1-R2 link delay: 70ms,
- access links delay: 10ms,
- R1-R2 link capacity: 10 Mbps,
- max. TCP window size: 5000 packets.

Only TCP1 connection is enabled. All other values are as default.

Theoretical background - throughput for 1s intervals

cwnd(time) and rtt(time) graphs are rather trivial. The former has been shown already. The latter has the same difficulty to plot.

Regarding the TCP momentary throughput(time) graph, that is calculated for every 1s period, we came up with the following method of doing it.
Throughput is how many bytes are travelling in a given time. Since it should be measured for 1s periods, it simply means we need to calculate Bytes/s amount for every second of the simulation. Since probing intervals are accurate, we simply take into account 100 records in the out.csv file for every second. I.e. 100 records in out.csv is 1 second, because probing time is 0.01 s. 0.01 s * 100 = 1s

Then, we will use the bytes column that shows the total amount of bytes transferred. Since it shows a TOTAL amount of bytes transferred, we will need to calculate the difference between the beginning of our 1 second interval and the end of it.

Theoretical background - estimate throughput

For this subpoint, the formula is in the EINTE presentation about TCP (05):

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

To use this formula, we will also need to take 1 second intervals and calculate mean cwnd and mean RTT for this 1 second interval. Then we divide these two mean values and we get the momentary throughput for 1 second interval.
Both throughput calculations can be done rather easily thanks to the Microsoft Excel functions. Excel formulas used to calculate needed values will be shown later.

TCP throughput dependence on buffer size

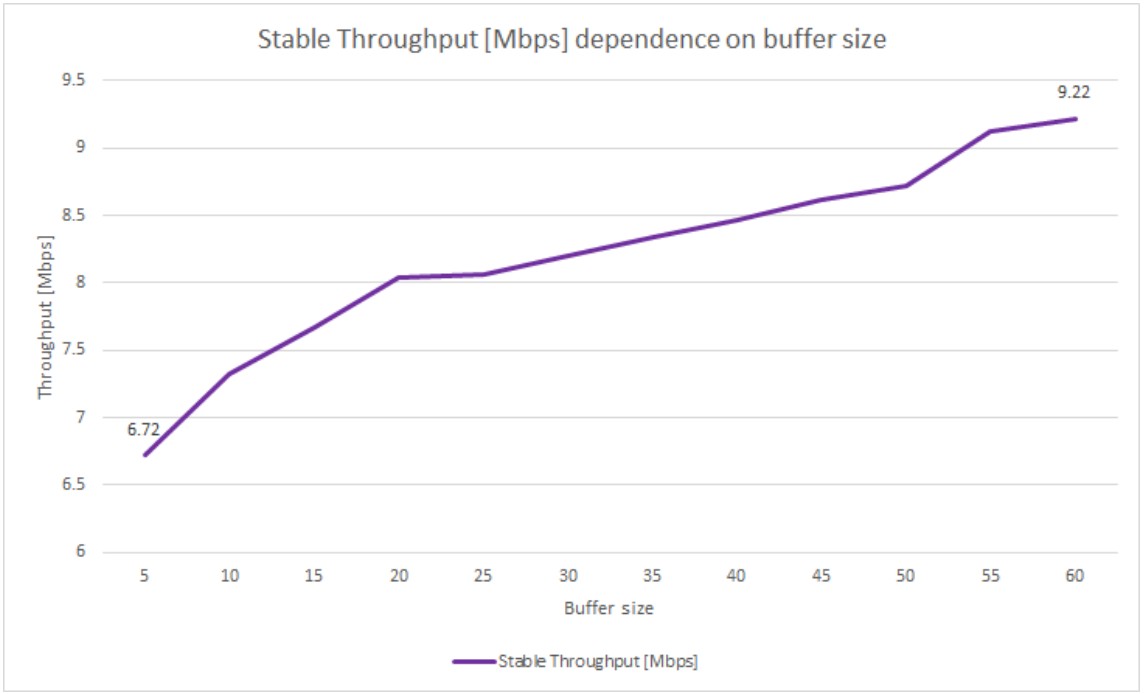
Below we present a plot showing how the stable throughput changes depending on the chosen buffer size. We start from buf size = 5, and then increase by 5.
i.e. for buf size being 5, 10, 15, … until we reach about 90% bandwidth efficiency (stable throughput being >9 Mbps). As it is described in the task description - “few percent accuracy is enough”.
First, we will include the console outputs for buf size 5:

```
Simulation time      300
Initialization time  20
Active sources       [1 0 0 1]
TCP Windows          [5000 5000 5000]
Link delay           70ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 6.3860810666666668 [Mbps]
      Stable Throughput = 6.7227428571428653 [Mbps]
```

And here is the console output for the biggest buf size that resulted in throughput being slightly bigger than 9.00 Mbps. It is for buf size = 60:

```
Simulation time      300
Initialization time  20
Active sources       [1 0 0 1]
TCP Windows          [5000 5000 5000]
Link delay           70ms
Link capacity        10Mb
Link buffer          60

TCP1 Average Throughput = 8.9598410666666659 [Mbps]
      Stable Throughput = 9.2194714285714401 [Mbps]
```

In this case, the bandwidth usage is about 92.2% of the max bandwidth. For bigger buffer sizes, the increase in throughput was not high, so we decided to settle for buf size = 60 as the optimal buf size.
This is the plot showing throughput change for different buffer sizes:



Plot realisation in excel - background

Below we show used formulas in excel that were used to obtain all the plots specified in the task description. Hardest part was to find a general formula for 1s intervals and then address the proper cells for calculations:

cwnd upper bound addr	cwnd lower bound addr	rtt upper bound addr	rtt lower bound addr	bytes upper bound addr	bytes lower bound addr	upper row	lower row
B2	B102	C2	C102	D2	D102	2	102
B103	B202	C103	C202	D103	D202	103	202
B203	B302	C203	C302	D203	D302	203	302

Upper row and lower row bounds for every second were chosen manually, then extended with autofill excel function. The lower and upper bound cell addresses are created with ADDRESS(row_num, col_num). Row_num is chosen from the shown upper row and lower row data column. col_num is chosen based on in which column is the cwnd, rtt or bytes data

Simulation throughput is the value of total bytes transferred at the lower bound minus value of total bytes transferred at the upper bound. e.g.:

=INDIRECT(O2) - INDIRECT(N2)

	E	F	G	H	I	J	K	L	M	N	O
;1			second	throughput	theoretical throughput	cwnd upper bound addr	cwnd lower bound addr	rtt upper bound addr	rtt lower bound addr	bytes upper bound addr	bytes lower bound addr
0			1	INDIRECT(N2)	123113.7102	B2	B102	C2	C102	D2	D102

INDIRECT is to take the address shown in the N and O column rather than the text value. INDIRECT interprets this as an address, not “some string”.

