Computer Network's Computer Assignment Number 2

Network Design Using NS2 and Python

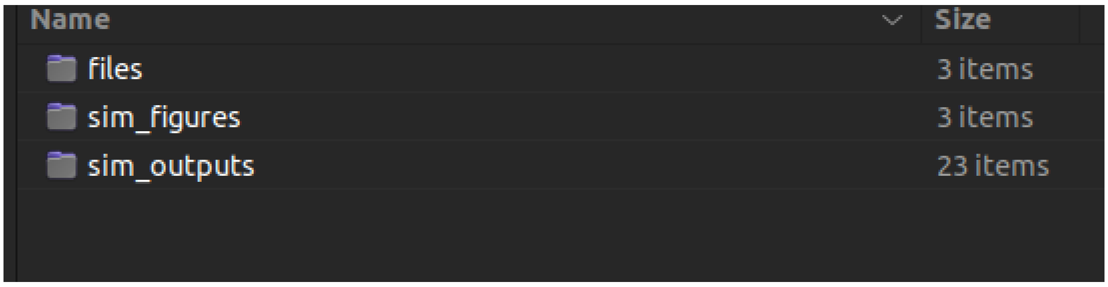
Soroosh Mirzasarvari - Mohammad Farrahi
810198478 - 810198451

Overview:

In this project, we have attempted to design and simulate protocols in a network with a given topology using NS software.

Project Structure:

There are three folders in the uploaded zip. *files* directory contains Python files and the *.tcl* file for running the simulation.
The *sim_figrues* folder contains the images of simulation results that are shown in the report too.
The *sim_output* folder contains files with *.tr* format that are the result of simulations in NS2.



Name	Size
files	3 items
sim_figures	3 items
sim_outputs	23 items

Project Description:

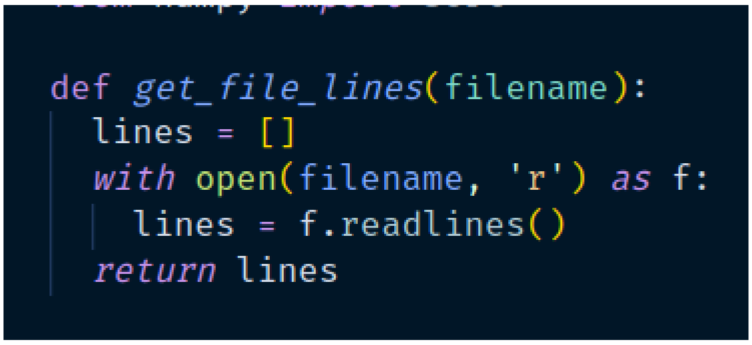
There are three main files in this project located in the *files* directory in the uploaded zip file. This contains:

1. **sim_script.tcl**: this file is the topology of the network expressed in TCL format. To run this file three parameters are needed:
    a. Bandwidth: unit is Mbps and the value will vary between 1.5, 55, and 155 Mbps.
    b. Packet size: unit is the byte.

