1 Simulation Setup:

I utilized the provided FIFO queue sample project as the starting point for my simulation.

1.1 Network Configuration

My "Tandem Satellite" network consisted of one generator, two FIFO queues/servers, a satellite node, and a sink node. The generator, queue, and sink nodes are all the same as given in the FIFO sample (and the same as Project #1). I defined the satellite node as a simple node which handled any incoming message by forwarding it to the following node immediately. Due to the distance to the satellite, the simulation needed to account for the propagation delay.

$$t_{prop} = \frac{d}{c} \qquad d = 42,000km \qquad c = 299,792,458\frac{m}{s}$$

$$t_{prop} = \frac{4.2 \times 10^7 m}{299792458\frac{m}{s}}$$

$$\boxed{t_{prop} = 0.1401s}$$

The satellite node accounted for the propagation delay of communication with the satellite by using a channel delay on the links connecting the satellite to the two queues. Figure #1 shows this network setup.

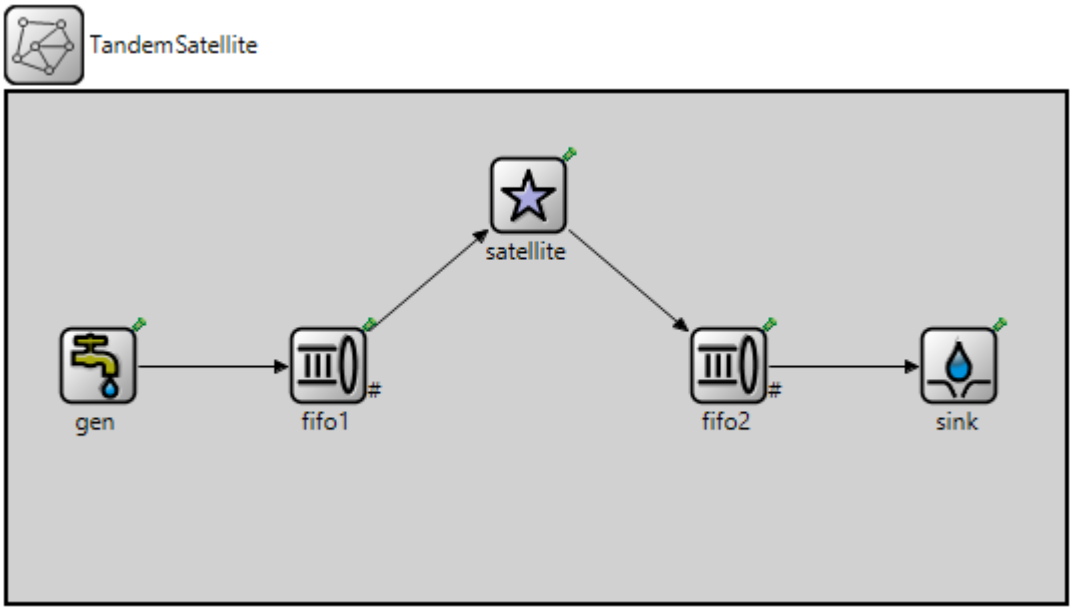

Figure 1: Network diagram

When a packet arrives at the sink, it generates a packet delay data point as defined in Eq. #1

$$Packet\,Delay = time_{arrival} - time_{created} \tag{1}$$

1.2 Simulation Time:

Before determining a simulation run-time, I calculated λ, μ, and ρ from the mean service time and interarrival times specified. Table #1 below.

Case #:	**Interarrival Time (s)**	λ	μ	ρ
1	1.00	1.00	2.00	0.50
2	0.52	1.92	2.00	0.96
3	0.50	2.00	2.00	1.00

Table 1: Queue Parameters

Given enough time, Cases 1 and 2 will tend to an average value, as discussed in section 2.1. Case #3 should become unstable, growing its queue indefinitely. I started by running the three cases for 1 hour each. However, when I plotted the data, the second case with $t_{IA} = 0.52\,seconds$ actually had a longer average system delay than the final case with $t_{IA} = 0.50\,seconds$, and similar average queue lengths. Based on this result, I decided that a longer simulation was needed to allow Cases 1 and 2 to achieve stability. My final simulation run-time was 10 hours, which generated a result aligning with the expectation given the relationship between λ and μ.