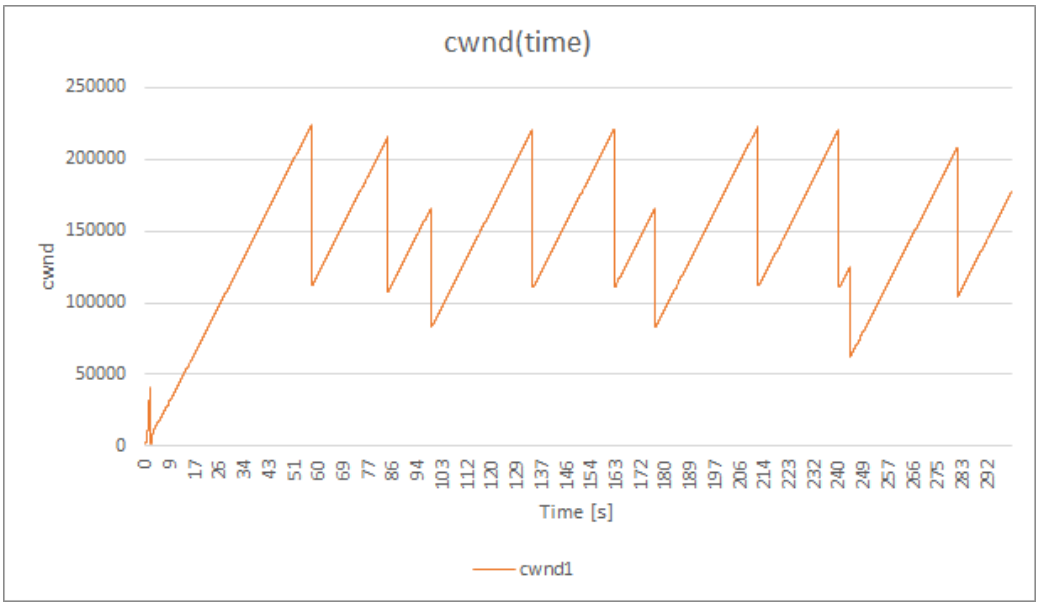
=AVERAGE(INDIRECT(J2):INDIRECT(K2)) / AVERAGE(INDIRECT(L2):INDIRECT(M2))

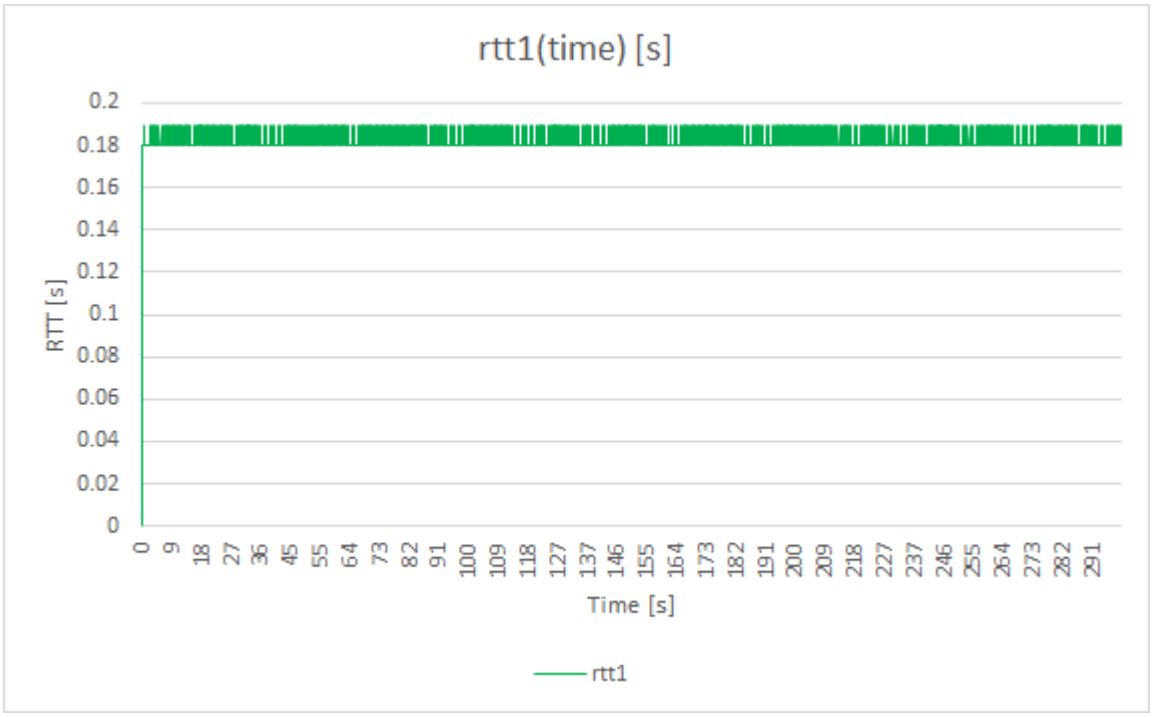
	E	F	G	H	I	J	K	L	M
;1			second	throughput	theoretical throughput	cwnd upper bound addr	cwnd lower bound addr	rtt upper bound addr	rtt lower bound addr
0			1	80040	INDIRECT(M2))	B2	B102	C2	C102

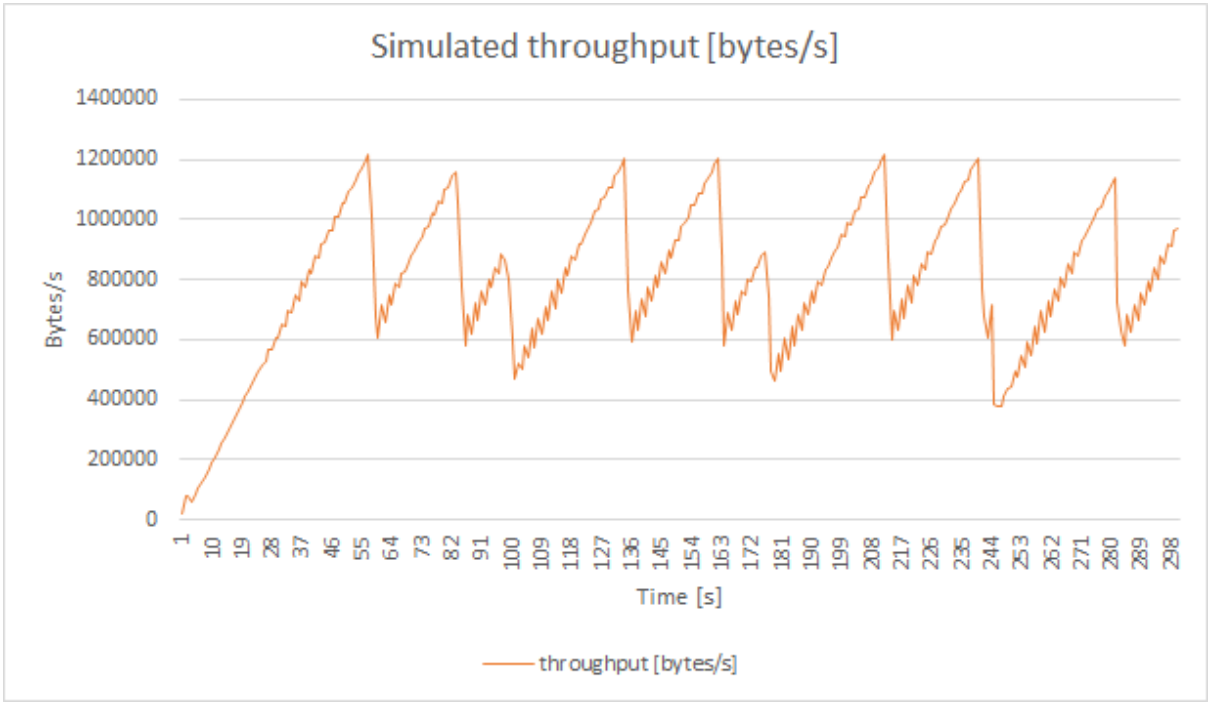
Then theoretical throughput is the average of cwnd in the 1s time period, divided by the average rtt in the same 1s period. It is the same formula as discussed earlier. Furthermore, we introduced two additional columns to add two plots showing throughputs, but in bits rather than in bytes (main plots are in bytes).

