ECE 5566 NETWORK ARCHITECTURE AND PROTOCOLS
Project 1: Using simulation to evaluate IEEE 802.11 network performance

Code changes corresponding to the simulation setup

- Wifi network with 8 wifi mobile stations, an access point (AP) and a server

```
//creates two P2P nodes
NodeContainer p2pNodes;
p2pNodes.Create (2);
//create 8 wireless nodes
NodeContainer wifiStaNodes;
wifiStaNodes.Create (8);
//set one of the p2p nodes as the access point
NodeContainer wifiApNode=p2pNodes.Get (0);
```

- The P2P link between the AP and the server has a capacity of 500Mbps and delay of 1ms

```
//creates p2p channel and assign attribute values
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("1ms"));
```

- The WiFi network has a capacity of 1Mbps and the fragmentation threshold of the WiFi network is set to be 2346 Bytes and later 256 Bytes

```
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager","DataMode", StringValue ("DsssRate1Mbps"),"ControlMode", StringValue ("DsssRate1Mbps"),
"FragmentationThreshold", StringValue ("256"));
```

- Application running in each mobile station generates traffic following an ON-OFF model, where the duration of the “ON” state and the duration of the “OFF” state follow an exponential distribution with a mean of 4 seconds. During the “ON” state, the application generates a 1000 bytes UDP packet for every 60ms to send to the server

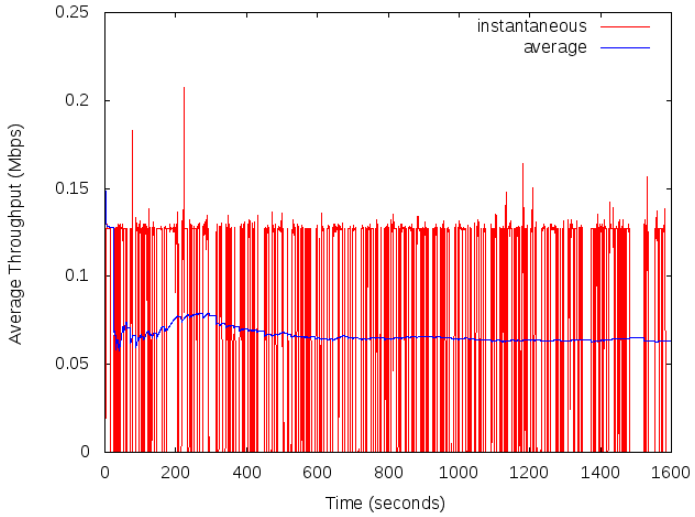
```
OnOffHelper onoff("ns3::UdpSocketFactory",InetSocketAddress(p2pInterfaces.GetAddress(1),10));
onoff.SetAttribute("DataRate", DataRateValue(DataRate("0.13Mbps")));
onoff.SetAttribute("PacketSize", UintegerValue(1000));
int i;
ApplicationContainer apps;
for(i=0;i<8;i++)
{
        SeedManager::SetRun (2*i+1);
        onoff.SetAttribute("OnTime", StringValue ("ns3::ExponentialRandomVariable[Mean=4]"));
        SeedManager::SetRun (2*i+2);
        onoff.SetAttribute("OffTime", StringValue ("ns3::ExponentialRandomVariable[Mean=4]"));
        apps = onoff.Install(wifiStaNodes.Get(i));
        apps.Start(Seconds (2.0));
        apps.Stop(Seconds (1600.0));
}
```

Exercises

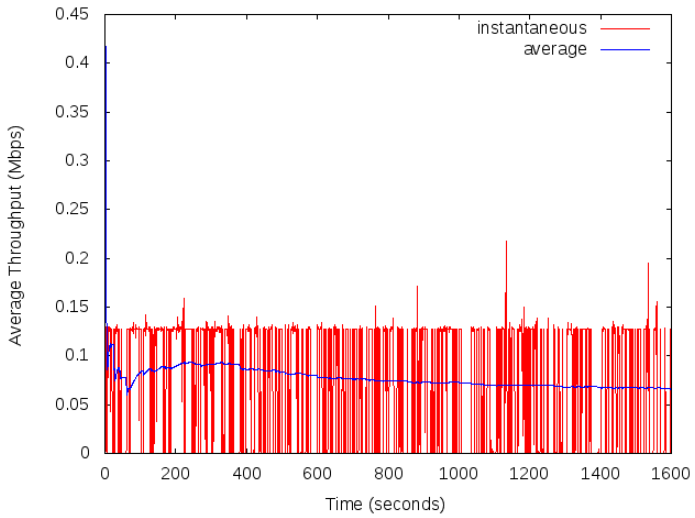
(1) Based on the simulation output data, plot a graph to show the throughput vs time for each mobile station in this network.