2 Results & Analysis:

2.1 System Delay

2.1.1 Expected Results

Using λ and μ, and the calculated values for ρ, one can calculate the expected time in the system, $E[r]$; and the expected time in the queue $E[w]$, using the relationships below, derived from Little's Law.

$$E[r]_{system} = \frac{1}{\mu - \lambda} \qquad E[w]_{queue} = \frac{\rho}{\mu - \lambda}$$

However, the above equations do not account for duplicate queues, or propagation delay. Thus, the actual relationship for the tandem queue system requires some additional manipulation, shown below:

$$\begin{aligned} E[r]_{actual} &= 2 \cdot E[r] + 2 \cdot t_{prop} \\ &= 2 \cdot E[r] + 0.28\,\text{seconds} \end{aligned} \qquad E[w]_{actual} = 2 \cdot E[w]\,\text{seconds}$$

Table #2 shows these results for each of the three cases. Case 3 has an infinite expectation because it is an unstable system with $\rho = 1$.

Case #	$\mathbf{E[r]}_{actual}$	$\mathbf{E[w]}_{actual}$
1	2.28	1.00
2	26.28	24.00
3	∞	∞

Table 2: Expected System and Queue Delay

2.1.2 Simulation Results

Figure #2 shows the system delay of each case.

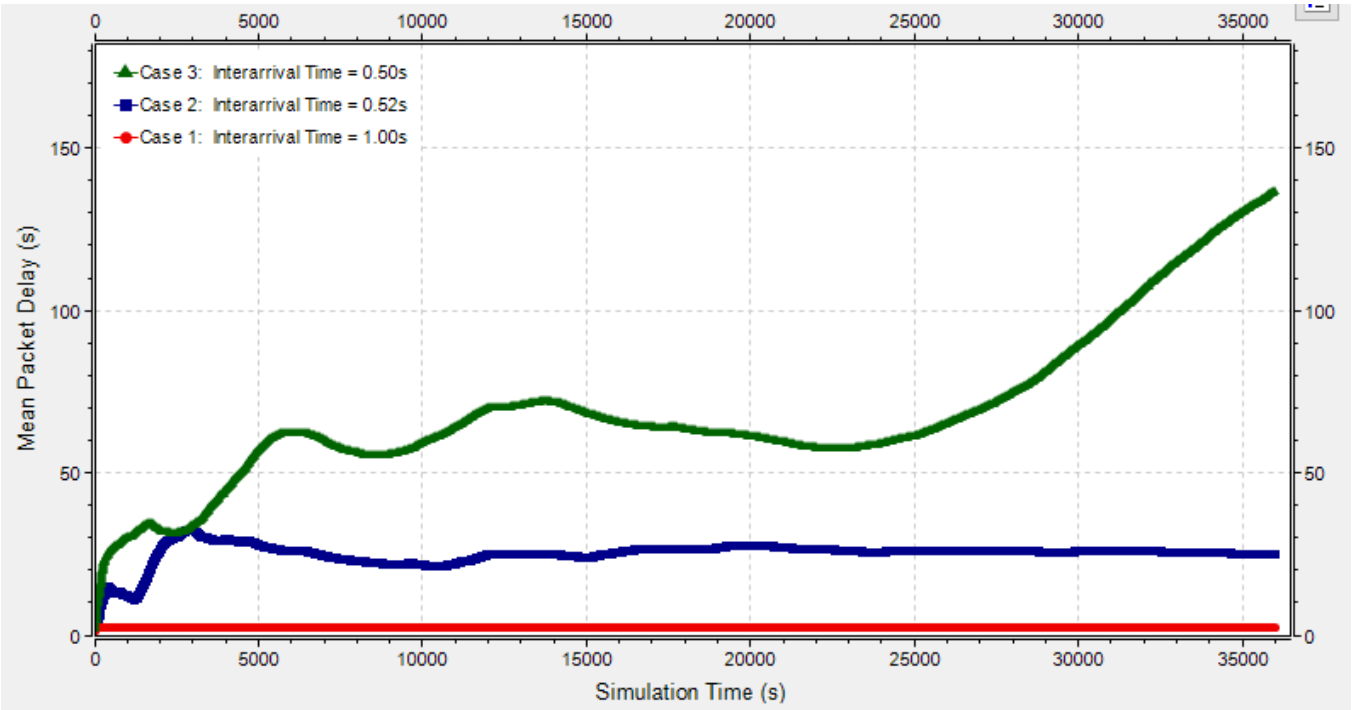

Figure 2: System Delay of varied interarrival times

This simulation matched the expected system delays calculated previously, with averages of 2.24 seconds, 24.71 seconds, and 136.75 seconds, respectively for Cases 1-3.

Case #	Mean Packet Delay (s)
1	2.24
2	24.71
3	136.75

Table 3: Simulated Mean Packet Delay

Case #1 had a 1.8% difference from the expected value, while Case #2 had a 6.0% difference. Despite only having a difference of $0.02s$ in interarrival times, the average delay of Cases 2 and 3 were significantly different. The mean delay for Case #3 was 136.75 seconds. This occurs because when $\rho = 1$, the system is unstable and the queue will grow to indefinitely.

2.2 Queue Length & Queue Time

2.2.1 Expected Results

Again using Little's Law, one can calculate the expected number in the queue N_Q. Note, although there are two separate FIFO queues, they both have the same expected queue length and time.

$$N_Q = \frac{\rho^2}{1-\rho}$$

Case #	N_Q
1	0.50
2	23.04
3	∞

Table 4: Expected Queue Length

The expected queue time was detailed in the Expected Results for the System Delay