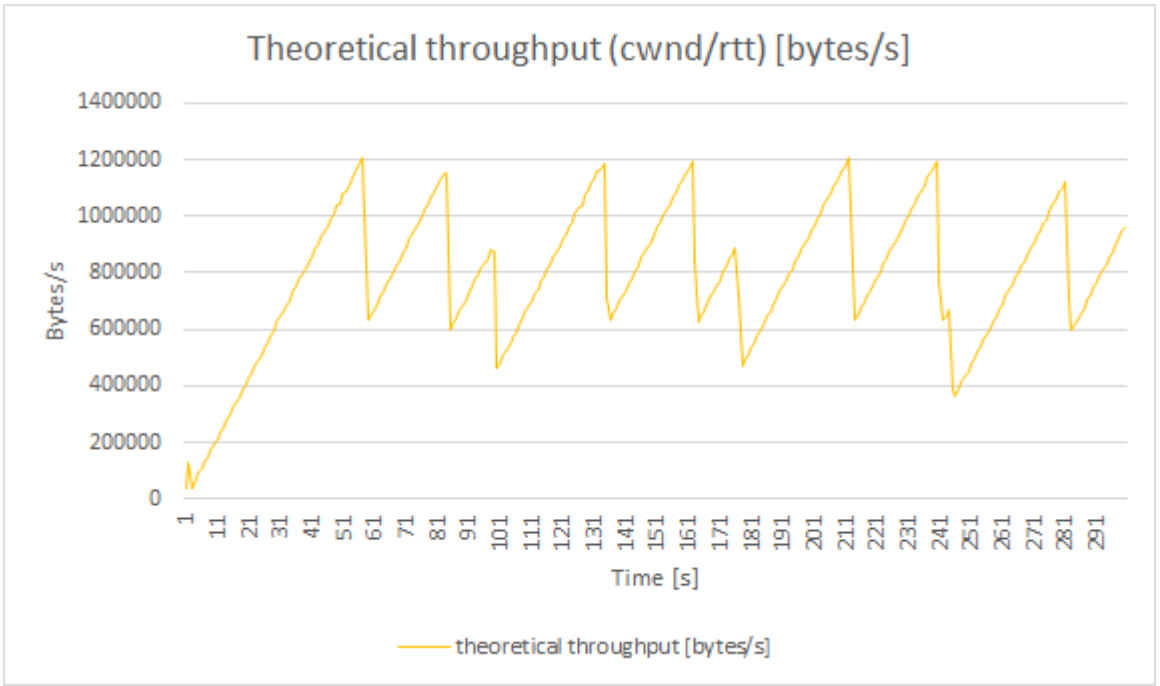
Plots for buffer size = 5

Below are shown all plots required by the task. For *buffer size = 5*:



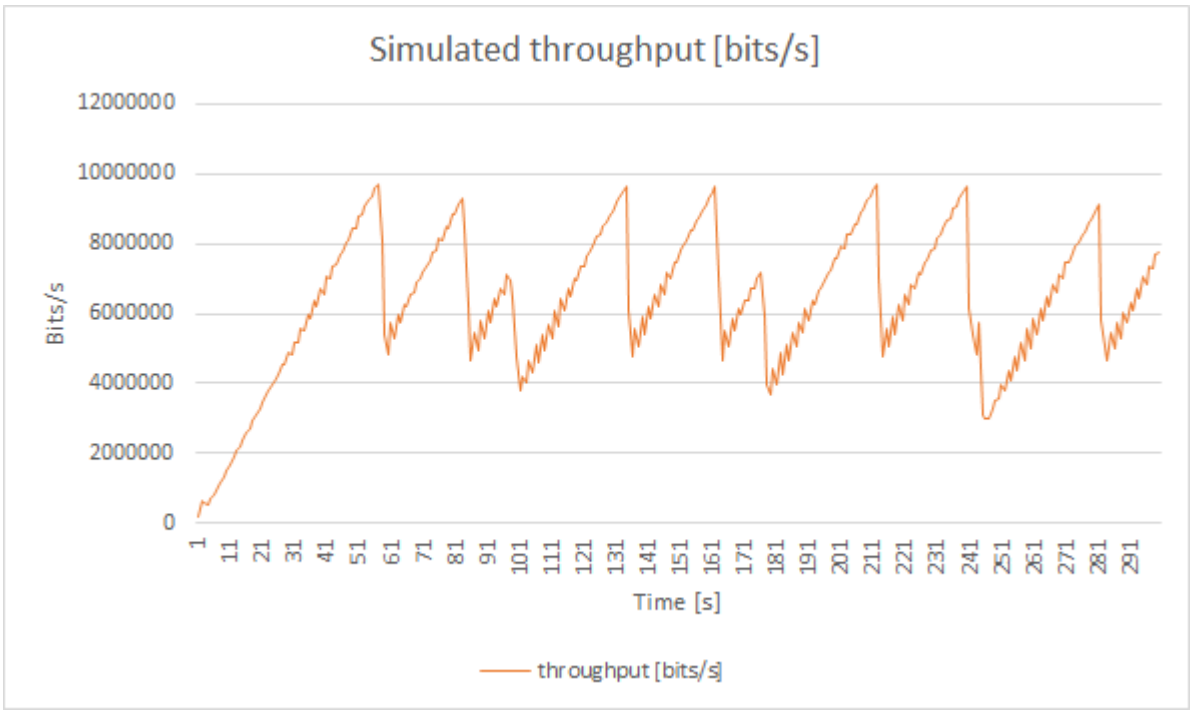


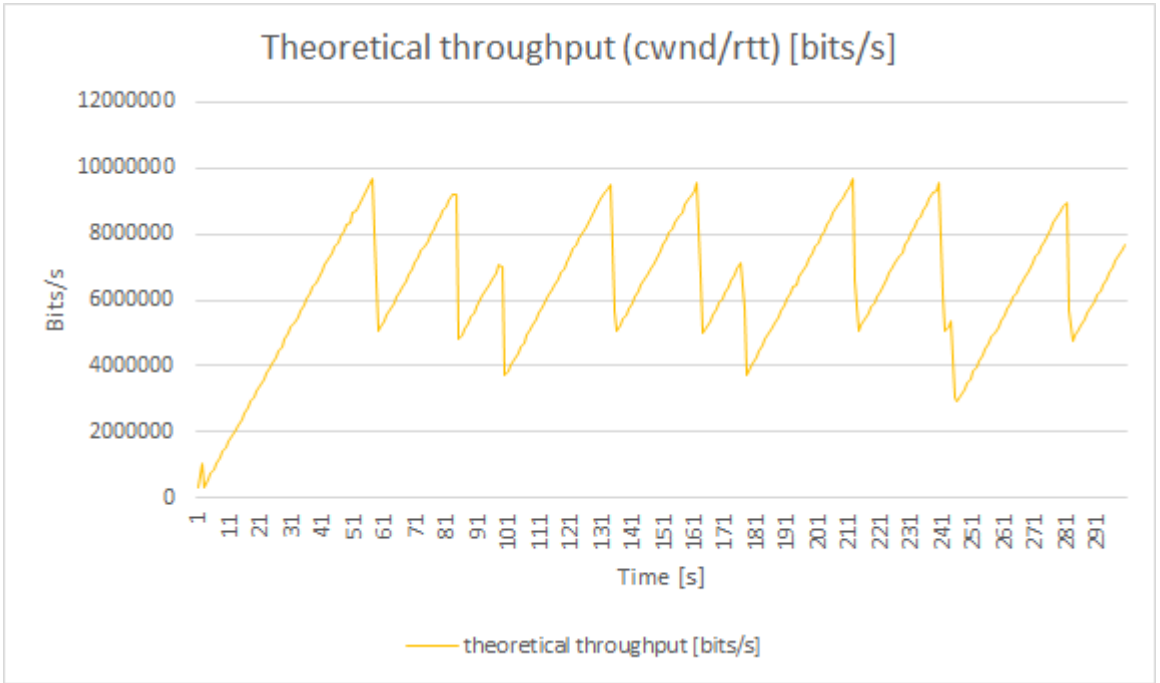

Simulated throughput [bytes/s]
Bytes/s
1400000
1200000
1000000
800000
600000
400000
200000
0
1
10
19
28
37
46
55
64
73
82
91
100
109
118
127
136
145
154
163
172
181
190
199
208
217
226
235
244
253
262
271
280
289
298
Time [s]
throughput [bytes/s]