I have used Flow Monitor to study the statistics of each node flow. From this information, two types of throughput were computed: the instantaneous value and the average value. Ideally, the instantaneous throughput would be the data rate (0.13 Mbps) for on-periods and 0 for the off-periods. The average throughput is computed over the entire simulation time (both on- and off-periods) and hence must be around half the data rate ((0.13+0)/2=0.065 Mbps). The simulation results closely follow the same. This is observed both in the graphical output and by the average value obtained by taking the mean from the output dataset. Any minor discrepancy is due to the granularity of the monitoring interval.

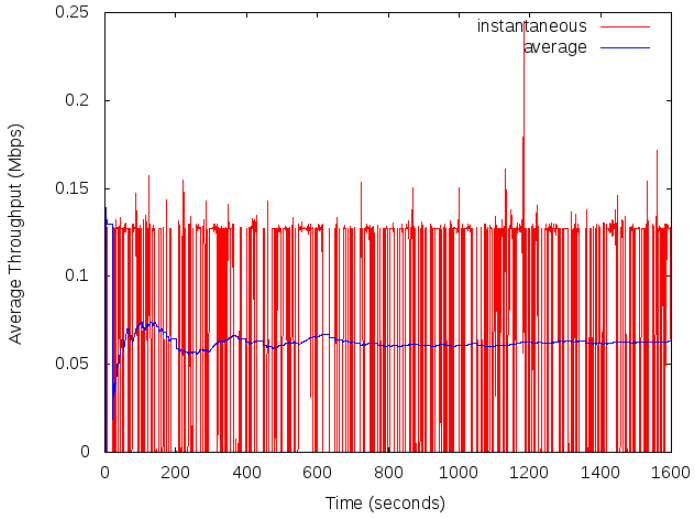
Node 1



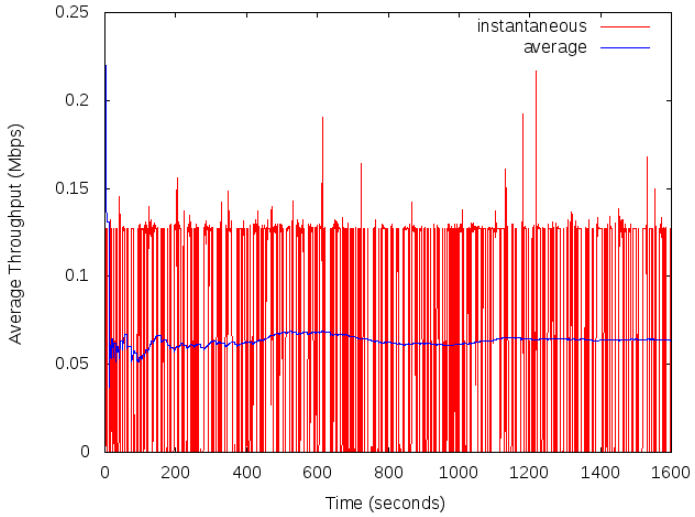
Node 2



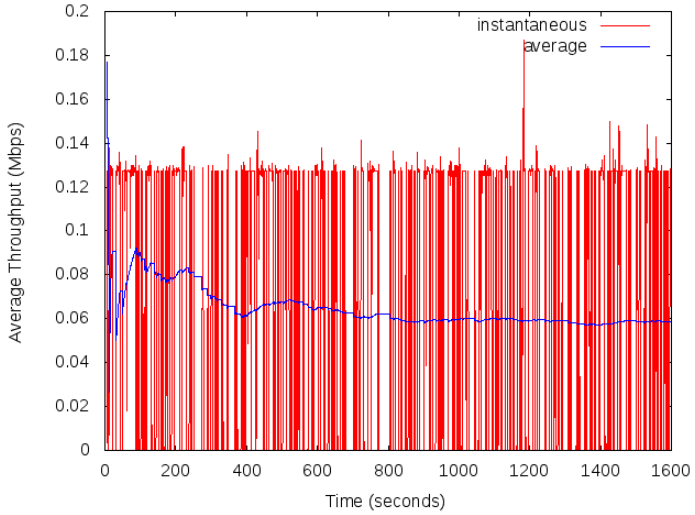