2.2.2 Simulation Results

As shown in Tables 2 & 3, the average queue lengths and queue times increased relative to the system delay. Case #1 had a low number of packets in the queue because its arrival rate $\lambda_1 = 0.5 \cdot \mu$.

Module	**Case #**	**Mean (packets)**	**Std. Dev**
fifo1	1	1.51	1.53
fifo2	1	1.46	1.46
fifo1	2	22.95	22.35
fifo2	2	25.20	22.68
fifo1	3	111.81	82.53
fifo2	3	165.06	206.42

Table 5: Queue Length

Module	**Case #**	**Mean (s)**	**Std. Dev**
fifo1	1	0.50	0.87
fifo2	1	0.47	0.82
fifo1	2	11.15	11.36
fifo2	2	12.27	11.46
fifo1	3	54.92	40.91
fifo2	3	80.52	101.61

Table 6: Queue Time

These results matched the expected results fairly well. For Case 1, there was a mean queue length of 1.49, compared to an expected queue length of 0.5, so a 1 packet difference. Case 2 had an average queue length of 24.08, compared to an expected queue length of 23.04. Finally, Case 3 had an average queue length of 138.44.
For the queue time, one would expect the expected waiting time calculated in the system delay to equal the sum of the queue time in fifo1 and fifo2. For Case 1, there was an average **total** queue time of 0.97 seconds, compared to 1.00 seconds. Case 2 had an average total queue time of 23.42 seconds, compared to the expected queue time of 24.00 seconds. Finally, Case 3 had an average total queue time of 135.44 seconds.

2.3 Propagation Delay

One can see the propagation delay of the simulation by comparing the average lifetime to the sum of the queue times and service times.

$$\text{Prop Delay} = \text{Mean lifetime} - time_{fifo1} - time_{fifo2} - 2 \cdot \text{service time}$$

In practice, the mean times produced the following calculated propagation delays:

Case #	**Prop. Delay (s)**
1	0.27
2	0.29
3	0.31

Table 7: Calculated Propagation Delays

3 Conclusions

This project demonstrated the long term effects of a queuing system for different values of ρ. When λ is much less than μ, the system is able to keep up, and produces minimal delays. As λ approaches μ, the queue will increase in length; however, as long as $\lambda < \mu$, the system will still be stable. Once $\lambda = \mu$, the system becomes unstable, and will have a system delay and queue length that will grow indefinitely.

