CS224m PA2- 160010031

October 1, 2017

```
In [1]: cd /Users/krishna/ns-allinone-3.26/ns-3.26/
```

```
/Users/krishna/ns-allinone-3.26/ns-3.26
```

```
In [2]: %matplotlib inline
        import numpy as np
        import matplotlib as plt
        from pylab import *
        import pandas as pd
```

1 Question 1

Average Throughput(Frome conversations in wireshark) = 4890kbps = 4.89 mbps

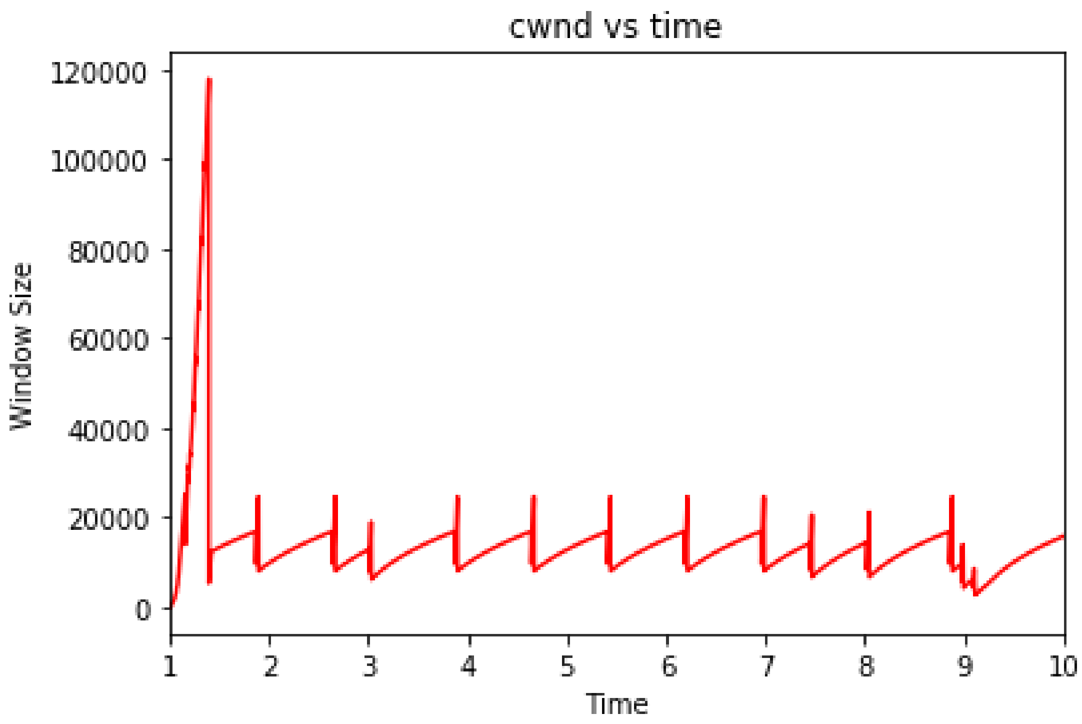
Maximum Expected Throughput= Bandwidth= 5mbps

The acheived throughput is approximately equal to the maximum expected value.

Regarding how many times tcp reduces cwnd, it has been explained below with graph and script.

```
In [12]: # Scipt for plotting graph of cwnd vs time
         tim1, init1, final1= np.loadtxt("tcp-example1.cwnd", unpack=True)
         plt.plot(tim1, init1, 'red')
         plt.xlim([1,10])
         plt.ymin=0
         plt.xlabel("Time")
         plt.ylabel("Window Size")
         plt.title("cwnd vs time")
```

```
Out[12]: <matplotlib.text.Text at 0x11a860f60>
```



```
In [46]: # Script for evaluating number of times cwnd was reduced
         reduce=0
         seq=[]
         for i in range(len(init1)-1):
             if init1[i]>init1[i+1]:
                 reduce+=1
                 seq.append(i)
         print("Number of times cwnd reduced by TCP algorithm =", reduce)
```

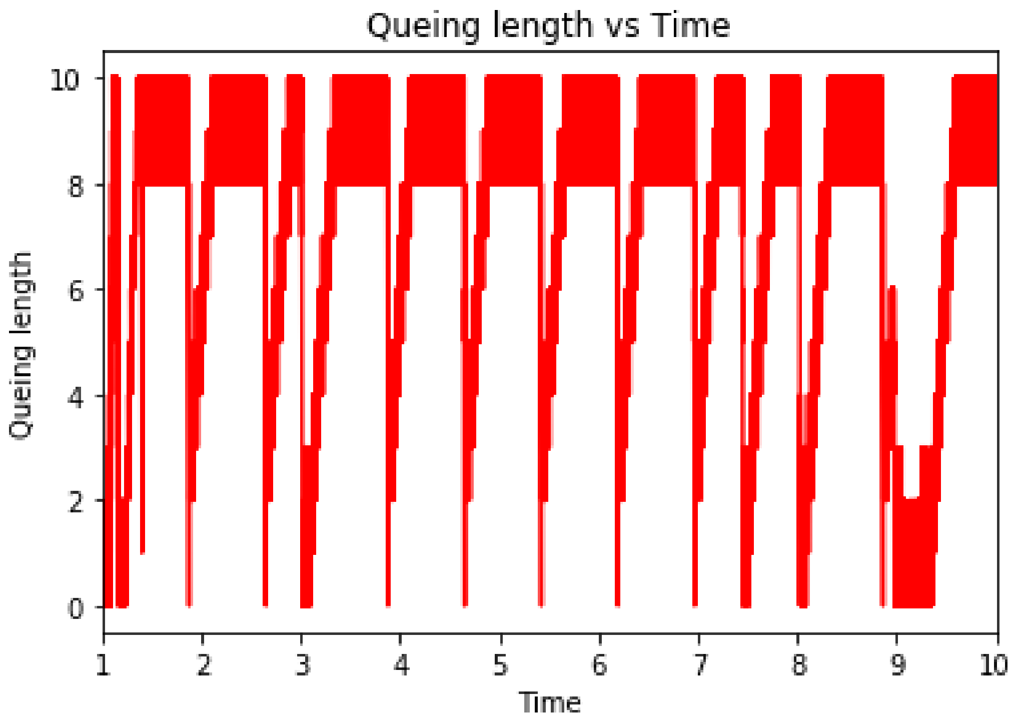
```
Number of times cwnd reduced by TCP algorithm = 43
```

Reasons for reduction in cwnd: - Loss of packets as evident by retransmissions in the tcp-example1.tr file. - The list seq stores the serial number of the transmissions(in tcp-example1.tr file) correspoding to cwnd reduction

```
In [19]: # Queing length vs time for Question 1
         length=0
         l1=[]
         t1=[]
         for line in open("tcp-example1.tr"):
             if line[0]=='+':
                 length=length+1
                 l1.append(length)
```

```
                t1.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
            if line[0]=='-':
                length=length-1
                l1.append(length)
                t1.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
        plt.plot(t1,l1,'r')
        plt.title("Queing length vs Time")
        plt.xlim([1,10])
        plt.ylabel("Queing length")
        plt.xlabel("Time")
```