c. Error value: between 0 and 1. The value in this project is 10^{-5}

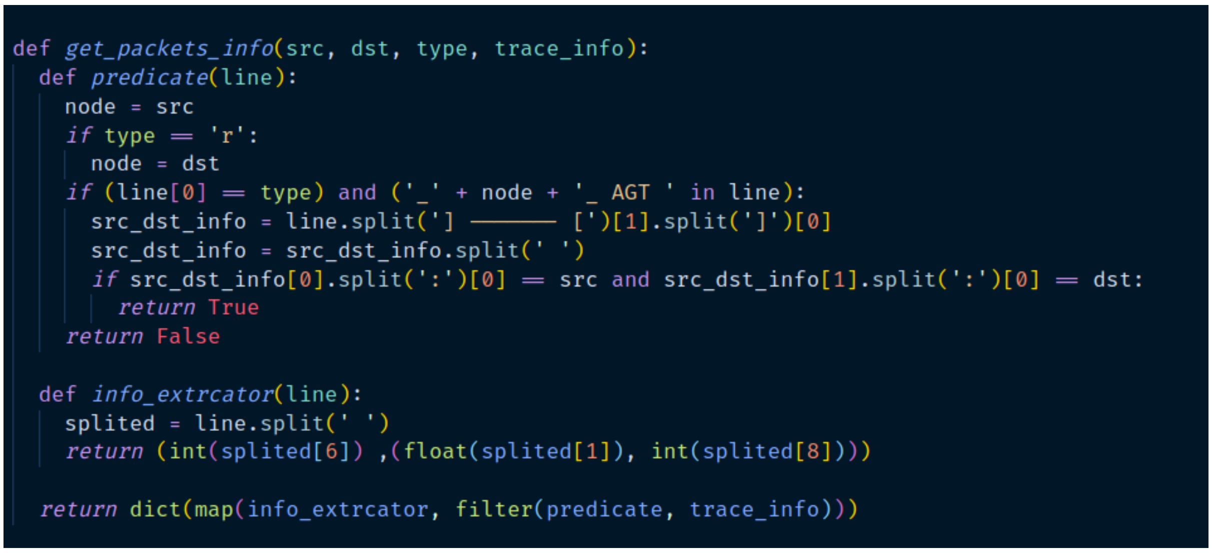
After running this script in NS2, there will be 2 files as output for every variation of the inputs. One of them is in *.tr* format. This file is the file that we parse using Python programming language. The parser is explained later in the document.

2. **parser.py**: this file has the Python code to contain functions to parse the *.tr* output. The file has several functions:
    a. get_file_lines: utility function to open, read and write file's content into a list of strings as lines.



```python
def get_file_lines(filename):
    lines = []
    with open(filename, 'r') as f:
        lines = f.readlines()
    return lines
```

    b. get_packet_info: takes source, destination, packet type, and the trace info of a certain packet and returns a dictionary containing id of the packet, its size and the time that packet has transferred.



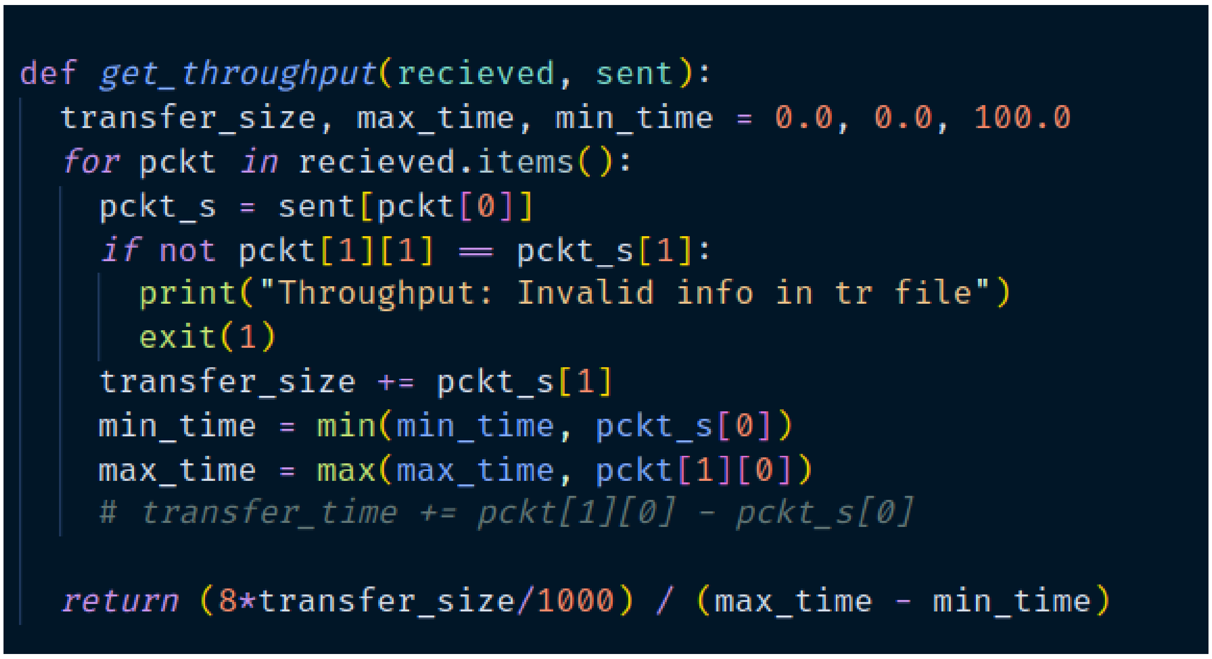
```python
def get_packets_info(src, dst, type, trace_info):
    def predicate(line):
        node = src
        if type == 'r':
            node = dst
        if (line[0] == type) and ('_' + node + '_ AGT ' in line):
            src_dst_info = line.split('] ------- [')[1].split(']')[0]
            src_dst_info = src_dst_info.split(' ')
            if src_dst_info[0].split(':')[0] == src and src_dst_info[1].split(':')[0] == dst:
                return True
        return False

    def info_extrcator(line):
        splited = line.split(' ')
        return (int(splited[6]) ,(float(splited[1]), int(splited[8])))

    return dict(map(info_extrcator, filter(predicate, trace_info)))
```

