Network Simulation Lab Report

Spencer Gardner

January 29, 2015

1 Introduction

In this lab, we use network simulation library "Bene" to simulate various scenarios involving network traffic through nodes. First, we will examine traffic through 2 nodes and one link. Then, we will use a network of 3 nodes and 2 links of varying speeds to analyze queuing behavior. Finally, we will use the simulator so simulate a larger scale queuing scenario and compare our results with a theoretical formula.

2 Two Nodes

For our first set of trials, we construct a network of two nodes, n1 and n2 and create a unidirectional link from n1 to n2 to connect them. The following code block illustrates this.

```
1 n1 = net.get_node('n1')
2 n2 = net.get_node('n2')
3 n1.add_forwarding_entry(address=n2.get_address('n1'),link=n1.links[0])
```

2.1 First Trial

In the first trial, a 1000B packet is put through a link running at 1Mbps with a propagation delay of 1000ms.
Network Configuration Used: `n1 n2 1Mbps 1000ms`

ID	Size	Time Created	Time Received
1	1000B	0 s	1.008s

2.2 Second Trial

In the second trial, a 1000B packet is put through a slower link running at 100bps, but with a lower propagation delay of only 10ms.
Network Configuration Used: `n1 n2 100bps 10ms`

ID	Size	Time Created	Time Received
1	1000B	0 s	80.01s

2.3 Third Trial

For the final trial, we send 3 packets through the link.
Network Configuration Used: `n1 n2 1Mbps 10ms`

ID	Size	Time Created	Time Received
1	1000B	0 s	0.018 s
2	1000B	0 s	0.026 s
3	1000B	2 s	2.018 s

2.4 Proof of Simulator Accuracy

2.4.1 First Trial

Since $\frac{1000B}{1Mbps} = \frac{8000b}{1000000bps} = .008s$ and propagation delay is 1.00s, we obtain 1.008s as total delay.

2.4.2 Second Trial

Since $\frac{1000B}{100bps} = \frac{8000b}{100bps} = 80.00s$ and propagation delay is 10ms=.01s, we obtain 80.01s as total delay.

2.4.3 Third Trial

For packet 1 $\frac{1000B}{1000Mbps} = \frac{8000b}{1000000bps} = .008s$ and propagation delay is 10ms=.01s, we obtain .018s. Immediately after packet 1 is sent (after .008s for transmission), packet 2 is sent. Transmission takes .08s, so .018s for the first packet and .08s for the second packet yields .026 seconds for the first two.

Then, for packet 3, there is a 2 second wait from the beginning of the simulation. Therefore, there is effectively no time effect from the first two packets since $2.00s > .026s$ So $2.00s + .018s$ for packet transmission yields 2.018s as the final result.

3 Three Nodes

In this section, we set up a network consisting of 3 nodes: n1, n2, and n3. We will run two trials on this network with differing link configurations. The following code sets up the forwarding entries for the nodes:

```
1 n1.add_forwarding_entry(address=n3.get_address('n2'),link=n1.get_link('n2'))
2
3     n2.add_forwarding_entry(address=n2.get_address('n1'),link=n2.get_link('n3'))
```

We will run 1000 packets of 1000B through each of the two trials. However, we will vary link speeds to draw conclusions about the effects of transmission versus propagation speeds.

3.1 First Trial

In this trial, we configure the two links in the network with equal rates:

```
1 n1  n2  1Mbps  100ms
2 n2  n3  1Mbps  100ms
```

The results of sending 1000 packets of 1000B through this network show that it took `8.208 s` to complete the simulation. If both links are upgrades to 1Gbps, it would only take `0.208008 s` to transmit the packets.][]

For each packet it takes: $\frac{1000B}{1000Mbps} = \frac{8000b}{1000000bps} = .008s$
For all 1000 packets, it takes: $1000 * .008s = 8.00s$ And for the first packet, it takes $100ms + 100ms = 200ms = .2s$ to reach the 3rd node, plus $.008s$ to be propagated.
Therefore, the simulation is correct since $8.00s + .2s + .008s = 8.208s$

3.2 Second Trial

In this trial, we configure the two links in the network with heterogeneous speeds to mimic a more probable real-world scenario:

```
1 n1  n2  1Mbps    100ms
2 n2  n3  256Kbps  100ms
```

The results of sending 1000 packets of 1000B through this network show that it took `31.458 s` to complete the simulation.

4 Queuing Theory

We can validate that the simulator in fact can mimic real-world situations where a queue is involved. The following code and data will illustrate this.

4.1 Approach

We begin by showing the high-level code used to run this experiment. Notice the `util_rates` list which represents the experimental queue utilization rates. We loop through each of these utilization rates and run the simulator as if the queue were being utilized at these percentages.