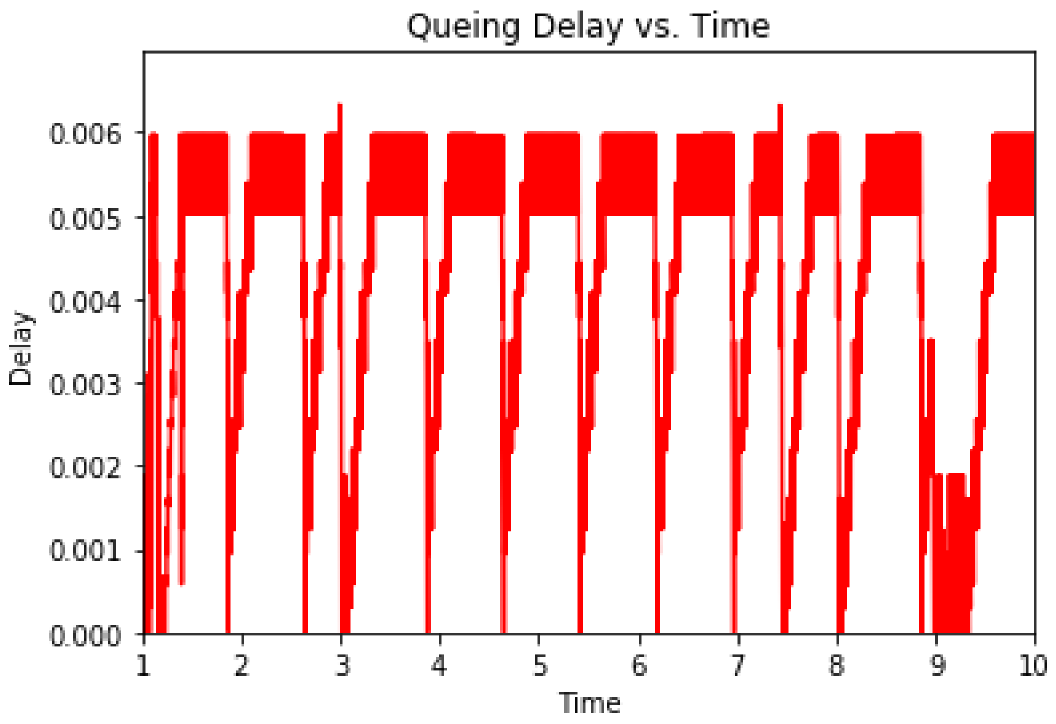
```
Out[19]: <matplotlib.text.Text at 0x11b1aec50>
```



```
In [20]: #Queuing Delay vs Time for Question 1
         flag=0
         t=0
         n=0
         time1=[]
         delay1=[]
         for line in open("tcp-example1.tr"):
             if line[0]=='+':
                     tm= float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
                     time1.append(tm)
                     t=tm
```

```
                flag+=1
                n+=1
        if line[0]=='-':
                tm=float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
                t=tm-time1[n-flag]
                delay1.append(t)
                flag-=1
plt.plot(time1[0:len(delay1)], delay1, 'r')
plt.xlim([1,10])
plt.ylim([amin(delay1),amax(delay1)*1.1])
plt.title("Queing Delay vs. Time")
plt.xlabel("Time")
plt.ylabel("Delay")
```

```
Out[20]: <matplotlib.text.Text at 0x11b847390>
```



2 Question 2

Average Throughput= 16Mbps (From wireshark conversations)

Maximum expected value of the throughput=Bandwidth= 50Mbps

In this case, the average throughput is not close to bandwidth in this case. The queuing delay, queue size and error rate were (almost)ideal for a bandwidth of 5Mbps but for their values in question 2, we are underutilizing the link. Increasing queue size and decreasing error rate and queuing delay will bring the average throughput close to the maximum possible throughput.

To change throughput, we can vary queue size, delay, simulation time and error rate

```
In [21]: # Increasing queue size to a certain limit increases average throughput
         # The below data is obtained from running pcap files on wireshark
         sz=[1,10,100,1000]
         tp=[14,16,18,18]
         f=pd.DataFrame({"Throughput(in Mbps)":tp, "Queue Size": sz})
         f[0:]
```

```
Out[21]:    Queue Size  Throughput(in Mbps)
         0           1                   14
         1          10                   16
         2         100                   18
         3        1000                   18
```

```
In [22]: # decreasing error rate upto a certain increases throughput
         # The below data is obtained from running pcap files on wireshark
         err=[1e-8, 1e-7, 5e-7, 1e-6,1e-5, 1e-4]
         atp=[30,36,24,16,4.76, 0.34]
         ff=pd.DataFrame({"Error Rate":err, "Throughput(in Mbps)":atp})
         ff[0:]
```

```
Out[22]:      Error Rate  Throughput(in Mbps)
         0  1.000000e-08                30.00
         1  1.000000e-07                36.00
         2  5.000000e-07                24.00
         3  1.000000e-06                16.00
         4  1.000000e-05                 4.76
         5  1.000000e-04                 0.34
```

```
In [23]: #decreasing delay increases throughput
         # The below data is obtained from running pcap files on wireshark
         atp2=[43,30,16,8.96]
         dl=[1,2.5,5,10]
         fff=pd.DataFrame({"Delay(in ms)":dl, "Throughput(in Mbps)":atp2})
         fff[0:]
```

```
Out[23]:    Delay(in ms)  Throughput(in Mbps)
         0           1.0                43.00
         1           2.5                30.00
         2           5.0                16.00
         3          10.0                 8.96
```

```
In [8]: # Throughput seems to slightly increase by decreasing simulation time
        # The below data is obtained from running pcap files on wireshark
```

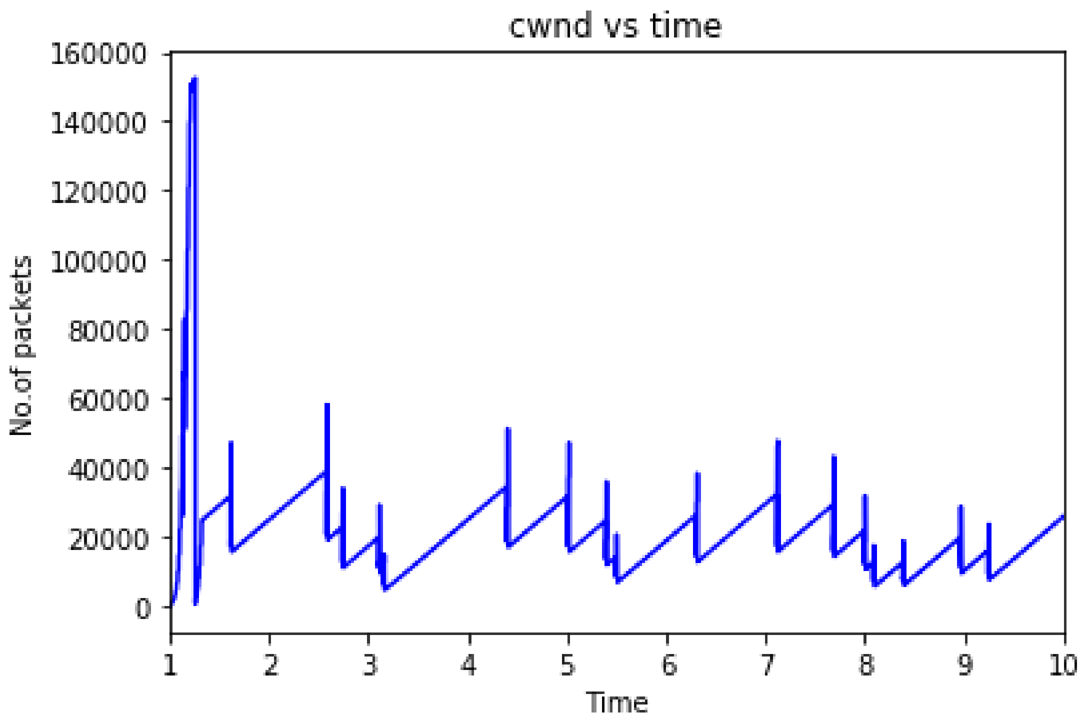
```
atp3=[18, 17,16, 16,15]
st=[5,7,10,15,20]
ffff=pd.DataFrame({"Simulation Time(in s)":st, "Throughput(in Mbps)":atp3})
ffff[0:]
```

```
Out[8]:    Simulation Time(in s)  Throughput(in Mbps)
        0                      5                   18
        1                      7                   17
        2                     10                   16
        3                     15                   16
        4                     20                   15
```

So by above analysis, for maximum average throughput if we set queue size=100, error rate= 1e-7, delay=1ms and simulation time= 10s, then we get an average throughput of 49Mbps, which is very close to the maximum throughput possible(50Mbps).