Node 3



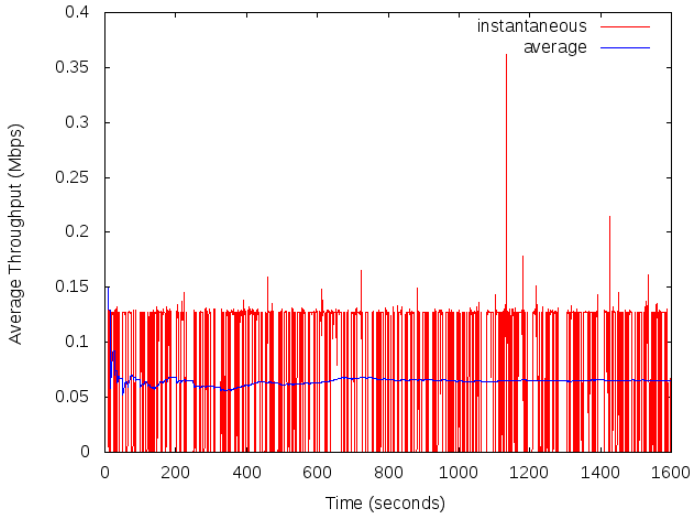
Node 4



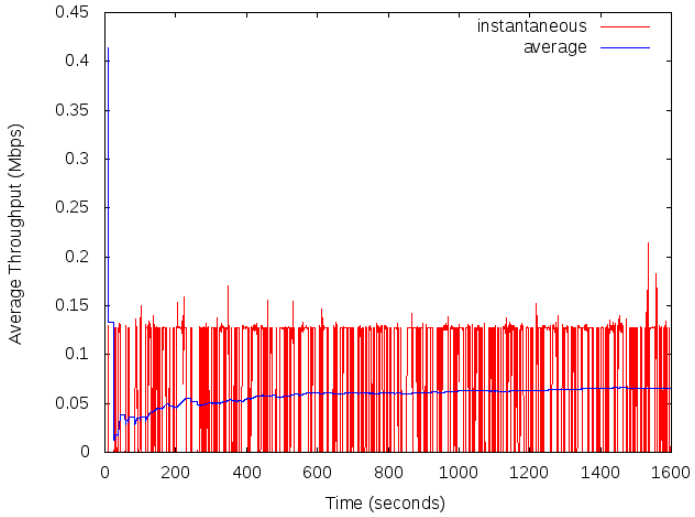
Node 5



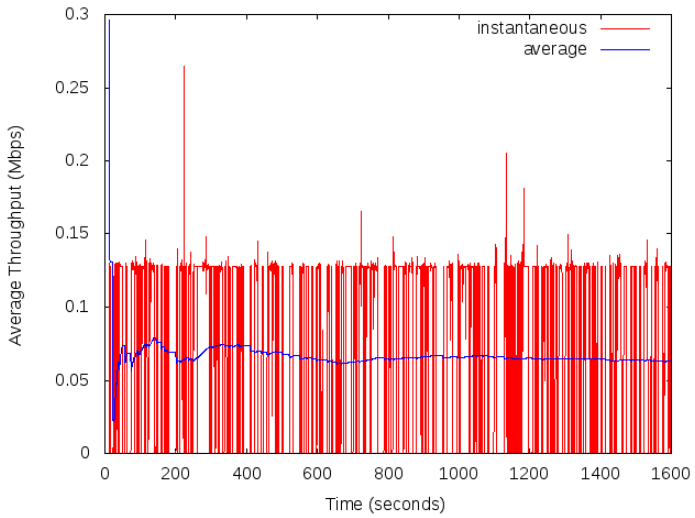
Node 6



Node 7



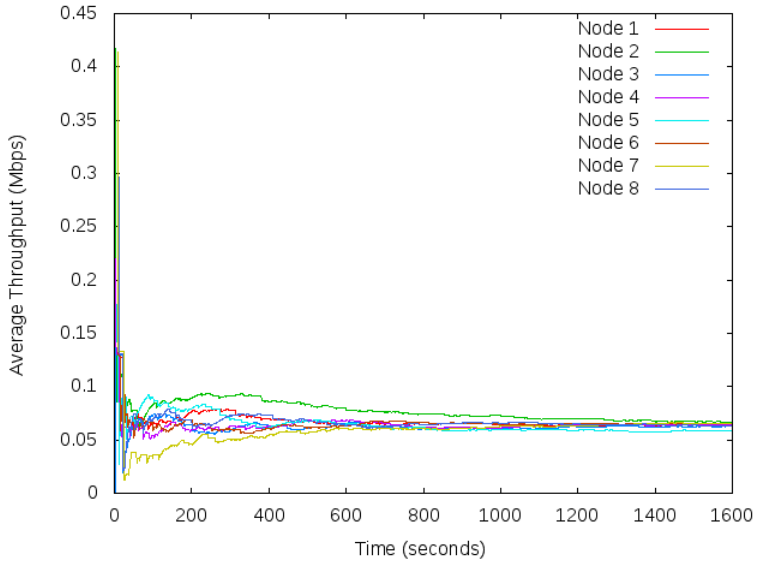
Node 8

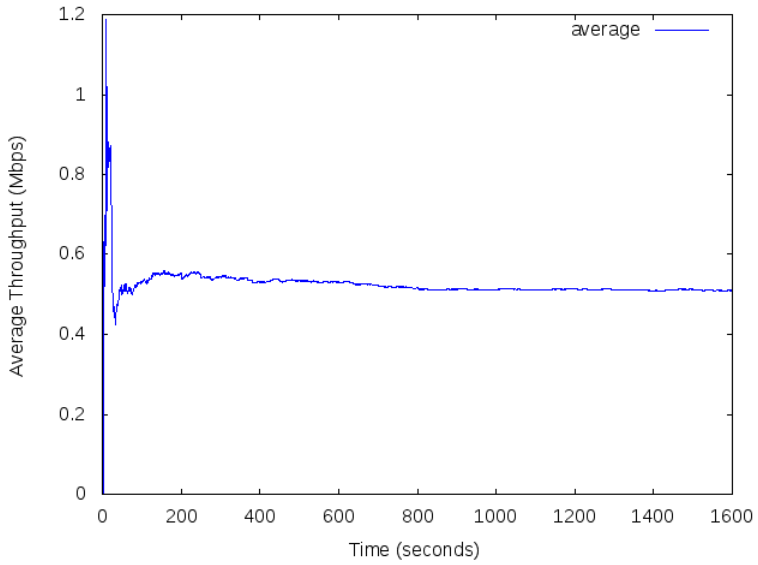


I have referred to examples worked out in the ns-3 manuals and tutorials to set up the flow monitor and calculate throughput.

(2) What is the average throughput of the network at steady-state?

The average throughput of the network is computed by summing together the throughputs from each of the nodes. The steady-state is achieved when the average throughput of all the nodes converge. From analyzing the number of nodes active for a given instant, on an average, half of the nodes are in the on state and the other half are off (The output dataset gives a mean of 4.3 nodes). The network throughput in this case would be 4*0.13=0.52 Mbps. This is closed to what is observed in the graphical output.





What is the confidence interval for the average throughput? Explain how you design the experiment to obtain the confidence interval.

The program outputs a data file with 3179 average throughput values spanning a simulation time from 2 to 1600 seconds. In order to compute the confidence interval, I copied these values onto an Excel worksheet and followed the method of batch means procedure described in the class (Lecture 11A4_Calculating confidence interval).

Procedure

1. I divided the samples into 6 batches of 500 samples each (last 179 values were ignored).
2. For each of these batches I computed the sample mean and sample standard deviation using the excel in-build formulae.