Theoretical throughput (cwnd/rtt) [bytes/s]
Bytes/s
1400000
1200000
1000000
800000
600000
400000
200000
0
1
11
21
31
41
51
61
71
81
91
101
111
121
131
141
151
161
171
181
191
201
211
221
231
241
251
261
271
281
291
Time [s]
theoretical throughput [bytes/s]

Plots for buffer size = 5 - appendix

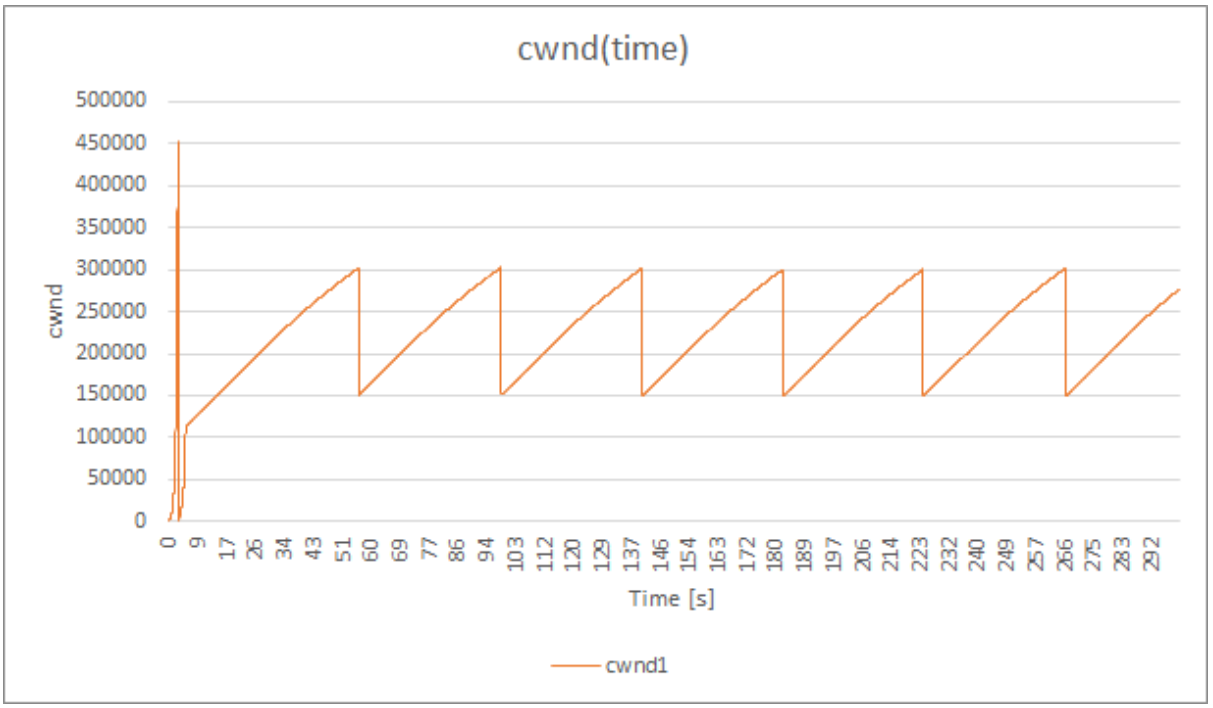
As an appendix, we attach plots for simulated and theoretical throughput in bits/s:

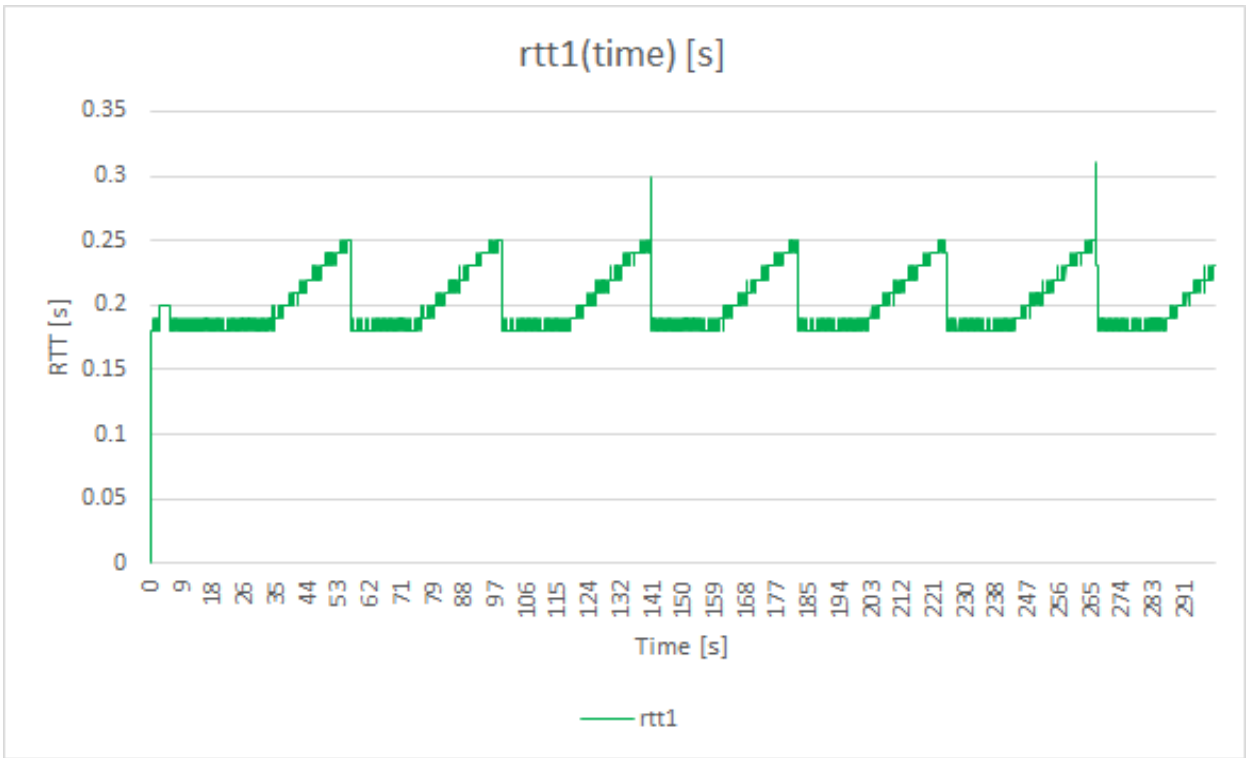


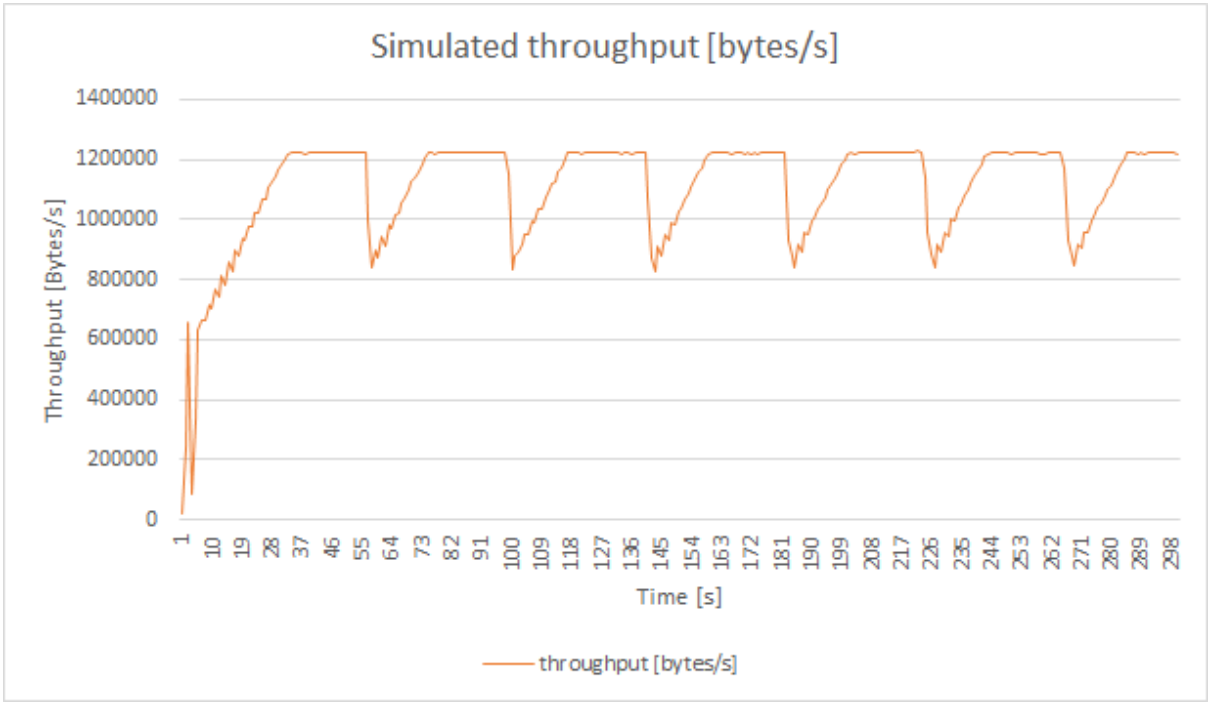


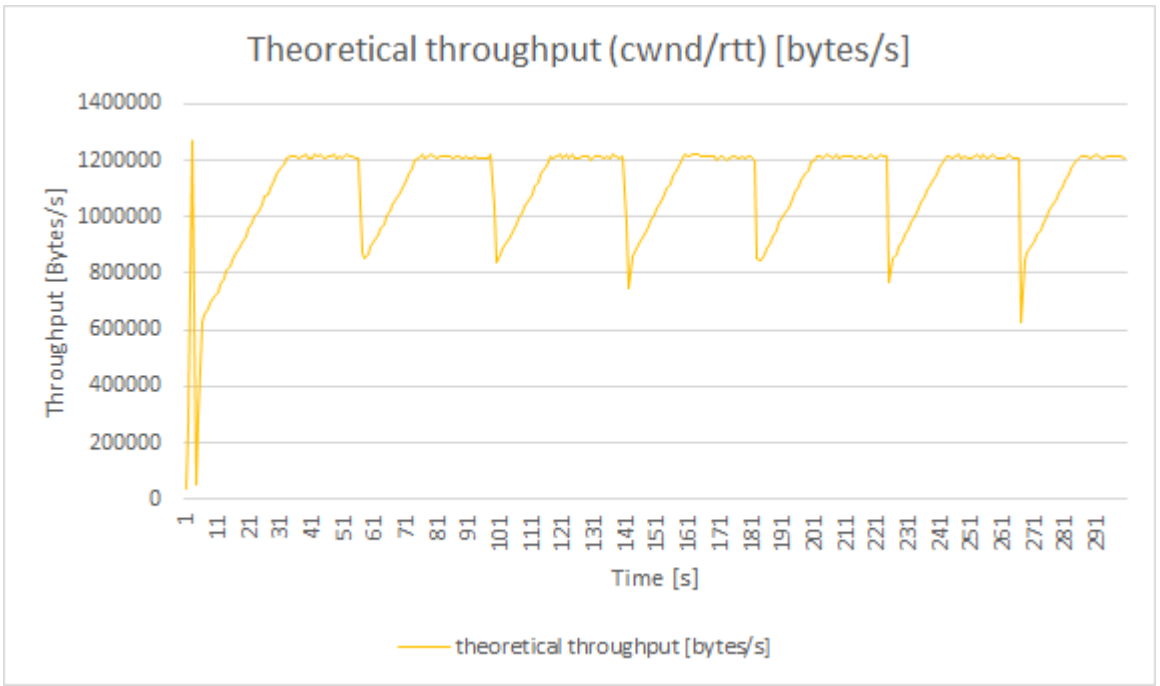
Plots for buffer size = 60

These plots were done just as above mentioned plots, but for different buf size data:



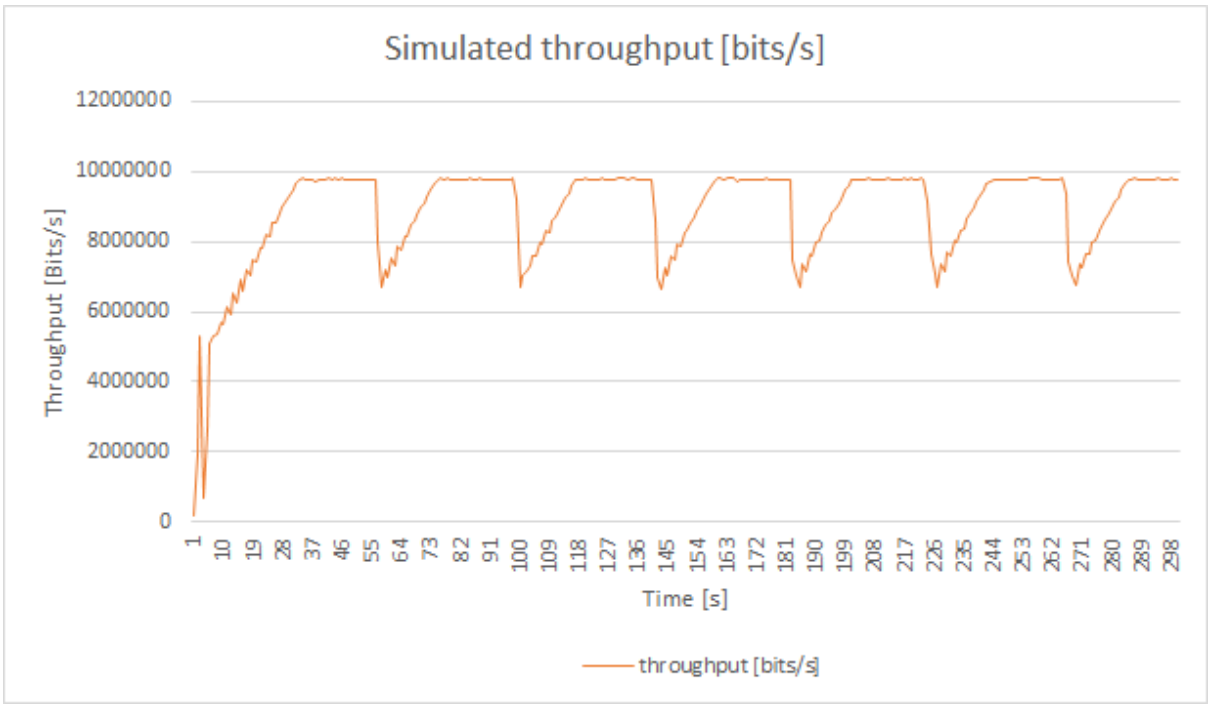


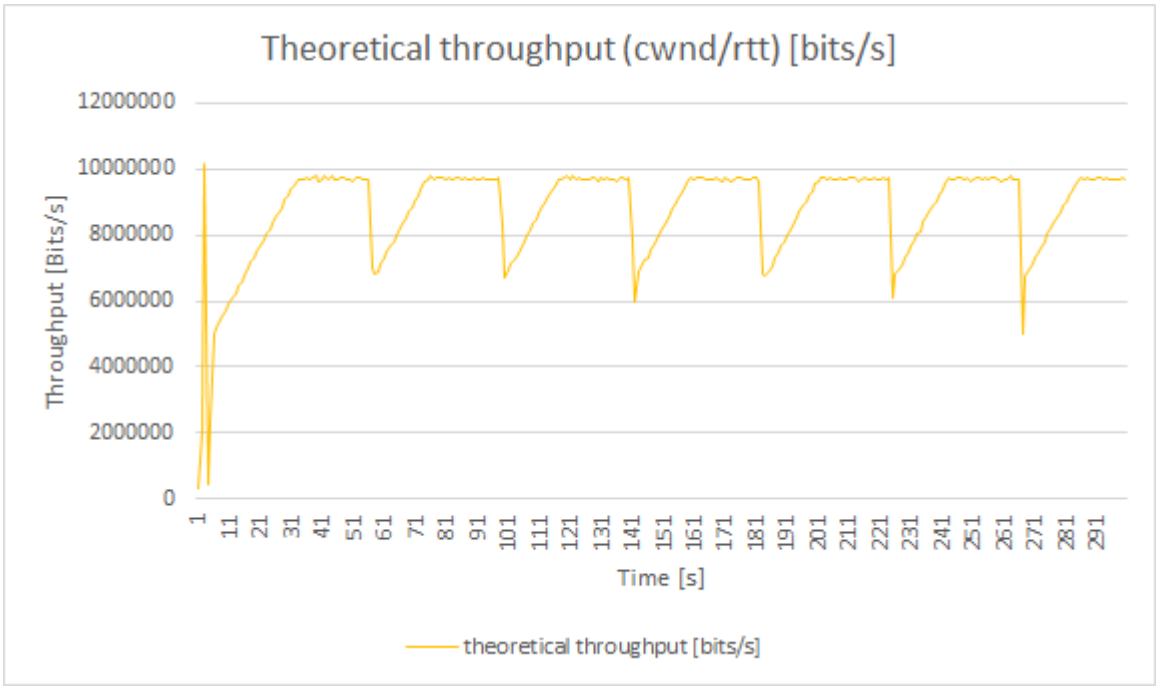

Simulated throughput [bytes/s]
Throughput [Bytes/s]
1400000
1200000
1000000
800000
600000
400000
200000
0
1
10
19
28
37
46
55
64
73
82
91
100
109
118
127
136
145
154
163
172
181
190
199
208
217
226
235
244
253
262
271
280
289
298
Time [s]
throughput [bytes/s]


Theoretical throughput (cwnd/rtt) [bytes/s]
Throughput [Bytes/s]
1400000
1200000
1000000
800000
600000
400000
200000
0
1
11
21
31
41
51
61
71
81
91
101
111
121
131
141
151
161
171
181
191
201
211
221
231
241
251
261
271
281
291
Time [s]
theoretical throughput [bytes/s]

Plots for buffer size = 60 - appendix

As an appendix, we attach plots for simulated and theoretical throughput in bits/s:





Plot analysis

For both cases theoretical throughput is roughly the same as corresponding simulated throughput. Note that for buffer size = 5, the throughput has the same shape as cwnd(time) plot. Furthermore, RTT is very stable for buffer size = 5.

For buffer size = 60, theoretical throughput reaches its maximum values at the same time as window size (cwnd) reaches roughly below our theoretical best value, which was *225000 [bytes].* The throughput is kept at above-mentioned max value until a packet is lost due to cwnd being increased (algorithm probing for bigger cwnd).

Task 2 - conclusions & further plot analysis

As it can be seen from the plots, choosing the best buffer size results in more stable throughput, which also results in bigger throughput overall. This can be seen from the fact that maximum cwnd value is higher for buf size = 60. We obtain a perfect sawtooth in the cwnd(time) graph for buf size = 60, whereas for buf size = 5 the cwnd(time) graph is more unstable. There are more congestion avoidance and fast recovery phases, and the cwnd sometimes drops to lower values than half of previous max cwnd in a short amount of time (after dropping by half, it drops further).

The lack of stability can be easily seen when comparing cwnd(time) plots for different buffer sizes (5 and 60). For buf size = 60, the line is more consistent; cwnd is regained more consistently and usually faster.

It is important to try finding the proper size of the buffer in order to both increase the throughput (bandwidth % efficiency), and to increase the stability of the packet flow.

Possible buf size estimation - theoretical

In theory, we want our buffer size to be equally filled with INCOMING ACKs and OUTGOING packets. This perfect scenario means that our buf size should be half the size of max window size. Since our found max window size in task 1 was 145 segments, then our best buf size would be roughly 70, which is relatively close to our obtained buf size.
This estimation seems legitimate, especially if we take into consideration the fact that we stopped at 9.22 Mbps, which isn't full link bandwidth. If we would assume the best buf size to be 70, then we would probably get even better throughput.

Task 3

In this task we will analyze how the bandwidth is shared in case of 3 simultaneous TCP connections over the same link. TCP connections will “compete” over the bandwidth that will be divided between these 3 different TCP connections being made at the same time.

Theoretical background - task 3

In the EINTE TCP presentation we got the following definition regarding Fair Share:

- How we define the TCP „fair share” (FS)?
  - assume we have link of capacity C serving N TCP connections and some UDP traffic

$$FS_{link} = \frac{C - UDP\ traffic}{N_{TCP}} \qquad Net.capacity = \min_{link} FS_{link}$$

UDP traffic doesn’t exist in our simulation. C - link capacity is set to 10 Mb/s. N is 3
So in accordance with the formula, *FS = 10 * ⅓ Mb/s = 3.(3) Mb/s*
We will check if the following formula holds by running simulations for the cases described in the task description.

Simulation types

We will need to run two types of simulations:

1. Equal RTT for every TCP link
2. Different RTTs for TCP links.

The 2nd point is possible thanks to the access links delay modification. We will set different delays for every link and see how that influences throughput.

Equal RTT for every TCP link - background & simulation results

Here are the results for a situation where RTT is equal for every link.
Per task, do at least 5 different simulations for RTT ranging from 10 ms to 200.
Reminding that RTT formula is:
*RTT = 2 * (R1-R2 delay + access link delay1 + access link delay2)*
First, we attach screenshots from the terminal:
RTT 10 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           3ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 3.1086810666666671 [Mbps]
     Stable Throughput = 3.1310142857142895 [Mbps]
TCP2 Average Throughput = 3.1207610666666667 [Mbps]
     Stable Throughput = 3.1095428571428609 [Mbps]
TCP3 Average Throughput = 3.1392010666666668 [Mbps]
     Stable Throughput = 3.1288714285714323 [Mbps]
```

RTT 50 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           5ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.861721066666667 [Mbps]
      Stable Throughput = 2.9181428571428607 [Mbps]
TCP2 Average Throughput = 2.7265210666666668 [Mbps]
      Stable Throughput = 2.7106714285714317 [Mbps]
TCP3 Average Throughput = 2.9699210666666667 [Mbps]
      Stable Throughput = 2.9543142857142892 [Mbps]
```

RTT 60 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           10ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.7929210666666671 [Mbps]
      Stable Throughput = 2.8298571428571466 [Mbps]
TCP2 Average Throughput = 2.6712010666666668 [Mbps]
      Stable Throughput = 2.6513571428571461 [Mbps]
TCP3 Average Throughput = 3.0325210666666669 [Mbps]
      Stable Throughput = 3.0409285714285752 [Mbps]
```

RTT 80 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           20ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.7733610666666668 [Mbps]
      Stable Throughput = 2.8455857142857179 [Mbps]
TCP2 Average Throughput = 2.8802010666666669 [Mbps]
      Stable Throughput = 2.8712571428571465 [Mbps]
TCP3 Average Throughput = 2.6002010666666671 [Mbps]
      Stable Throughput = 2.5935857142857173 [Mbps]
```