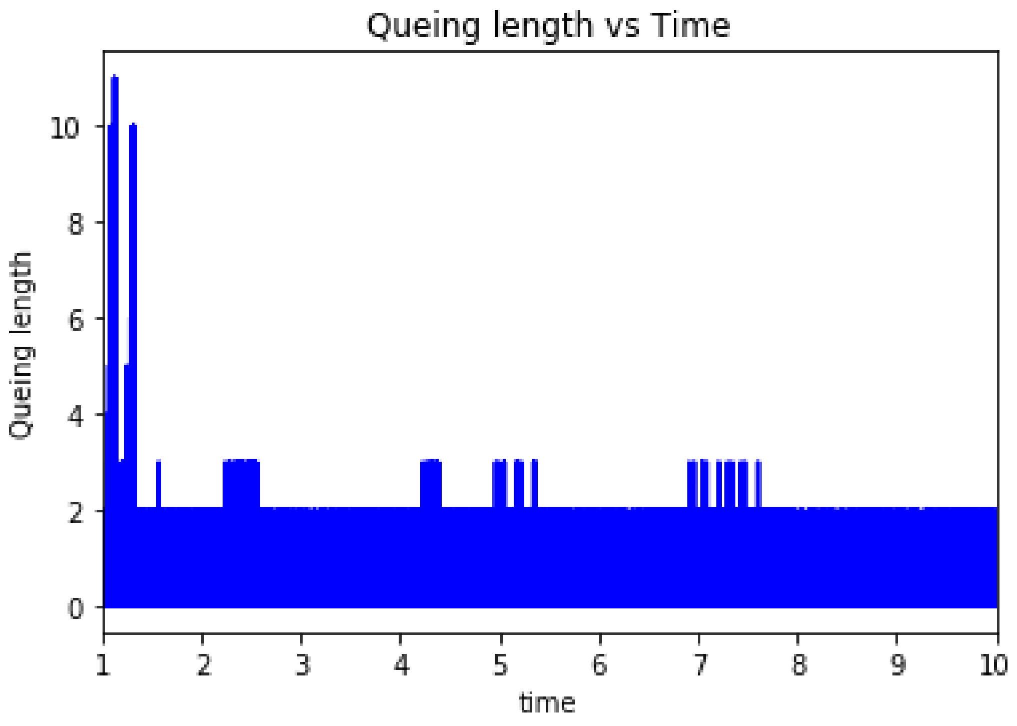
```
In [24]: # cwnd vs time for Question 2
         time, init, final= np.loadtxt("tcp-example2.cwnd", unpack=True)
         fig=plt.plot(time, init, 'b')
         plt.xlim([1,10])
         plt.xlabel("Time")
         plt.ylabel("No.of packets")
         plt.title("cwnd vs time")
```

```
Out[24]: <matplotlib.text.Text at 0x11c075ef0>
```



```
In [25]: # Queuing Length vs Time for Question 2

         length=0
         l=[]
         t=[]
         for line in open("tcp-example2.tr"):
             if line[0]=='+':
                 length=length+1
                 l.append(length)
                 t.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
             if line[0]=='-':
                 length=length-1
                 l.append(length)
                 t.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
         plt.plot(t,l,'b')
         plt.title("Queing length vs Time")
         plt.ylabel("Queing length")
         plt.xlabel("time")
         plt.xlim([1,10])
```

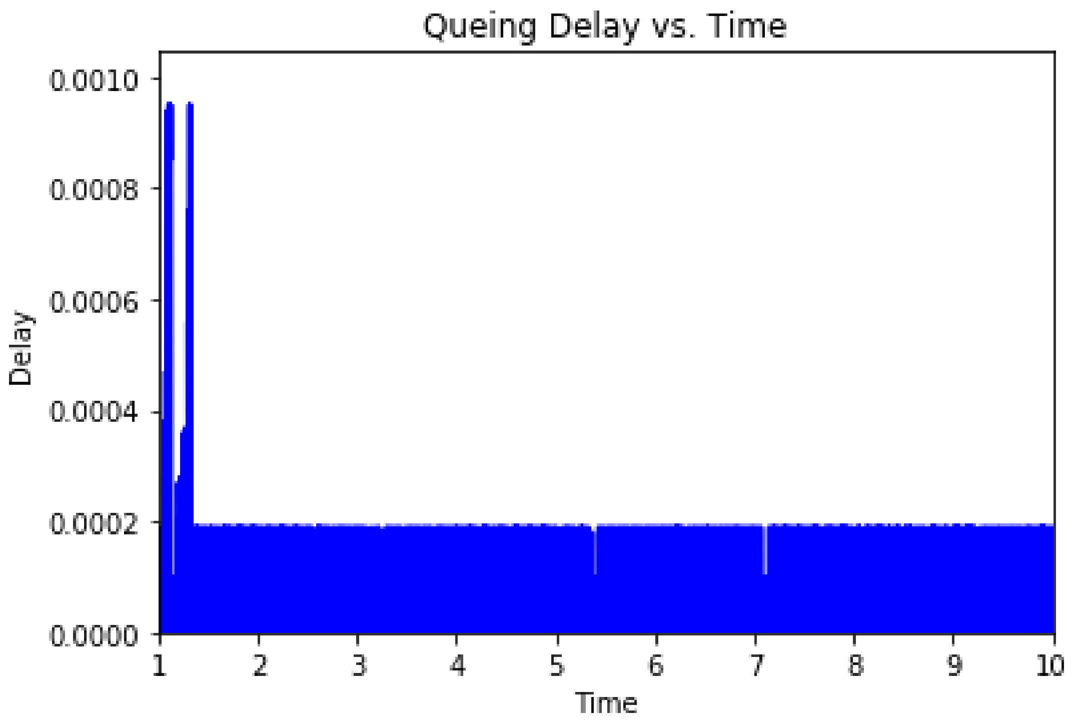
```
Out[25]: (1, 10)
```



```
In [26]: # queuing delay vs time for question 2
         flag=0
```

```
t=0
n=0
time=[]
delay=[]
for line in open("tcp-example2.tr"):
    if line[0]=='+':
            tm= float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
            time.append(tm)
            t=tm
            flag+=1
            n+=1
    if line[0]=='-':
            tm=float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
            t=tm-time[n-flag]
            delay.append(t)
            flag-=1
plt.plot(time[0:len(delay)], delay, 'b')
plt.xlim([1,10])
plt.ylim([amin(delay),amax(delay)*1.1])
plt.title("Queing Delay vs. Time")
plt.xlabel("Time")
plt.ylabel("Delay")
```

```
Out[26]: <matplotlib.text.Text at 0x11a931a20>
```



3 Question 3

Average Throughput= 0.779 kbps

Maximum Expected Throughput= 5 Mbps

In this case, maximum expected throughput is not equal to the bandwidth. This is because delay is very high which increases retransmissions due to timeouts and hence, limits the effective data transferred.

I will vary queue length,simulation time and delay to change average throughput.

```
In [27]: #Increasing queue size upto a certain limit increases throughput
         # The below data is obtained from running pcap files on wireshark
         tp2=[0.779, 3.687,3.687]
         qs=[10,100,1000]
         df2=pd.DataFrame({"Queue Size":qs, "Throughput(in Mbps)":tp2})
         df2[0:]
```

```
Out[27]:    Queue Size  Throughput(in Mbps)
         0          10                0.779
         1         100                3.687
         2        1000                3.687
```

```
In [28]: #Decreasing error rate does not increases throughput
         # The below data is obtained from running pcap files on wireshark
         tp3=[0.779,0.779,0.779, 0.456, 0.148]
         er=[1e-8,1e-7, 1e-6, 1e-5, 1e-4]
         df3= pd.DataFrame({"Error rate":er, "Throughput(in Mbps)":tp3})
         df3[0:]
```

```
Out[28]:      Error rate  Throughput(in Mbps)
         0  1.000000e-08                0.779
         1  1.000000e-07                0.779
         2  1.000000e-06                0.779
         3  1.000000e-05                0.456
         4  1.000000e-04                0.148
```