Batch	Sample Mean	Mean	Standard Deviation
1 (1-500)	0.551212728	0.525530296	0.016229602
2 (501-1000)	0.537305468		
3 (1001-1500)	0.52733493		
4 (1501-2000)	0.513291824		
5 (2001-2500)	0.512556534		
6 (2500-3179)	0.511480294		

3. In order to calculate the 90% confidence intervals, I used the formula

$$(\hat{\mu}_M - z_{\alpha,M-1}\frac{\hat{\sigma}_M}{\sqrt{M}}, \hat{\mu}_M + z_{\alpha,M-1}\frac{\hat{\sigma}_M}{\sqrt{M}})$$

M	6
Confidence interval	90%
z value for M-1	2.015
Confidence interval lower limit	0.512179496
Confidence interval upper limit	0.538881097

4. The dataset and computation details used in this experiment are provided as a spreadsheet in the other folder.

(3) How you make sure that you correctly set the random number generator so that there is no undesirable correlation in your simulation?

According to the random variable section of the NS-3 manual, "the more statistically rigorous way to configure multiple independent replications is to use a fixed seed and to advance the run number" (https://www.nsnam.org/docs/manual/html/random-variables.html). In the code, I have used the function "SeedManager::SetRun(#)" in a loop to set a different run number prior to creating the exponential random variables for the on and off duration.

```
for(i=0;i<8;i++)
{
        SeedManager::SetRun (2*i+1);
        onoff.SetAttribute("OnTime", StringValue ("ns3::ExponentialRandomVariable[Mean=4]"));
        SeedManager::SetRun (2*i+2);
        onoff.SetAttribute("OffTime", StringValue ("ns3::ExponentialRandomVariable[Mean=4]"));
        apps = onoff.Install(wifiStaNodes.Get(i));
        apps.Start(Seconds (2.0));
        apps.Stop(Seconds (1600.0));
}
```

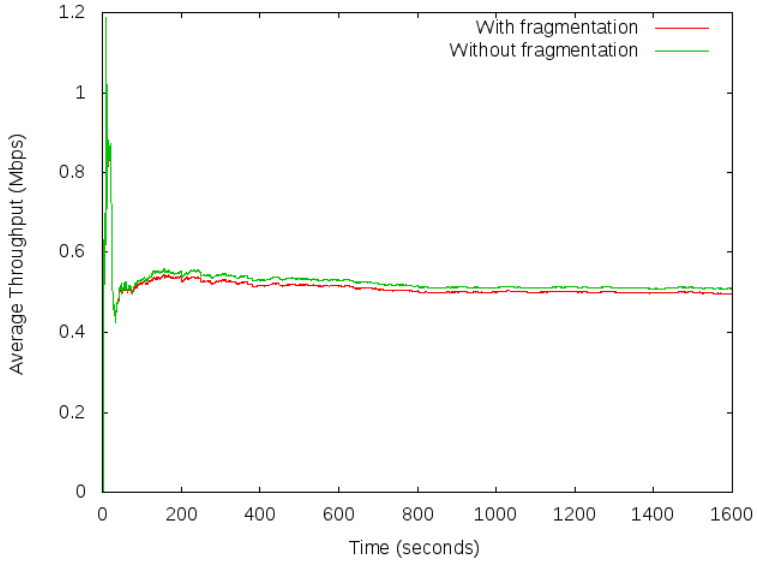
(4) Roughly, how long does it take for the network to reach steady-state? How did you figure that out?

In order to determine when the network reaches steady-state, the average throughput data must be analyzed to identify the time when the effects of the initial perturbations have died down and enough time has gone by to equalize its effects on the network throughput. From the 3179 average

throughput values, I found that the truncated mean throughput is 0.52Mbps after 2400 values. In other words, the steady-state is attained around 1200 seconds.

(5) Change the fragmentation threshold of the WiFi network to 256Bytes. Rerun your experiments. Can you see any difference in the throughput of the mobile stations? If there is a difference, explain why there is this difference.

On rerunning the experiments and comparing the average network throughput, it is observed that the throughput is consistently lower when the fragmentation threshold is set to 256 Bytes. This is due to the overhead caused by the headers for each of the extra fragments.



I also noticed that the number of packets dropped also increases when the fragmentation threshold is lower.