After the data has been collected in `util_rates_plot` and `times` lists, it is then plotted against a curve representing a theoretical queuing theory formula.

```
util_rates = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.98]
util_rates_plot = []
times = []
for rate in util_rates:
  iter, util_rate, time = run_delay_simulation(rate)
  util_rates_plot.append(util_rate)
  times.append(time/iter)

p = Plotter()
p.linePlotData(util_rates_plot,times,'Utilization Rate','Average Queueing Time')
```

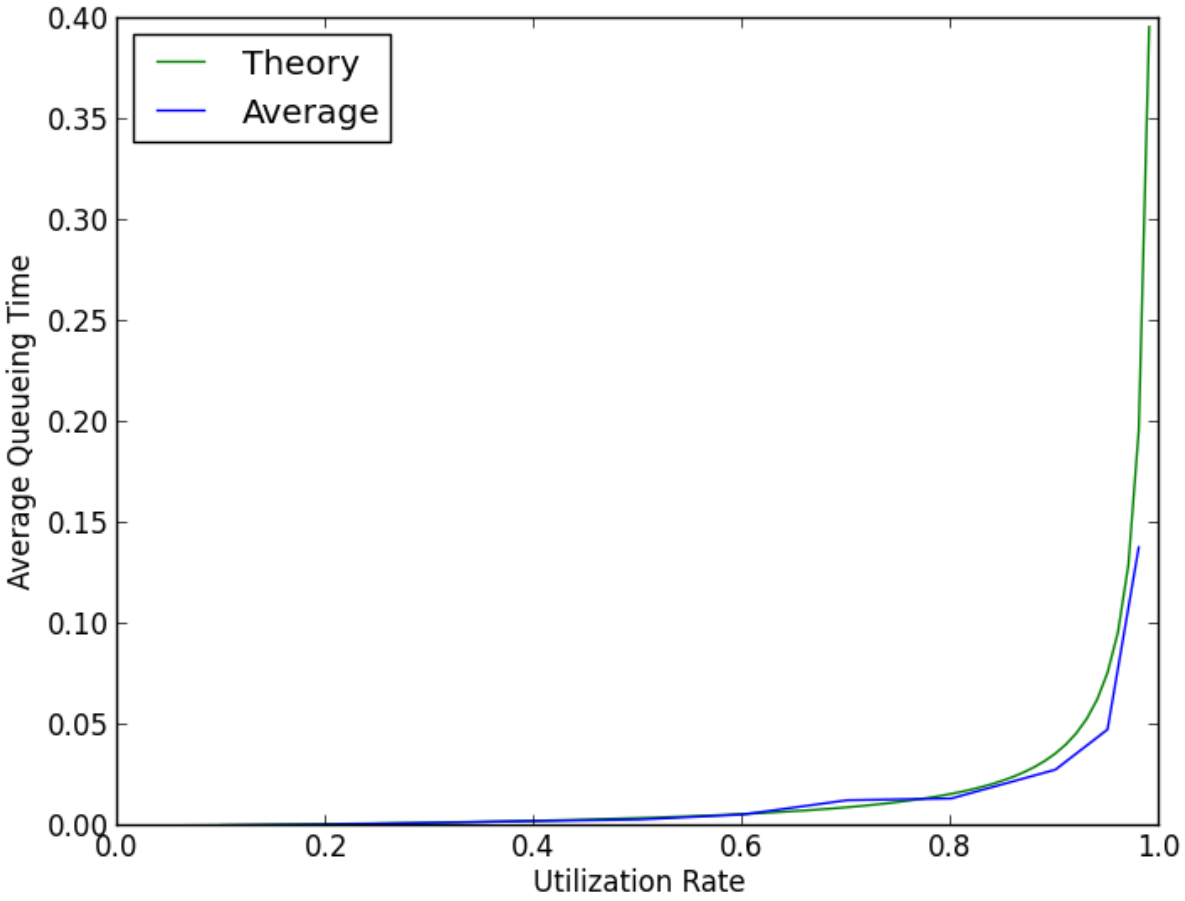
The following code creates the theoretical and actual plots on the same chart. Notice the #theoretical section where the theoretical formula is used to graph an equation. After that, the actual experimental values are plotted.

```
def linePlotData(self,xdata, ydata, xname, yname):
    """ Create a line graph. """
    clf()

    #theoretical
    rho = np.arange(0,1,1.0/100)
    mu = 1000000.0 / (1000.0 * 8.0)
    plot(rho,(1/(2*mu))*(rho/(1-rho)),label='Theory',color="green")

    #actual
    plot(xdata, ydata,label="Average",color="blue")

    legend(loc='upper left')

    #theoretical
    xlabel(xname)
    ylabel(yname)
    savefig('line.png')
```

Additionally, in order to set up the simulation correctly, variables were created for the max rate that packets would be transmitted as well as a load percentage on the node.

4.2 Results

Below is a graphic of the theoretical and actual averages of the experiment. Notice that the Utilization Rate is plotted against Average Queuing Time. AQT is measured by dividing the sum total of the time each packet was in queue by the number of packets.



We see that our experimental results mimic the theoretical curve closely. Since this is the case, we can conclude that the simulator does in fact mimic real-world scenarios and is fit for use to demonstrate similar scenarios.