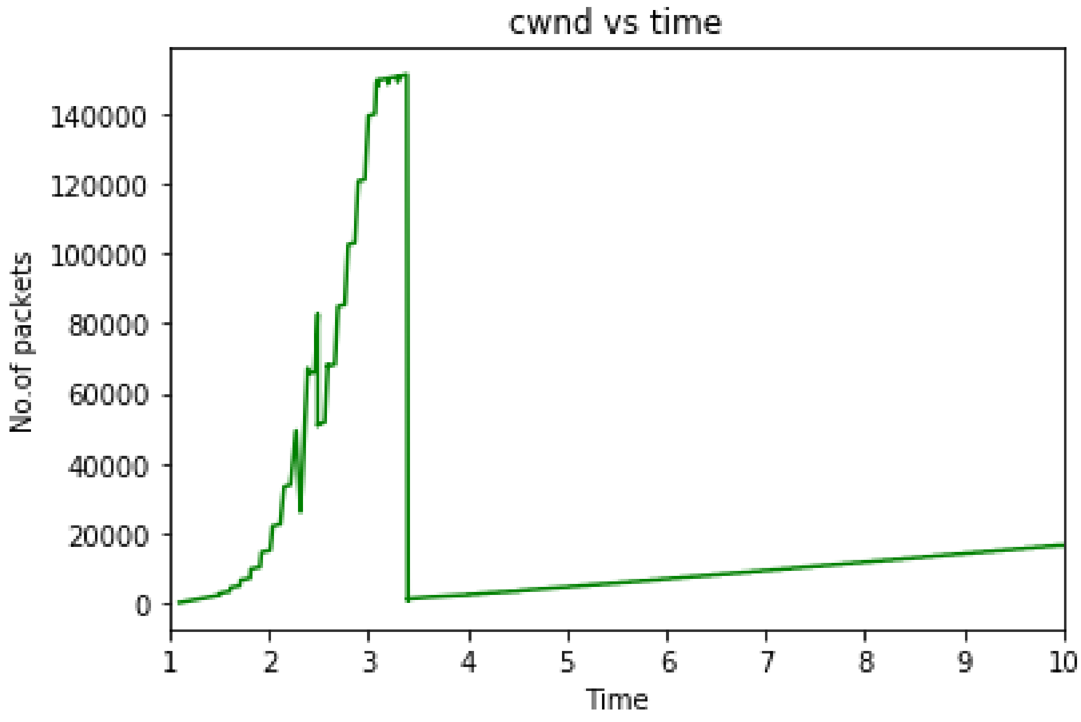
```
In [10]: # Throughput seems to slightly increase by increasing simulation time
         # The below data is obtained from running pcap files on wireshark
         tp4=[5,10,15,20]
         stm=[0.662,0.779,0.965, 1.242]
         df3=pd.DataFrame({"Simulation Time(in s)":stm, "Throughput(in Mbps)":tp4})
         df3[0:]
```

```
Out[10]:    Simulation Time(in s)  Throughput(in Mbps)
         0                  0.662                    5
         1                  0.779                   10
         2                  0.965                   15
         3                  1.242                   20
```

By above analysis, throughput does not change as much as it for question 2. This is due to the delay being too high which will cause more retransmissions. For queue size=100, error rate=1e-6 and simulation time=10s we get average throughput as 3.687 Mbps which is not that close to the maximum possible throughput(5 Mbps)

```
In [29]: # cwnd vs time for question 3
         time, init, final= np.loadtxt("tcp-example3.cwnd", unpack=True)
         fig=plt.plot(time, init, 'g')
         plt.xlim([1,10])
         plt.xlabel("Time")
         plt.ylabel("No.of packets")
         plt.title("cwnd vs time")
```

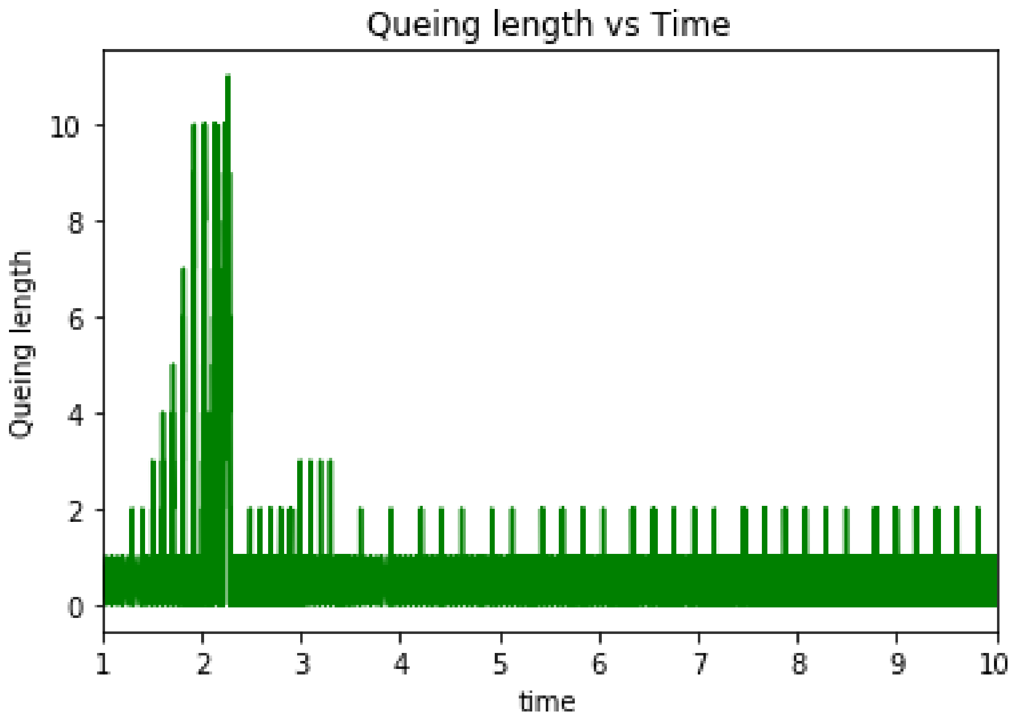
```
Out[29]: <matplotlib.text.Text at 0x11d0479e8>
```



```
In [30]: #queue length vs time for question 3
         length=0
         l=[]
         t=[]
         exit=0
         for line in open("tcp-example3.tr"):
             if line[0]=='+':
                 length=length+1
                 l.append(length)
```

```
            t.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
        if line[0]=='-':
            length=length-1
            l.append(length)
            t.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
    plt.plot(t,l,'g')
    plt.xlim([1,10])
    plt.title("Queing length vs Time")
    plt.ylabel("Queing length")
    plt.xlabel("time")
```

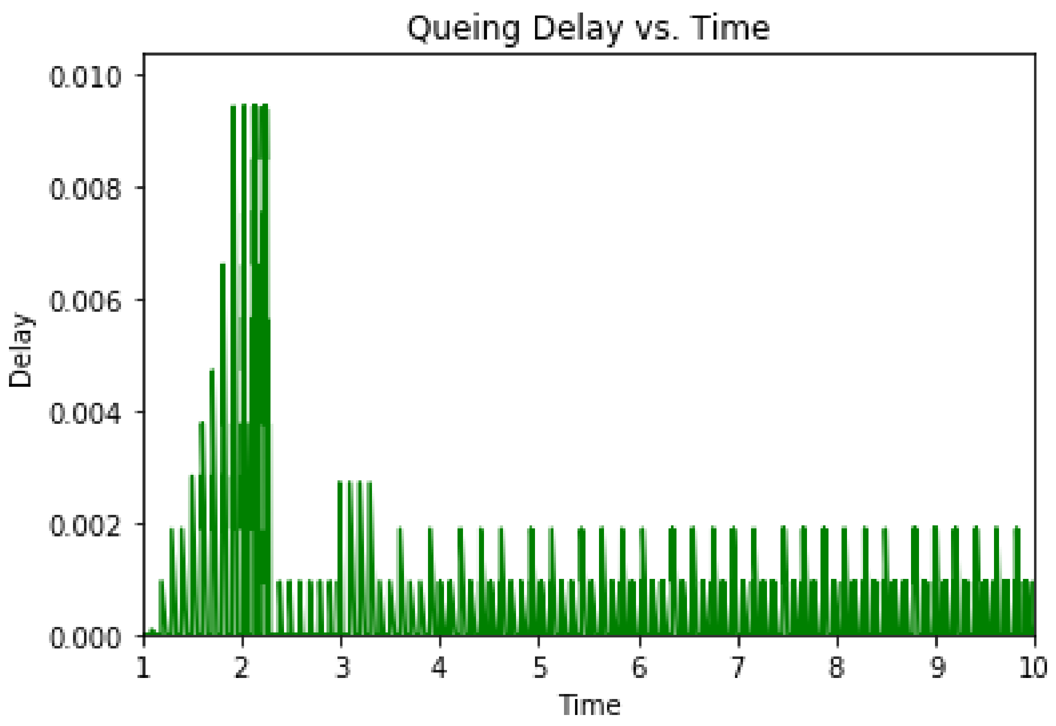
Out[30]:
```
<matplotlib.text.Text at 0x11875aa20>
```