c. get_throughput: get recieved and sent conections and calculates and returns the throughput in the transfer. The throughput is calculated from below formula:

$$Throughput = \frac{Size}{Time}$$

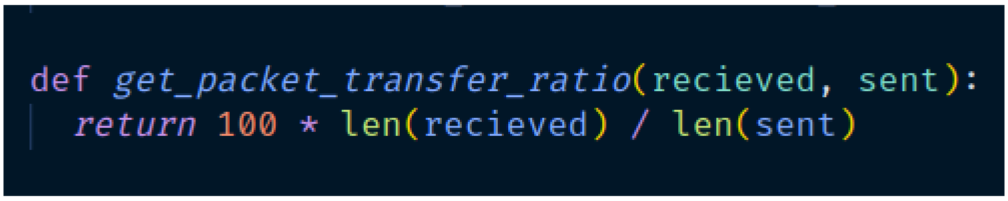


```python
def get_throughput(recieved, sent):
    transfer_size, max_time, min_time = 0.0, 0.0, 100.0
    for pckt in recieved.items():
        pckt_s = sent[pckt[0]]
        if not pckt[1][1] == pckt_s[1]:
            print("Throughput: Invalid info in tr file")
            exit(1)
        transfer_size += pckt_s[1]
        min_time = min(min_time, pckt_s[0])
        max_time = max(max_time, pckt[1][0])
        # transfer_time += pckt[1][0] - pckt_s[0]

    return (8*transfer_size/1000) / (max_time - min_time)
```

d. get_packet_transfer_ratio: calculates transfer ratio besed on the below formula:

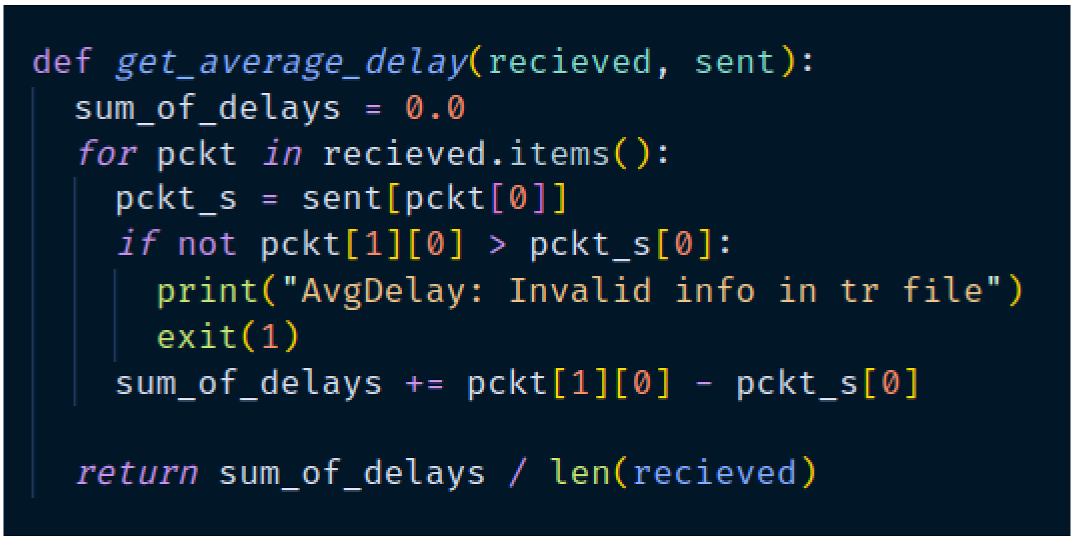
$$Packet\ Transfer\ Ratio = \frac{Packet\ received}{Packet\ Sent}$$



```python
def get_packet_transfer_ratio(recieved, sent):
  return 100 * len(recieved) / len(sent)
```

e. get_avarage_delay: returns the average end-to-end value of the delay between to connections in the system. The formula is calculated based on the formula below:

$$Avg\ E2E\ Delay \; = \frac{\sum receive\ time}{packet\ received}$$

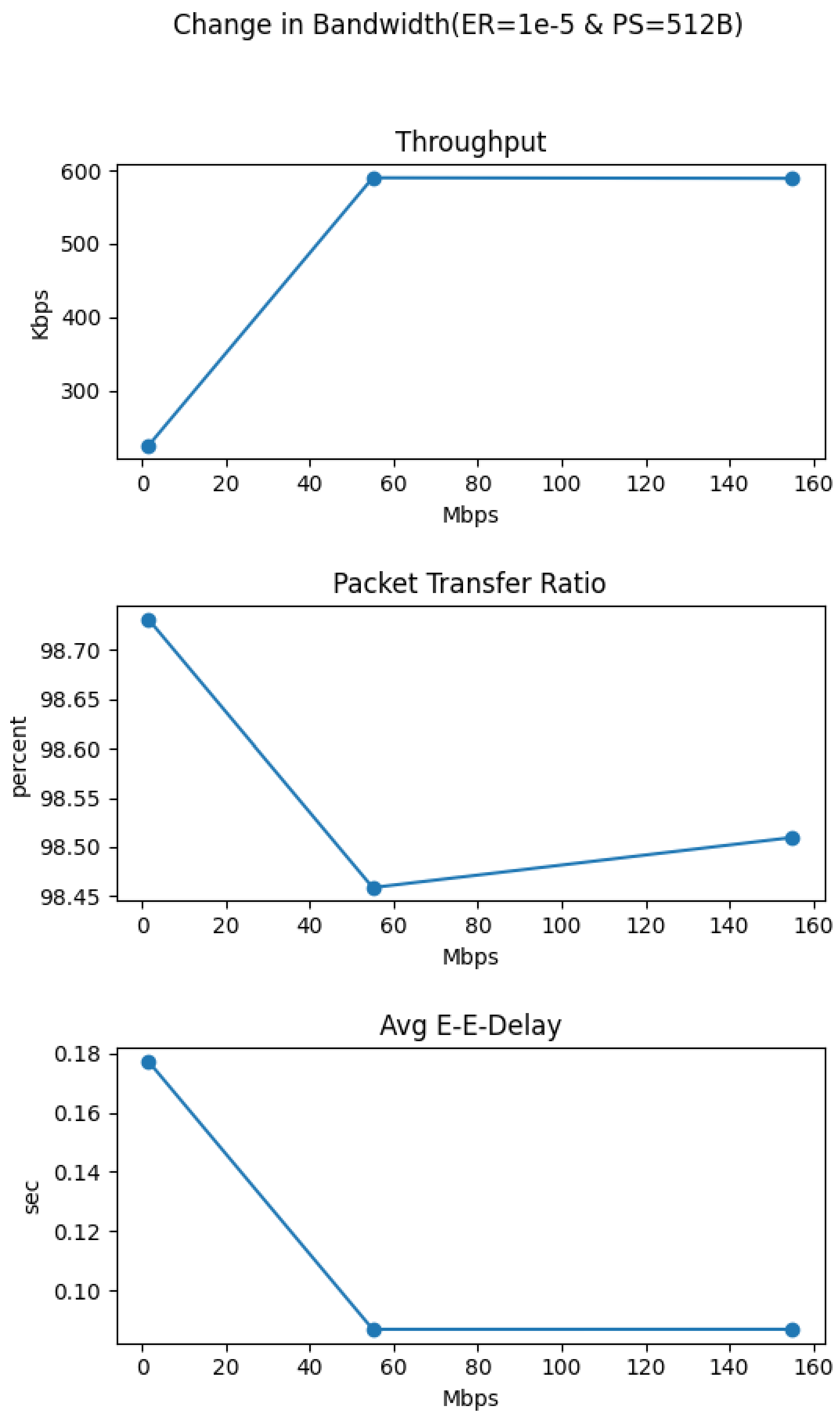


```python
def get_average_delay(recieved, sent):
  sum_of_delays = 0.0
  for pckt in recieved.items():
    pckt_s = sent[pckt[0]]
    if not pckt[1][0] > pckt_s[0]:
      print("AvgDelay: Invalid info in tr file")
      exit(1)
    sum_of_delays += pckt[1][0] - pckt_s[0]

  return sum_of_delays / len(recieved)
```

