1 Summary

In this report, I give the results of running different programs using the `bene` simulator. The first, with two nodes, highlights the difference in timing for propagation and transmission delay. The second, with three nodes, highlights how queuing delay works together with transmission and propagation delays. The last uses two nodes and exponential delay to highlight how fractional load on a network interacts with queuing delay.

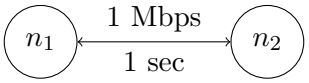
I managed to have some difficulty in getting the simulator to work as I wanted to. A bidirectional two-node link was simple to set up, but getting a unidirectional three-node link to forward properly to the third node was much more difficult than expected.

I made many quality of life improvements to the library, including some static typing. The semantics of forwarding was also modified to allow for more logical usage. I also made a very useful `NetHelper` class, which managed many of the more complicated simulation operations, such as sending packets to other nodes or creating forwarding entries.

2 Two Nodes

Calculating the theoretical arrival time would be $\frac{8L}{R} + p$, where L is the length of the message in bytes, R is the bandwidth of the link in bits/second, and p is the propagation delay of the link.

1. **Network Configuration:**

   Output:

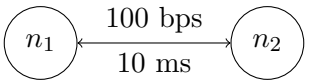
   ```
   0.000 0 1.008
   ```

   Calculation:

   $$\frac{8L}{R} + p = \frac{8000\,\text{bit}}{1\,\text{Mbit/ sec}} + 1\,\text{sec} = 1.008\,\text{sec}$$

   Clearly, the propagation delay is the primary cause of the delay in this scenario.

2. **Network Configuration:**

   Output:

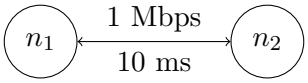
   ```
   0.000 0 80.010
   ```

Calculation:

$$\frac{8L}{R} + p = \frac{8000\,\text{bit}}{100\,\text{bit/sec}} + 10\,\text{ms} = 80.010\,\text{sec}$$

In this case, 80 seconds of delay is caused by the abysmally slow bandwidth.

3. **Network Configuration:**



Output:

```
0.000 0 0.018
0.000 1 0.026
0.000 2 0.034
2.000 3 2.018
```

Calculation:

For the first packet:

$$\frac{8L}{R} + p = \frac{8000\,\text{bit}}{1\,\text{Mbit/sec}} + 10\,\text{ms} = 18\,\text{ms}$$

.

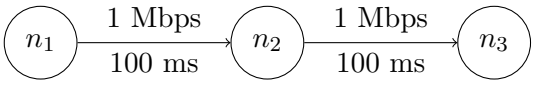
After that, the next two packets simply add the transmission delay (8 ms). Then, the fourth packet, delayed after 2 seconds, will have the same 18 ms to reach the end as the first packet.

3 Three Nodes

For three nodes, I define the transmission delays $t_1 = \frac{8L}{R_1}$ and $t_2 = \frac{8L}{R_2}$, and the propagation delays p_1 and p_2, and $p = p_1 + p_2$. Assuming that packets are sent one after another, and $t_1 \leq t_2$, packet n will arrive at the third node after $t_1 + nt_2 + p$.

1. **Two Fast Links**

Network Configuration:



Output:

```
7.960 995 8.176
7.968 996 8.184
7.976 997 8.192
7.984 998 8.200
7.992 999 8.208
```

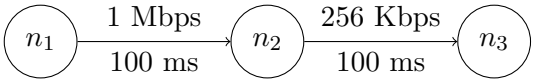
Calculation:

$$t_1 + nt_2 + p = \frac{8000\,\text{bit}}{1\,\text{Mbit/sec}} + 1000\frac{8000\,\text{bit}}{1\,\text{Mbit/sec}} + 200\,\text{ms} = 8.208\,\text{sec}$$

For a single packet, the propagation delay dominates. However, over the entire file, the transmission delay of 8 seconds is the largest factor.