In [31]:
```
# Queuing Delay vs Time for question 3
flag=0
t=0
n=0
time=[]
delay=[]
for line in open("tcp-example3.tr"):
    if line[0]=='+':
            tm= float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
            time.append(tm)
            t=tm
```

```
                flag+=1
                n+=1
        if line[0]=='-':
                tm=float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
                t=tm-time[n-flag]
                delay.append(t)
                flag-=1
plt.plot(time[0:len(delay)], delay, 'g')
plt.xlim([1,10])
plt.ylim([amin(delay),amax(delay)*1.1])
plt.title("Queing Delay vs. Time")
plt.xlabel("Time")
plt.ylabel("Delay")
```

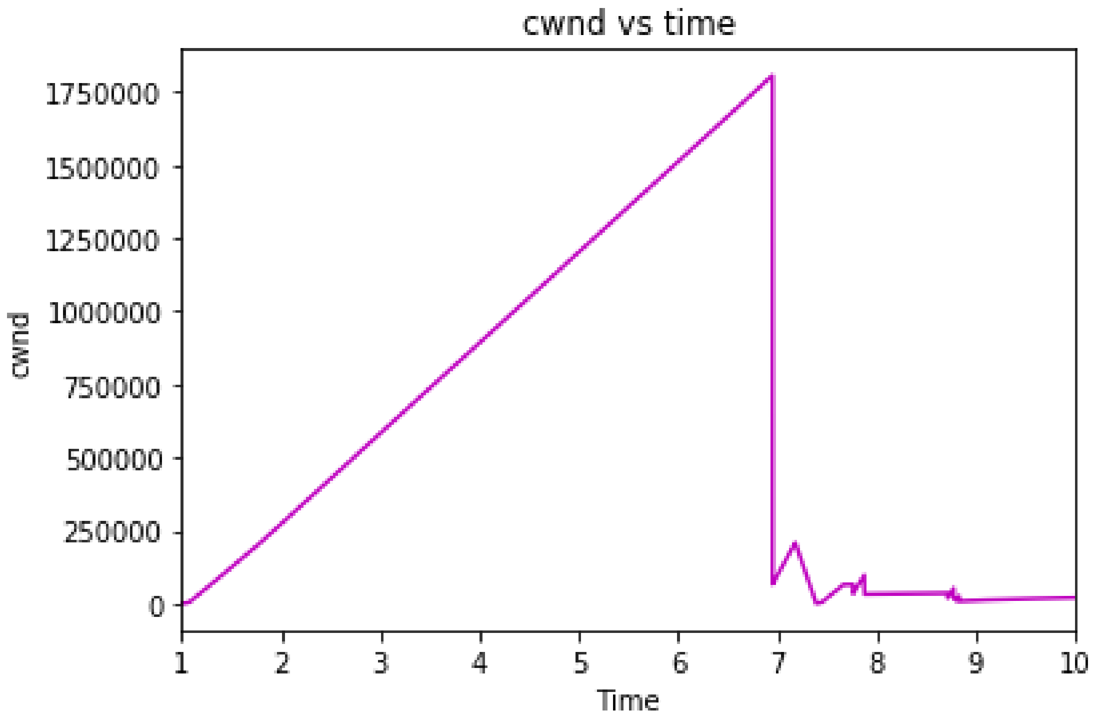
```
Out[31]: <matplotlib.text.Text at 0x11c6d1828>
```



4 Question 4

Throughput in Question 4 = 4.825 Mbps where as throughput in default configuration is 4.890 Mbps which means that the average throughput in both cases is approximately same. In default case, we are slightly underutilizing the link due to less queue size and in second case(queue size=1000), we have greatly increases the congestion. So, the net data transferred is almost identical in both cases despite different queue size.

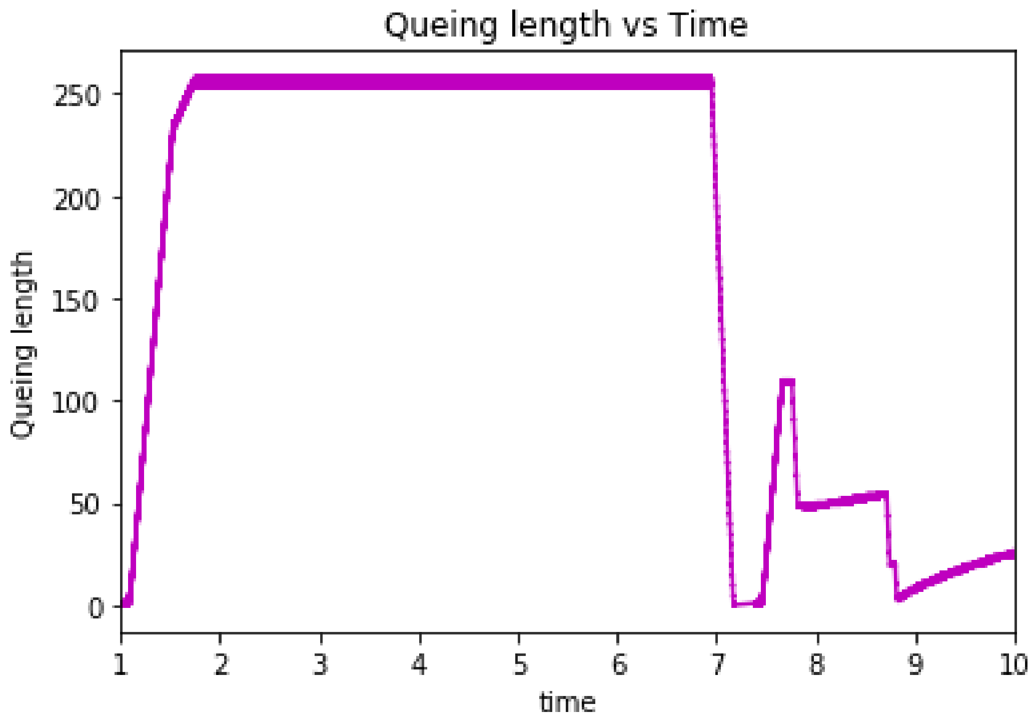
```
In [32]: # cwnd vs time for queue size=1000
         tim4, init4, final4= np.loadtxt("tcp-example4.cwnd", unpack=True)
         fig=plt.plot(tim4, init4, 'm')
         plt.xlim([1,10])
         plt.xlabel("Time")
         plt.ylabel("cwnd")
         plt.title("cwnd vs time")

Out[32]: <matplotlib.text.Text at 0x11ab3be80>
```



```
In [33]: # Queuing length vs time for queue size =1000
         length=0
         l4=[]
         t4=[]
         for line in open("tcp-example4.tr"):
             if line[0]=='+':
                 length=length+1
                 l4.append(length)
                 t4.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
             if line[0]=='-':
                 length=length-1
                 l4.append(length)
                 t4.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
         plt.plot(t4,l4,'m')
```

```
        plt.title("Queing length vs Time")
        plt.ylabel("Queing length")
        plt.xlabel("time")
        plt.xlim([1,10])
```