With fragmentation set to 2346 bytes

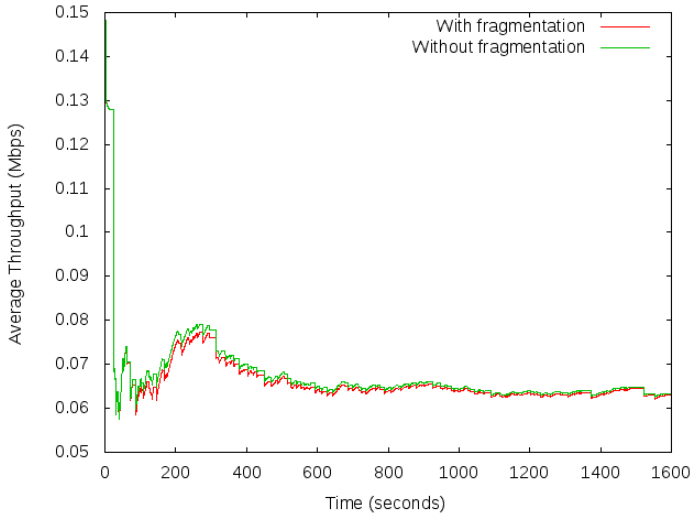
```
Lost packets count
Node 0:1
Node 1:27
Node 2:8
Node 3:17
Node 4:4
Node 5:6
Node 6:2
Node 7:22
student@ubuntu:~/Wor
```

With fragmentation set to 256 bytes

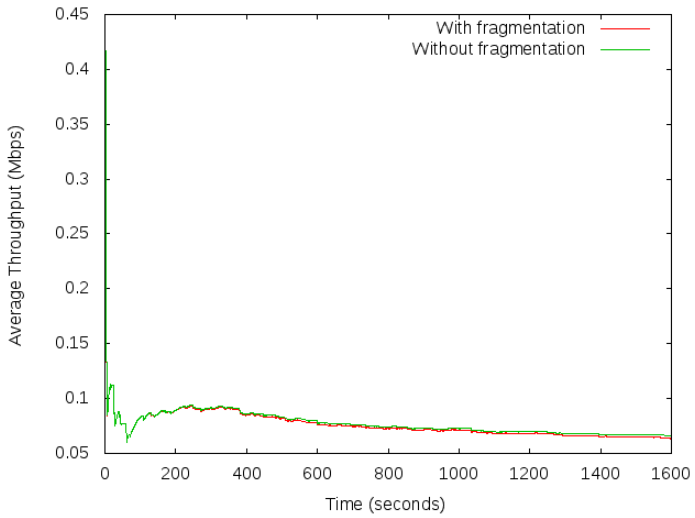
```
Lost packets count
Node 0:85
Node 1:518
Node 2:486
Node 3:403
Node 4:64
Node 5:241
Node 6:273
Node 7:482
student@ubuntu:~/Work
```

The throughput comparison for each of the individual nodes is shown below.