This file will calculate and shows the three needed values for every said output file as .tr.

3. **runner.py**: In this python file there are 3 scenarios for simulations. This file has the responsibility to first run the simulation and then plot the result. In each scenario one factor is variant and the other two factors are constant.
   a. Variant bandwidth: the value of the bandwidth varies between 1.5, 55, and 155 Mbps.
      The plot shown below will show the simulation result:


Change in Bandwidth(ER=1e-5 & PS=512B)
Throughput
Kbps
600
500
400
300
0
20
40
60
80
100
120
140
160
Mbps
Packet Transfer Ratio
percent
98.70
98.65
98.60
98.55
98.50
98.45
Mbps
Avg E-E-Delay
sec
0.18
0.16
0.14
0.12
0.10
Mbps

This explains a lot about the network:
As the bandwidth expands, the throughput first increases but it stops and remain constant at a point. This happens due to the fact that although throughput is dependent on bandwidth, it also is dependent on packet size.

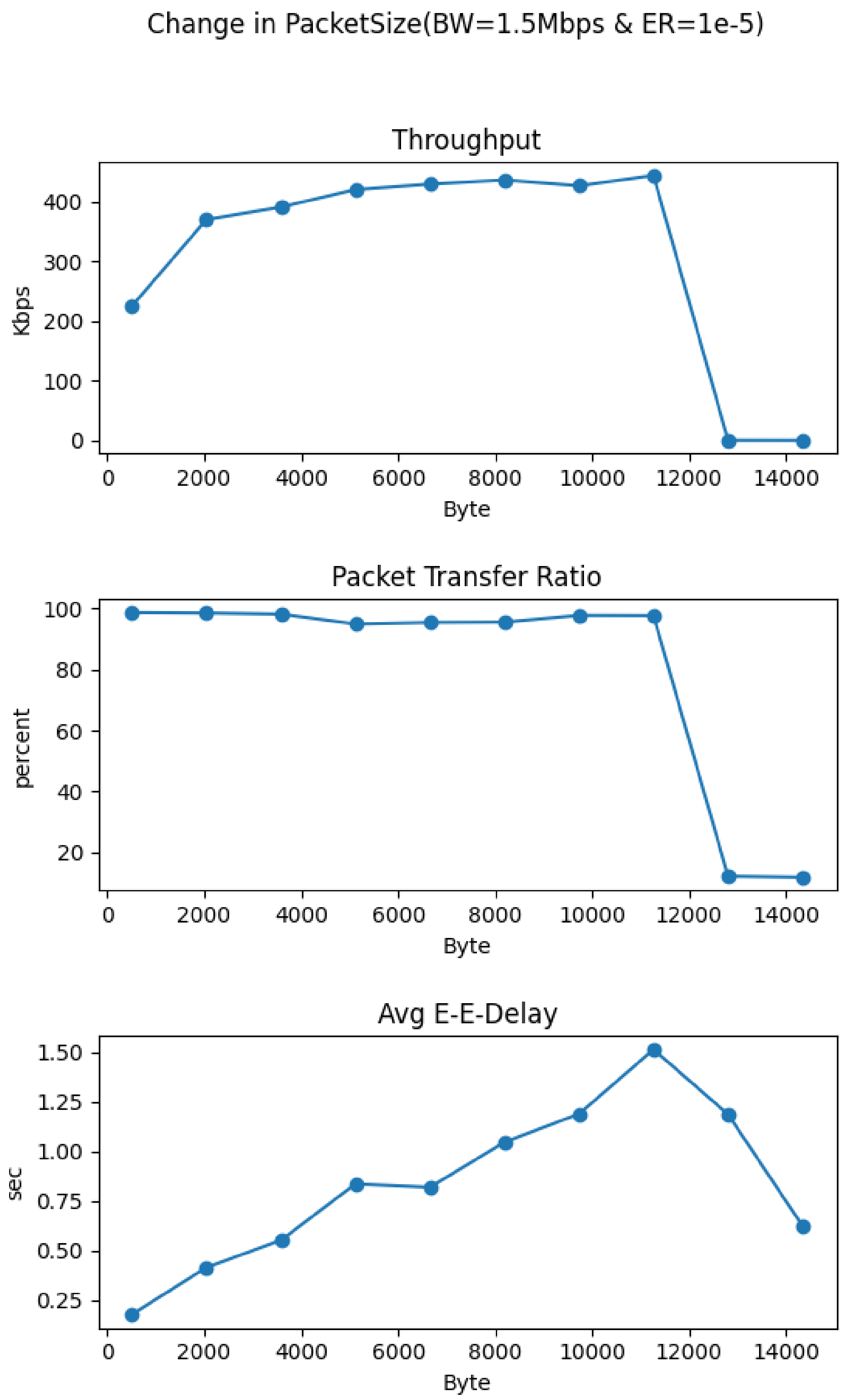
Packet transfer ratio first drops at the value of 50Mbps because the first as we expand the bandwidth, the accuracy of the networks drops.

The average delay also has a reasonable plot as the delay drops when the bandwidth increases but it wont drop from a certain point due to the fact that physical constraints are envolved.

b. Variant packet size: the size starts from 512 and follows the follwoing principle in change rate:

$$packetSize = 512 * i \ (for\ i\ in\ [1, 4, 7, 10, \ldots, 28]$$

The following figure is the result of the said change:

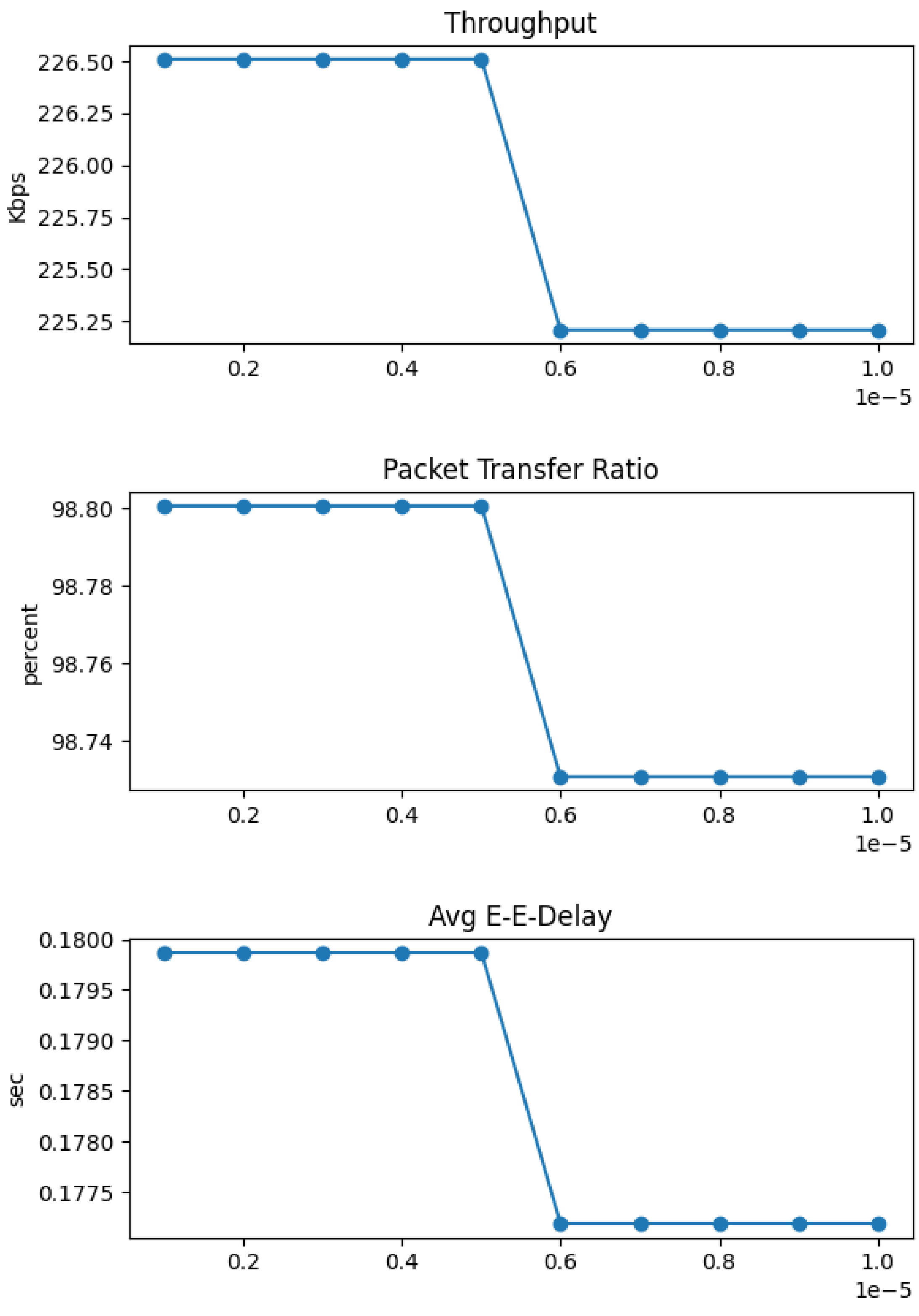

Change in PacketSize(BW=1.5Mbps & ER=1e-5)
Throughput
Kbps
0
100
200
300
400
Byte
0
2000
4000
6000
8000
10000
12000
14000
Packet Transfer Ratio
percent
20
40
60
80
100
Byte
Avg E-E-Delay
sec
0.25
0.50
0.75
1.00
1.25
1.50
Byte

The plots suggest a very interesting point in the graph. In about 12KB as the packet size, we reach to a point that the network cannot handle the size of the packet anymore.

Before that crucial point, the throughput increases. Yet, since the size has enlarged, the average delay is increased too. Also, since the size of the packet is increased, the probability of packet loss increases.

c. Variant error rate: 10 values that range between 10^{-6} to 10^{-5}. The follwogin graph shows the result of the said simulation:

Change in ErrorRate(BW=1.5Mbps & PS=512B)



This set of simulations implies that in about 6×10^{-6} as the value of error, the system suddenly breaks. Resulting in a loss in throughput, and thus, the delay and packet ratio decreases.