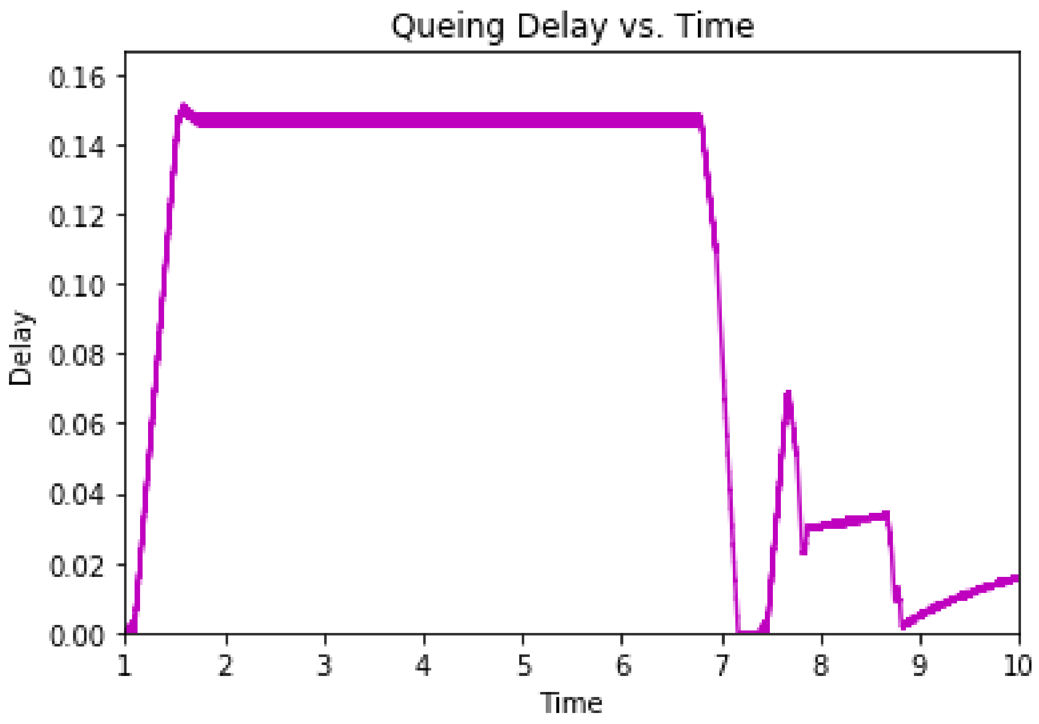
```
Out[33]: (1, 10)
```



```
In [34]: #Queuing Delay vs time for queue size=1000
         flag=0
         t=0
         n=0
         time=[]
         delay4=[]
         for line in open("tcp-example4.tr"):
             if line[0]=='+':
                     tm= float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
                     time.append(tm)
                     t=tm
                     flag+=1
                     n+=1
             if line[0]=='-':
                     tm=float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))])
                     t=tm-time[n-flag]
                     delay4.append(t)
```

```
                flag-=1
plt.plot(time[0:len(delay4)], delay4, 'm')
plt.xlim([1,10])
plt.ylim([amin(delay4),amax(delay4)*1.1])
plt.title("Queing Delay vs. Time")
plt.xlabel("Time")
plt.ylabel("Delay")
```

```
Out[34]: <matplotlib.text.Text at 0x11af624e0>
```

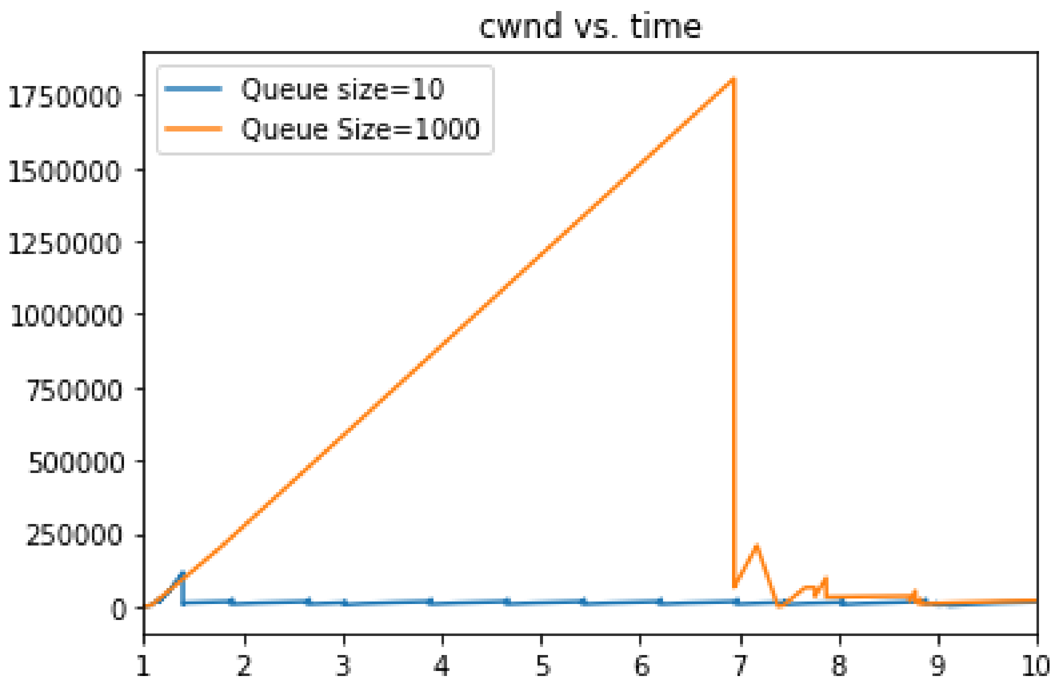


As evident by the graphs of queing delay and queing length, delay is maximum in the region where queing length is maximum and vice versa. Therefore the shapes of queuing delay and queuing length are same.

Comparisons between default config and with queue size=1000 with respect to cwnd, queuing length and queuing delay are as follows:

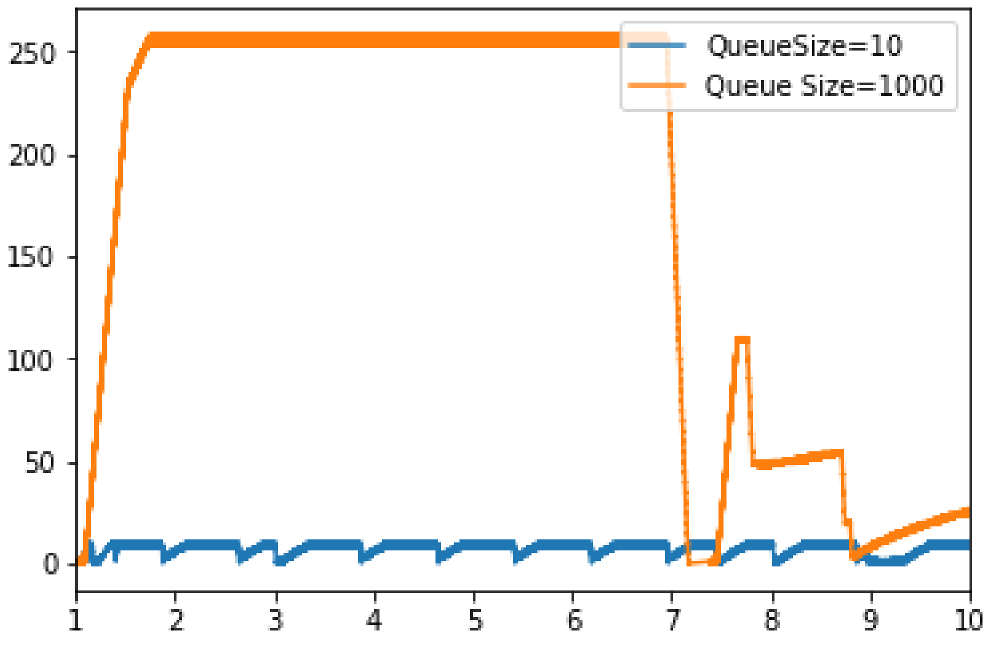
```
In [35]: #cwnd comparison
         plt.plot(tim1, init1)
         plt.plot(tim4, init4)
         plt.xlim([1,10])
         plt.legend(["Queue size=10", "Queue Size=1000"], loc='best')
         plt.title("cwnd vs. time")
```

```
Out[35]: <matplotlib.text.Text at 0x11b02b278>
```