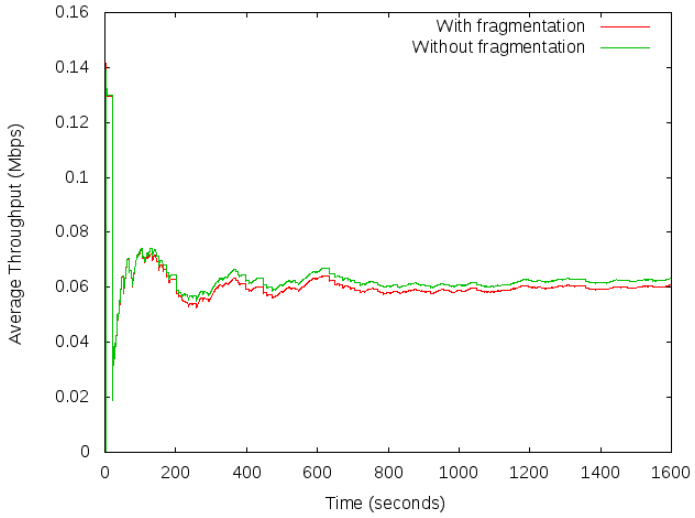
Node 1



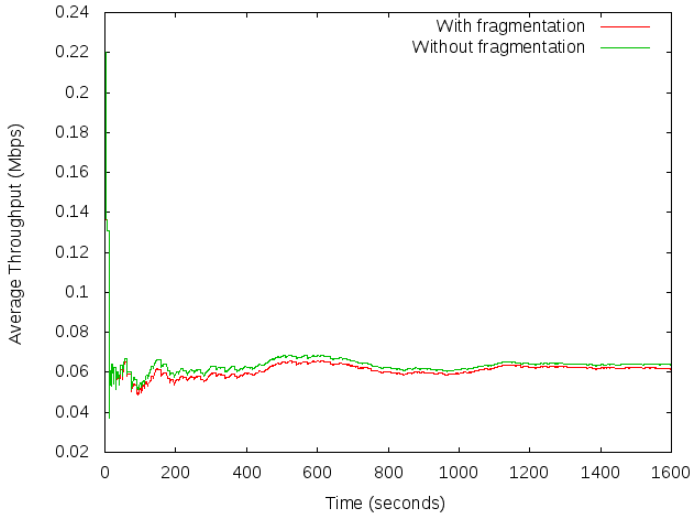
Node 2



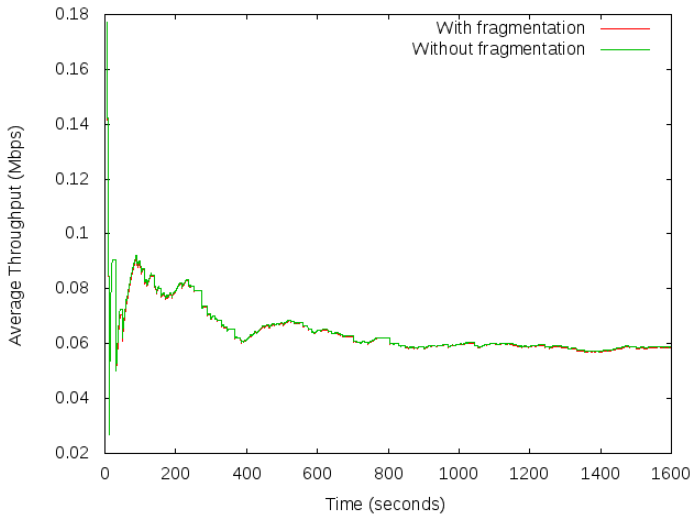
Node 3



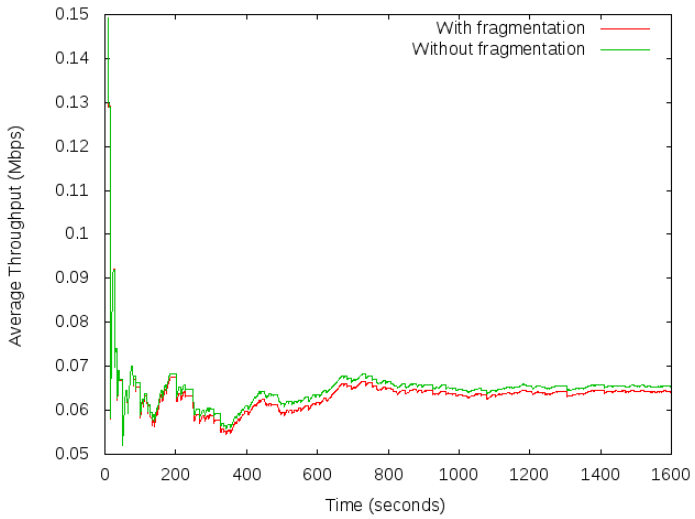
Node 4



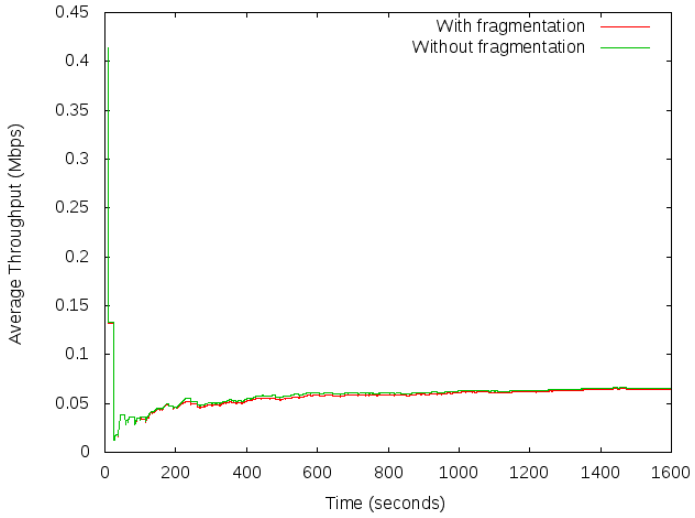
Node 5



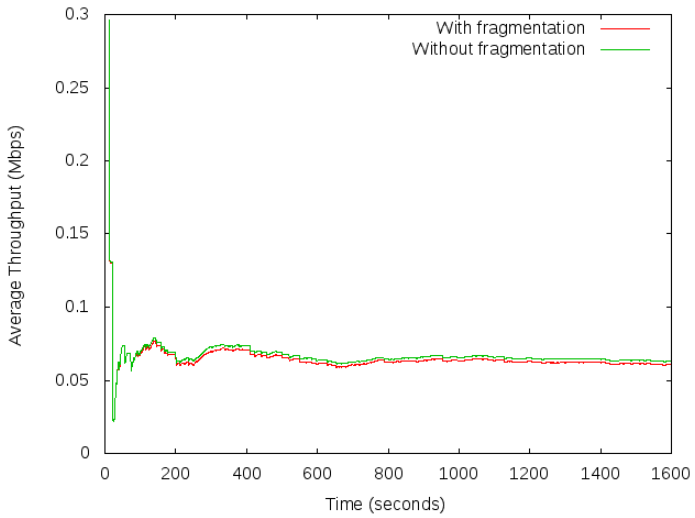
Node 6



Node 7



Node 8



(6) Can you come up with a theoretical analysis of the steady-state throughput of the network? You can make simplifications in your assumptions so that your theoretical analysis can be tractable. But you have to point out the potential negative impact of the simplification. Compare your theoretical analysis results with the simulation results. Is the theoretical value close to the simulation? If not, why?

There are two aspects to be addressed in describing this network:

- Analysis of the states and state transitions of each on-off traffic source
- Analysis of multiplexing many such on-off sources to form the network

Analysis of the states and state transitions of each on-off traffic source

An on-off traffic source alternates between being in an "on" state or in an "off" state. It remains in each of these states for an exponentially distributed duration of mean $1/a$ and $1/b$ respectively. In the "on" state, the source generates constant bit rate traffic at rate $R\ bps$ and there is no transmission

in the “off” state. As there are only two states, the possibility of being in the “on” and “off” state sum to 1. The average duration for which the node is active is given by $p = a/(a + b)$.

Plugging in the values for this simulation,

$$a = b = \frac{1}{4}, R = \frac{1000\ bytes}{60\ msec} = \frac{1000 * 8\ bits}{60 * 10^{-3} sec} = 1.3 * 10^5 bps = 0.13\ Mbps$$