RTT 120 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           40ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.5994810666666668 [Mbps]
      Stable Throughput = 2.5986000000000034 [Mbps]
TCP2 Average Throughput = 2.6782010666666669 [Mbps]
      Stable Throughput = 2.7453857142857179 [Mbps]
TCP3 Average Throughput = 2.5981210666666668 [Mbps]
      Stable Throughput = 2.6553857142857171 [Mbps]
```

RTT 160 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           60ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.2575610666666668 [Mbps]
      Stable Throughput = 2.2764428571428601 [Mbps]
TCP2 Average Throughput = 2.3538410666666669 [Mbps]
      Stable Throughput = 2.3789571428571459 [Mbps]
TCP3 Average Throughput = 2.3254010666666667 [Mbps]
      Stable Throughput = 2.4048428571428602 [Mbps]
```

And finally, RTT 200 ms:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           80ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.436481066666667 [Mbps]
      Stable Throughput = 2.5335000000000032 [Mbps]
TCP2 Average Throughput = 2.2567610666666669 [Mbps]
      Stable Throughput = 2.3359714285714315 [Mbps]
TCP3 Average Throughput = 1.9957210666666667 [Mbps]
      Stable Throughput = 2.0647714285714311 [Mbps]
```

Here is a table aggregating obtained results.
Bandwidth usage is also included, which is a sum of throughputs from every link divided by maximum throughput (10 Mbps). The value is shown as % of usage:

RTT [ms]	Stable Throughput @ TCP1 [Mb/s]	Stable Throughput @ TCP2 [Mb/s]	Stable Throughput @ TCP3 [Mb/s]	R1-R2 delay [ms]	Access link delay 1 [ms]	Access link delay 2 [ms]	Bandwidth utilisation
10	3.13	3.12	3.13	3	1	1	93.80%
50	2.92	2.71	2.95	5	10	10	85.80%
60	2.83	2.65	3.04	10			85.20%
80	2.85	2.87	2.59	20			83.10%
120	2.6	2.75	2.66	40			80.10%
160	2.28	2.38	2.4	60			70.60%
200	2.53	2.34	2.06	80			69.30%

Equal RTT for every TCP link - table analysis

Overall, throughput for every different TCP connection lowers as we increase RTT. Furthermore, the bandwidth % utilisation decreases as we increase RTT. Theoretically every throughput should be at around 3.(3) Mb/s, but as we can see from the table, this is not the case. For the 10 ms case it is more or less correct, but not for every other RTT case.

Additional result analysis

We noticed one thing when looking into the config file. There is an option to enable unbeknownst to us UDP connection, that is shown as the 4th source in the active source simulation output. Compare this console screenshot to the one for RTT 200 ms. The only difference is that now we disabled UDP connection:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 0]
TCP Windows          [5000 5000 5000]
Link delay           60ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 2.7912010666666669 [Mbps]
      Stable Throughput = 2.8401857142857176 [Mbps]
TCP2 Average Throughput = 2.2247610666666668 [Mbps]
      Stable Throughput = 2.2600714285714316 [Mbps]
TCP3 Average Throughput = 3.244721066666667 [Mbps]
      Stable Throughput = 3.3698142857142894 [Mbps]
```

Calculating throughput usage from this, we obtain % usage to be about:
*[(2.84 + 2.26 + 3.37) / 10] * 100% =* ***84.7%***
Comparing obtained value to the one shown in the plot, we see a difference. Turning off UDP we got an increase in the throughput utilisation by 15.4 percentage points.
For RTT = 10 ms case, without UDP:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 0]
TCP Windows          [5000 5000 5000]
Link delay           3ms
Link capacity        10Mb
Link buffer          5

TCP1 Average Throughput = 3.2940410666666669 [Mbps]
      Stable Throughput = 3.2930142857142894 [Mbps]
TCP2 Average Throughput = 3.0561610666666668 [Mbps]
      Stable Throughput = 3.0551142857142897 [Mbps]
TCP3 Average Throughput = 3.2932410666666669 [Mbps]
      Stable Throughput = 3.2964000000000042 [Mbps]
```

The % usage is about:
*[(3.29 + 3.06 + 3.30) / 10] * 100% =* ***96.5%.*** 2.7 percentage points difference, which is way lower than the above-mentioned case. This still proves that there exists some UDP connection that is unaccounted for in the table.

Presence of the UDP connection most likely explains low utilisation of 69.3% calculated beforehand in the table for highest RTT = 200ms. We did not account for the UDP connection when running the simulations that are analyzed in the table.

Since in the task description there is no mention about UDP connection, we will deem the provided table as satisfactory in the sense of task description.

Varying RTT for TCP links - background & simulation results

Here are the results showing throughput for TCP links in a situation where the RTTs for different TCP links are not the same. We have chosen situations with different links having different RTT relations with each other.
There are a few possible relations that can be simulated (equal RTT omitted):

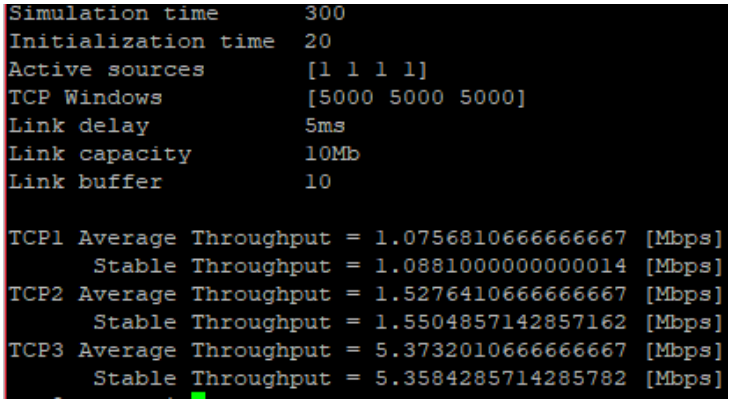
1. Two TCP connections have equal RTT, 3rd has smaller RTT
2. Two TCP connections have equal RTT, 3rd has bigger RTT
3. Every TCP connection has different RTT

Note that the console does not output access link delay, hence for this task the configuration for access link delay will be shown in the excel table. Note that for one TCP connection there are two access link delays. In our realisation, they are equal to each other in value. Furthermore, R1-R2 link delay is 5 ms for every case.

Another note: after running a few simulations for buffer size = 5, we noticed the bandwidth use was really bad in some cases. Usually around 50%. To increase bandwidth usage % we increased the buffer size to 10. This will also allow us to better show the dependencies between throughputs for different TCP links.
Here is the configuration for every case we will consider:

no.	Access link delay 1 & 2 @ TCP1 [ms]	Access link delay 1 & 2 @ TCP2 [ms]	Access link delay 1 & 2 @ TCP3 [ms]	link delay [ms]
1	22	10	5	
2	8	30	15	
3	10	3	27	
4	15	15	4	
5	8	20	20	5
6	20	8	20	
7	8	8	20	
8	22	12	12	
9	7	21	7	

And below I show simulation results (console outputs). For case 1:



For case 2:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           5ms
Link capacity        10Mb
Link buffer          10

TCP1 Average Throughput = 4.8237610666666662 [Mbps]
      Stable Throughput = 4.8615000000000057 [Mbps]
TCP2 Average Throughput = 1.4338810666666666 [Mbps]
      Stable Throughput = 1.4221285714285732 [Mbps]
TCP3 Average Throughput = 1.7928810666666666 [Mbps]
      Stable Throughput = 1.818257142857145 [Mbps]
```

For case 3:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           5ms
Link capacity        10Mb
Link buffer          10

TCP1 Average Throughput = 2.8380810666666667 [Mbps]
      Stable Throughput = 2.8265571428571463 [Mbps]
TCP2 Average Throughput = 4.4152410666666668 [Mbps]
      Stable Throughput = 4.4566714285714335 [Mbps]
TCP3 Average Throughput = 0.9224410666666667 [Mbps]
      Stable Throughput = 0.92040000000000122 [Mbps]
```

For case 4:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           5ms
Link capacity        10Mb
Link buffer          10

TCP1 Average Throughput = 2.3669610666666667 [Mbps]
      Stable Throughput = 2.4048857142857174 [Mbps]
TCP2 Average Throughput = 2.2288810666666667 [Mbps]
      Stable Throughput = 2.272028571428574 [Mbps]
TCP3 Average Throughput = 4.3134010666666667 [Mbps]
      Stable Throughput = 4.2716571428571486 [Mbps]
```