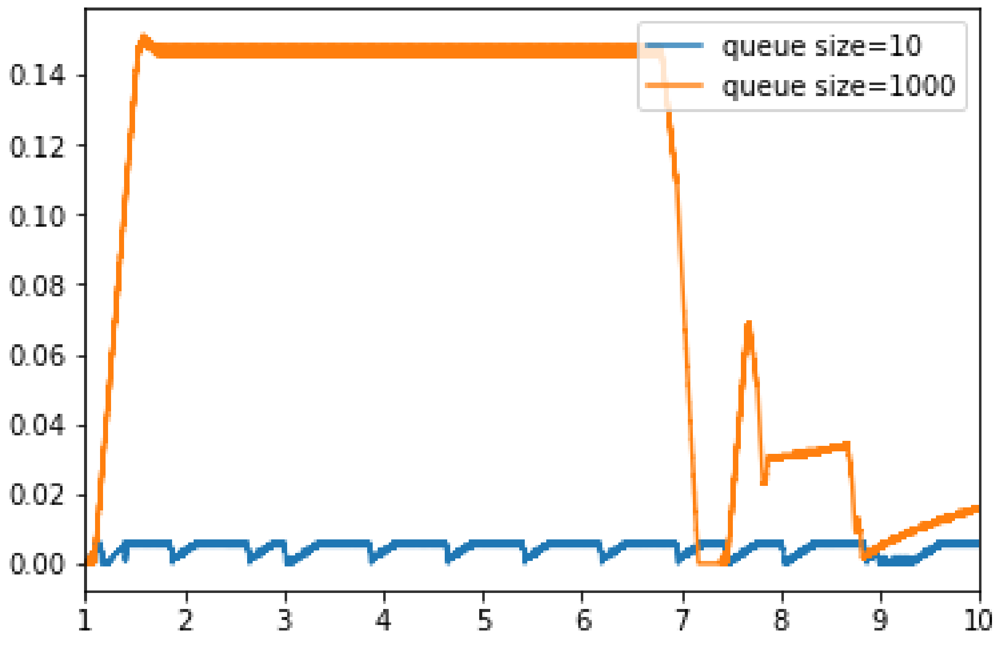
As queue size is small in default configuration, the cwnd in default configuration is very less due to less congestion and hence, less packets arriving at each router. Also due to smaller queue size in default configuration, cwnd is reduced a lot more that it does for the case where queue size=1000

```
In [36]: #queue length comparison
         plt.plot(t1,l1)
         plt.plot(t4,l4)
         plt.xlim([1,10])
         plt.legend(["QueueSize=10","Queue Size=1000"], loc='upper right')

Out[36]: <matplotlib.legend.Legend at 0x11d8aa908>
```



Due to greater queue size, queue length is greater for the case with queue size=1000 but queue length never reaches 1000 as due to greater congestion, queue delay and packet losses increase due to limitations of links, routers and processing time.

```
In [37]: #queuing Delay comparison
         plt.plot(time1[0:len(delay1)],delay1)
         plt.plot(time[0:len(delay4)],delay4)
         legend(["queue size=10", "queue size=1000"],loc='upper right')
         plt.xlim([1,10])

Out[37]: (1, 10)
```



Due to greater congestion at each router, delay experienced by the packets increases. Hence, packets experience much greater delay when the packet size is 1000 than the case when packet size is 10.

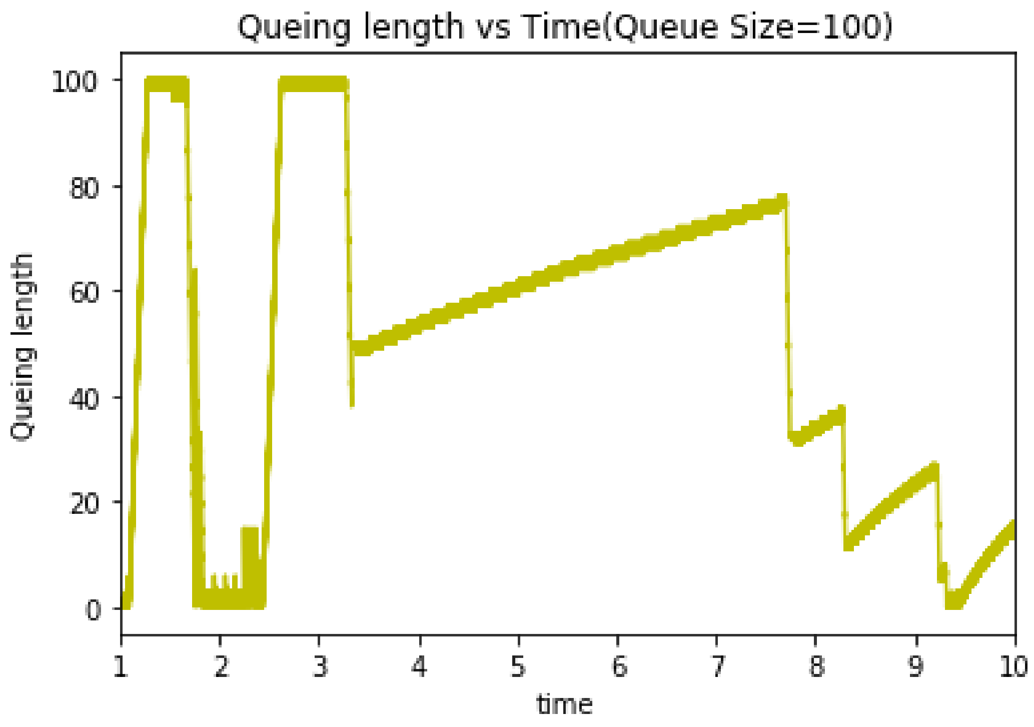
```
In [38]: print("Queuing Occupancy saturates at a queing length of :",np.amax(l))

Queuing Occupancy saturates at a queing length of : 11
```

Hence, Optimum queue size to be used should be 258 as queing occupancy saturates at that size which can be further analazyed by varying queue size.

```
In [39]: #For queue size=100
         length=0
         l100=[]
         t100=[]
         exit=0
         for line in open("tcp-example5.tr"):
             if line[0]=='+':
                 ni=line.find("length: ")
                 nf=line.find(" ", ni+8)
                 length=length+1
                 l100.append(length)
                 t100.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
             if line[0]=='-':
                 ni=line.find("length: ")
```

```
            nf=line.find(" ", ni+8)
            length=length-1
            l100.append(length)
            t100.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
    plt.plot(t100,l100,'y')
    plt.title("Queing length vs Time(Queue Size=100)")
    plt.ylabel("Queing length")
    plt.xlabel("time")
    plt.xlim([1,10])
```

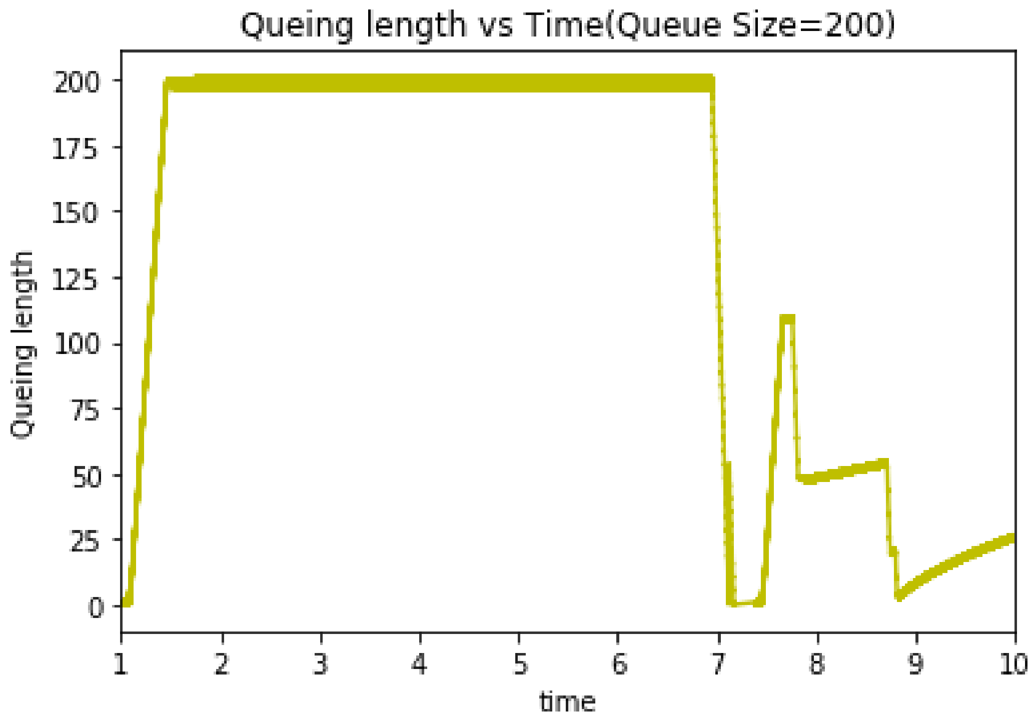
```
Out[39]: (1, 10)
```



```
In [40]: #For queue Size =200
         length=0
         l200=[]
         t200=[]
         exit=0
         for line in open("tcp-example6.tr"):
             if line[0]=='+':
                 ni=line.find("length: ")
                 nf=line.find(" ", ni+8)
                 length=length+1
                 l200.append(length)
                 t200.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
```