$$Duration\ for\ which\ node\ is\ active = \frac{a}{a+b} = \frac{1/4}{1/4 + 1/4} = \frac{1}{2}$$

$$Average\ throughput\ for\ a\ node = Active\ duration * Rate = \frac{1}{2} * 0.13 Mbps = 0.065\ Mbps$$

This is consistent with what has been observed from the simulation results.

Analysis of multiplexing many such on-off sources to form the network

In our network, M such on-off sources are superposed to transmit over a single channel. Each of the sources independently transition between the states and generate traffic. This can be thought of as a Bernoulli distribution. The parameter of importance is the activity factor that determines the number of active nodes contributing to the throughput. The probability of j nodes being active out of M is

$$\binom{M}{j} p^j (1-p)^{M-j}$$

For our case, $M = 8$. The probability of 1, 2…8 nodes being active and the network throughput in these cases is analyzed. Summing up the product of throughput and the probability of their occurrence gives the average network throughput. This is approximately the same as that got using simulation.

j	Probability	Throughput	Throughput*probability
1	0.03125	0.13	0.0040625
2	0.109375	0.26	0.0284375
3	0.21875	0.39	0.0853125
4	0.2734375	0.52	0.1421875
5	0.21875	0.65	0.1421875
6	0.109375	0.78	0.0853125
7	0.03125	0.91	0.0284375
8	0.00390625	1.04	0.0040625
Average network throughput			**0.52**

The reference papers used in this question are listed below.

- Chiotis, Tryfon, F. Stanatelopoulos, and Basil Maglaris. "Traffic source models for realistic ATM performance modelling." Proc. 5th IFIP Workshop on Performance Modelling and Evaluation of ATM Networks. 1997.
- Chandrasekaran, Balakrishnan. "Survey of network traffic models." Waschington University in St. Louis CSE 567 (2009).