For case 5:

```
Simulation time      300
Initialization time  20
Active sources       [1 1 1 1]
TCP Windows          [5000 5000 5000]
Link delay           5ms
Link capacity        10Mb
Link buffer          10

TCP1 Average Throughput = 3.583241066666667 [Mbps]
      Stable Throughput = 3.6116571428571476 [Mbps]
TCP2 Average Throughput = 2.5037610666666668 [Mbps]
      Stable Throughput = 2.5202142857142888 [Mbps]
TCP3 Average Throughput = 2.0134010666666668 [Mbps]
      Stable Throughput = 2.0013857142857168 [Mbps]
```

For case 6:

```
Simulation time       300
Initialization time   20
Active sources        [1 1 1 1]
TCP Windows           [5000 5000 5000]
Link delay            5ms
Link capacity         10Mb
Link buffer           10

TCP1 Average Throughput = 2.7520010666666668 [Mbps]
      Stable Throughput = 2.7614571428571466 [Mbps]
TCP2 Average Throughput = 3.2348410666666667 [Mbps]
      Stable Throughput = 3.2532857142857181 [Mbps]
TCP3 Average Throughput = 2.6409210666666669 [Mbps]
      Stable Throughput = 2.6886428571428604 [Mbps]
```

For case 7:

```
Simulation time       300
Initialization time   20
Active sources        [1 1 1 1]
TCP Windows           [5000 5000 5000]
Link delay            5ms
Link capacity         10Mb
Link buffer           10

TCP1 Average Throughput = 3.8598010666666669 [Mbps]
      Stable Throughput = 3.8520000000000048 [Mbps]
TCP2 Average Throughput = 3.962241066666667 [Mbps]
      Stable Throughput = 4.0071428571428616 [Mbps]
TCP3 Average Throughput = 1.1054810666666666 [Mbps]
      Stable Throughput = 1.0903285714285729 [Mbps]
```

For case 8:

```
Simulation time       300
Initialization time   20
Active sources        [1 1 1 1]
TCP Windows           [5000 5000 5000]
Link delay            5ms
Link capacity         10Mb
Link buffer           10

TCP1 Average Throughput = 1.6935610666666667 [Mbps]
      Stable Throughput = 1.7270142857142878 [Mbps]
TCP2 Average Throughput = 2.124121066666667 [Mbps]
      Stable Throughput = 2.1387428571428595 [Mbps]
TCP3 Average Throughput = 3.4727210666666668 [Mbps]
      Stable Throughput = 3.523371428571433 [Mbps]
```

For case 9:

```
Simulation time       300
Initialization time   20
Active sources        [1 1 1 1]
TCP Windows           [5000 5000 5000]
Link delay            5ms
Link capacity         10Mb
Link buffer           10

TCP1 Average Throughput = 3.7032010666666668 [Mbps]
      Stable Throughput = 3.6867000000000045 [Mbps]
TCP2 Average Throughput = 1.7585210666666666 [Mbps]
      Stable Throughput = 1.7486571428571451 [Mbps]
TCP3 Average Throughput = 3.5751210666666671 [Mbps]
      Stable Throughput = 3.608571428571433 [Mbps]
```

\

Below is a table showing all the data from above-mentioned simulations:

no.	RTT @ TCP1 [ms]	RTT @ TCP2 [ms]	RTT @ TCP3 [ms]	Stable Throughput @ TCP1 [Mb/s]	Stable Throughput @ TCP2 [Mb/s]	Stable Throughput @ TCP3 [Mb/s]
1	64	40	24	1.09	1.55	5.36
2	36	80	50	4.86	1.43	1.82
3	40	26	74	2.83	4.46	0.92
4	50	50	28	2.4	2.27	4.27
5	36	60	60	3.61	2.52	2
6	60	36	60	2.76	3.25	2.69
7	36	36	60	3.85	4.01	1.09
8	64	44	44	1.73	2.14	3.52
9	34	62	34	3.69	1.75	3.61

Access link delay 1 & 2 @ TCP1 [ms]	Access link delay 1 & 2 @ TCP2 [ms]	Access link delay 1 & 2 @ TCP3 [ms]	R1-R2 link delay [ms]	Bandwidth usage	RTT choice comment
22	10	2		80.00%	TCP1 highest, TCP3 lowest, TCP2 between
8	30	15		81.10%	TCP2 highest, TCP1 lowest, TCP3 between
10	3	27		82.10%	TCP3 highest, TCP2 lowest, TCP1 between
15	15	4		89.40%	TCP1 == TCP2, TCP3 lower than others
8	20	20	5	81.30%	TCP2 == TCP3, TCP1 lower than others
20	8	20		87.00%	TCP1 == TCP3, TCP2 lower than others
8	8	20		89.50%	TCP1 == TCP2, TCP3 higher than others
22	12	12		73.90%	TCP2 == TCP3, TCP1 higher than others
7	21	7		90.50%	TCP1 == TCP3, TCP2 higher than others

Varying RTT for TCP links - table analysis

For every case except for case number 8, the results clearly show that fair-share is not working as intended. Different RTT makes it so that links with smaller RTT have bigger throughput, and links with high RTT have smaller throughput. Theoretically they should have roughly equal throughputs like in the 1st considered case where RTT was equal for every link.

Bandwidth usage appendix

Note that the behavior for high RTTs in currently considered “different RTT for links” simulations is the same as in the equal RTT for links case. E.g. case 2 from “different RTT for links” simulations, but with access link delays multiplied by 3:

```
# Delays on access links
set delay_z1R1    24ms
set delay_z2R1    90ms
set delay_z3R1    45ms

set delay_R2o1    24ms
set delay_R2o2    90ms
set delay_R2o3    45ms

# Enable/disable UDP app
set enable_udp    1

# Write header to output
```

```
Simulation time       300
Initialization time   20
Active sources        [1 1 1 1]
TCP Windows           [5000 5000 5000]
Link delay            5ms
Link capacity         10Mb
Link buffer           10

TCP1 Average Throughput = 3.1297610666666671 [Mbps]
     Stable Throughput = 3.1797000000000035 [Mbps]
TCP2 Average Throughput = 0.86256106666666665 [Mbps]
     Stable Throughput = 0.85988571428571536 [Mbps]
TCP3 Average Throughput = 1.2911210666666666 [Mbps]
     Stable Throughput = 1.3307142857142873 [Mbps]
```

For this case the bandwidth utilisation % is roughly:

*[(3.18 + 0.86 + 1.33) / 10] * 100% = **53.7%***

This shows that the behavior in “varying RTT for links” simulations is the same as in “equal RTT for every link”. As the RTT increases, bandwidth use % decreases.

Value comparison - equal vs varying RTT

We can see that for a situation where RTT is equal for every link, the throughputs are also roughly equal when compared to each other, even when RTT is increased. The downside is that bandwidth % usage decreases. This is because throughput for every link decreases, but the share is still roughly the same for every link.

Whereas in the situation with different RTT for every link, the throughput on every link is influenced by the RTT value for a given link. If RTT is small then the total throughput used for that link takes a bigger portion of the available throughput, rather than distributing the available throughput equally between all links. **In general - links with higher RTTs tend to get a smaller part of the available bandwidth, compared to links with small RTTs that get a higher portion of the available bandwidth.**

Task 3 - conclusions

As it can be seen from the two types of simulation series, the fair-share works only if the RTT for different links is the same or roughly the same. For this case, every TCP connection gets roughly the same percentage of the available throughput.

Unfortunately, if the RTT differs for every link, the fair-share is unable to provide equal % of bandwidth for every of the links. The link with smaller RTT gets more throughput, whereas links with very high RTT get very low throughput.