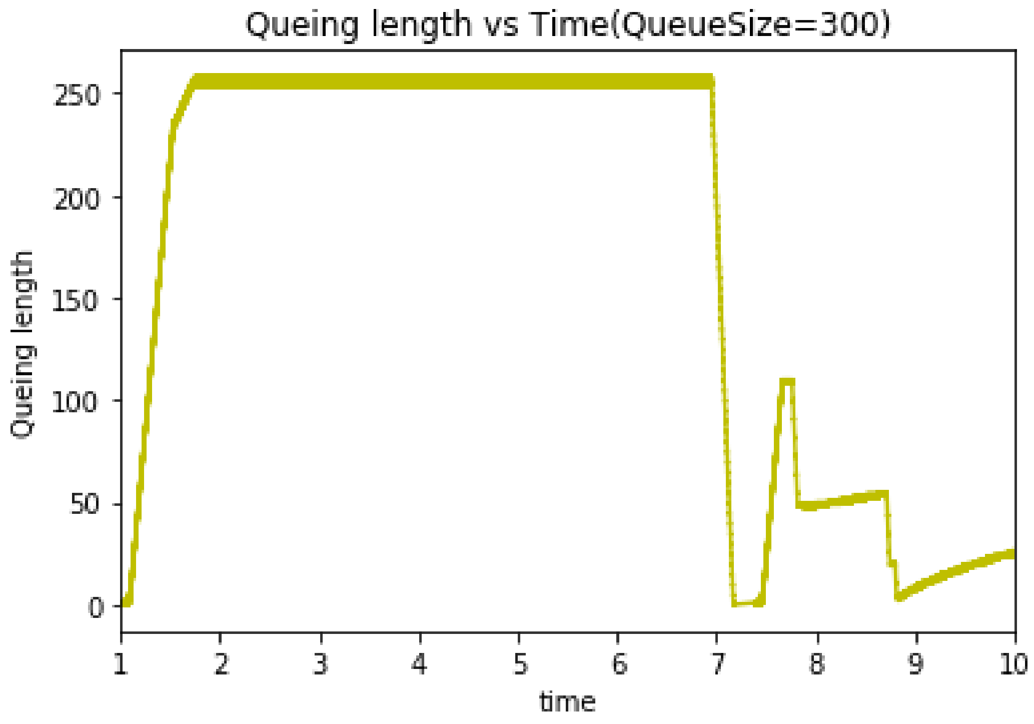
```
            if line[0]=='-':
                ni=line.find("length: ")
                nf=line.find(" ", ni+8)
                length=length-1
                l200.append(length)
                t200.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
        plt.plot(t200,l200,'y')
        plt.title("Queing length vs Time(Queue Size=200)")
        plt.ylabel("Queing length")
        plt.xlabel("time")
        plt.xlim([1,10])
```

```
Out[40]: (1, 10)
```



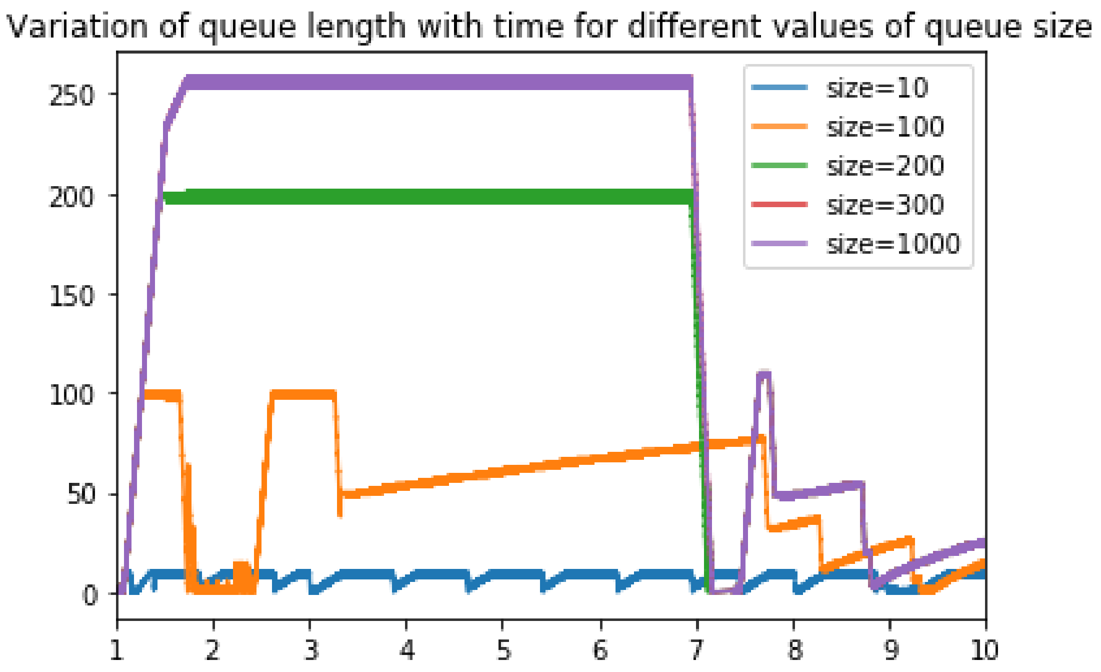
```
In [41]: #For queue size=300
        length=0
        l300=[]
        t300=[]
        exit=0
        for line in open("tcp-example7.tr"):
            if line[0]=='+':
                ni=line.find("length: ")
                nf=line.find(" ", ni+8)
                length=length+1
                l300.append(length)
                t300.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
            if line[0]=='-':
                ni=line.find("length: ")
                nf=line.find(" ", ni+8)
                length=length-1
                l300.append(length)
                t300.append(float(line[line.find(" ")+1: (line.find(" ", line.find(" ")+1))]))
        plt.plot(t300,l300,'y')
        plt.title("Queing length vs Time(QueueSize=300)")
        plt.ylabel("Queing length")
        plt.xlabel("time")
        plt.xlim([1,10])
```

```
Out[41]: (1, 10)
```



```
In [43]: # Queue length varaition with queue size
         plt.plot(t1,l1)
         plt.plot(t100,l100)
         plt.plot(t200,l200)
         plt.plot(t300,l300)
         plt.plot(t4,l4)
         plt.xlim(1,10)
         plt.legend(["size=10","size=100","size=200","size=300", "size=1000"], loc='upper right'
         plt.title("Variation of queue length with time for different values of queue size")
```

```
Out[43]: <matplotlib.text.Text at 0x11c55c7f0>
```



The above plot clearly shows that we cannot keep increasing queue size indefinitely. For size=10,100,200 the queue length saturates at queue size but for size= 300, 1000 the graph does not reach till the queue size. In fact, graphs for queue size=300 and queue size=1000 are almost identical. So, the optimum queue size is in between 200 and 300. By analyzing, we can also conclude that the optimum size must be the value at which the queue length for size=300 and size=100 saturates, which shown above is 258.

Therefore, optimum queue size=258

5 THE END