2. **One Fast Link, One Slow Link**

Network Configuration:



Output:

```
7.960 995 31.333
7.968 996 31.364
7.976 997 31.396
7.984 998 31.427
7.992 999 31.458
```

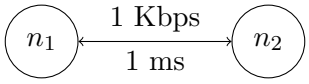
Calculation:

$$t_1 + nt_2 + p = \frac{8000\,\text{bit}}{1\,\text{Mbit/sec}} + 1000\frac{8000\,\text{bit}}{256\,\text{kbit/sec}} + 200\,\text{ms} = 31.458\,\text{sec}$$

For a single packet, the propagation delay dominates. Over the entire file, the majority of time the packets wait is in the queue of n_2.

4 Queuing Theory

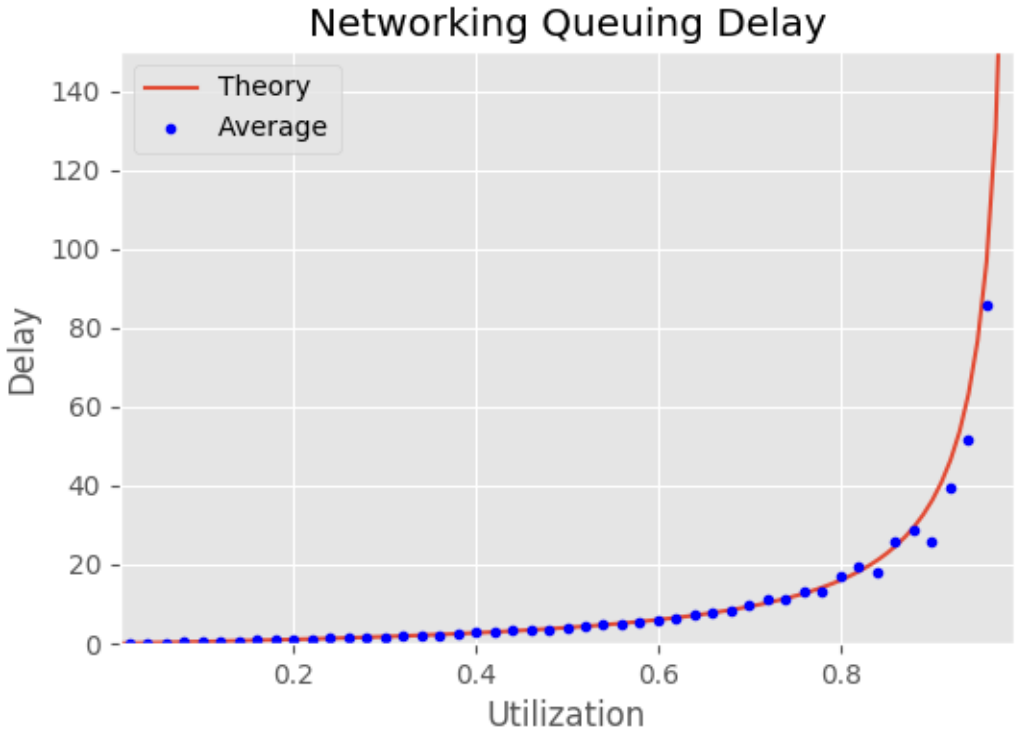
Network Configuration:



Load Settings:

I simulated continually sending 1000-byte packets at varying loads, for 100,000 seconds. For each of these, I outputted the queuing delay and link load of each item into a CSV file.

Results:



I used `pandas` to load the data points, average the queuing delay for each utilization, and plot it. Along with the results, I also plotted the theoretical queuing delay:

$$\frac{1}{2\mu} \cdot \frac{\rho}{1-\rho}$$

The similarity indicates that the simulator is working as intended.

5 Conclusion

In conclusion, the simulator does work, even if some finagling had to be done to get it to work as I had expected. The theoretical results and simulated results match very closely.

I already had experience with Python, `matplotlib`, and LaTeX. I spent more time than expected investigating how routing is supposed to work between nodes. I made modifications to make it so that an address of a destination node is the address of a link where it is an endpoint. This made automatic route calculation much simpler, and now for future simulations, usage of the library is much more succinct.

Overall, this was an interesting lab, and I learned quite a bit, including using `pandas`. I also had to think quite a bit on how to calculate three-node delays with different link speeds. I'm looking forward to working on the next lab.