Another interesting outcome of this experiment was seeing the effects of the propagation delay. One would expect the two queues to have identical times in the long run; however, they were slightly different for all simulations. Because of these variations, using the average queue times and average lifetime to calculate the propagation delay produced a slightly different value than the constant specified in the simulation, as shown in Table #7. This shows the inherent complexity of a system of queues, even when the queues have the same parameters for expected arrival rates and service rates.

Appendix A: Figures

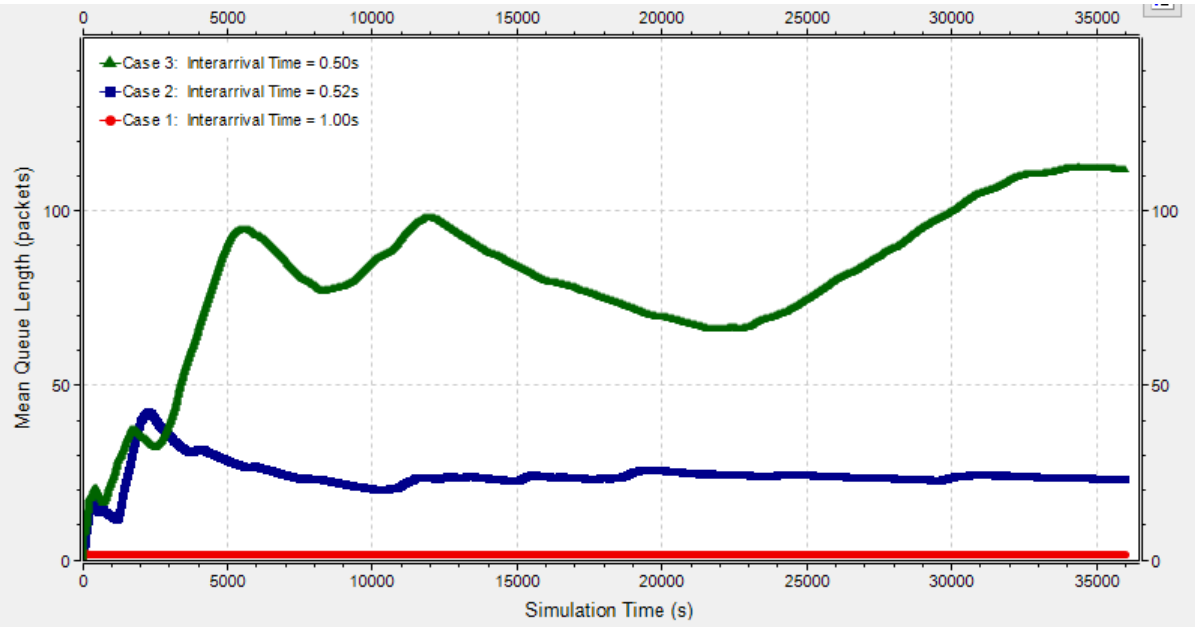

Figure 3: Fifo1 Mean Queue Length

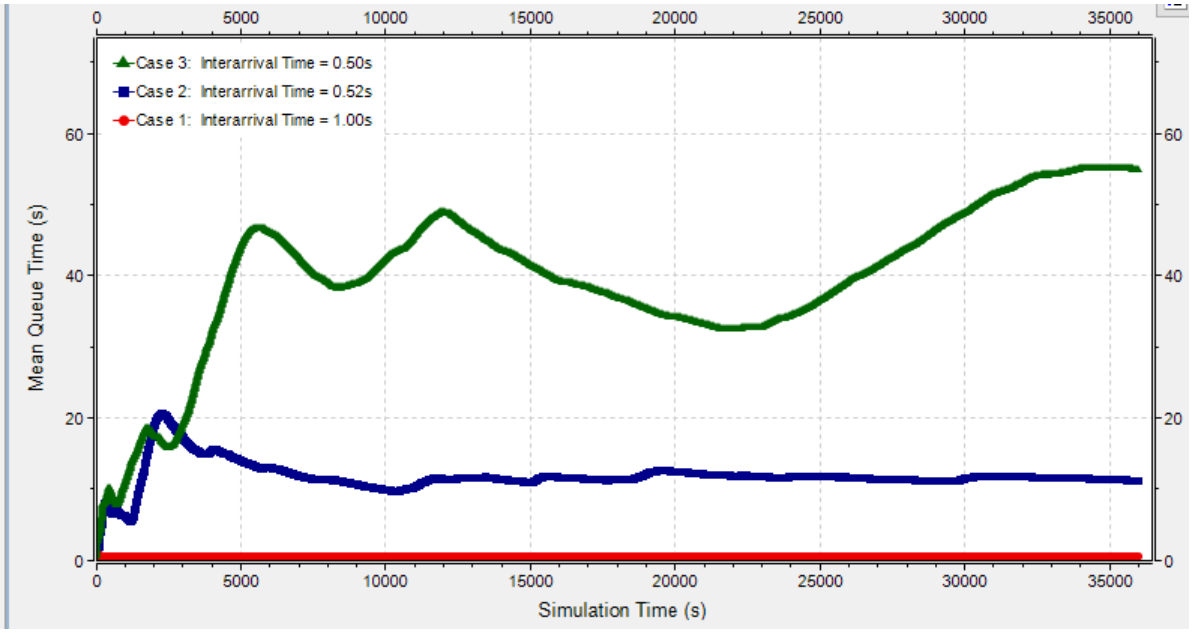

Figure 4: Fifo1 Mean Queue Time

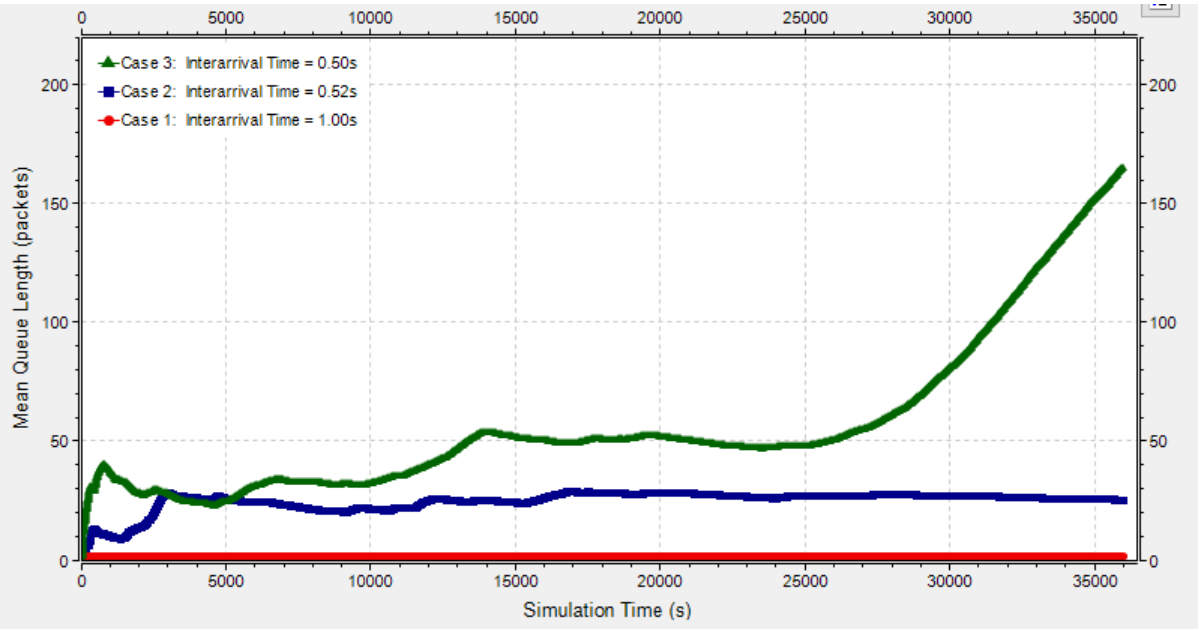

Figure 5: Fifo2 Mean Queue Length

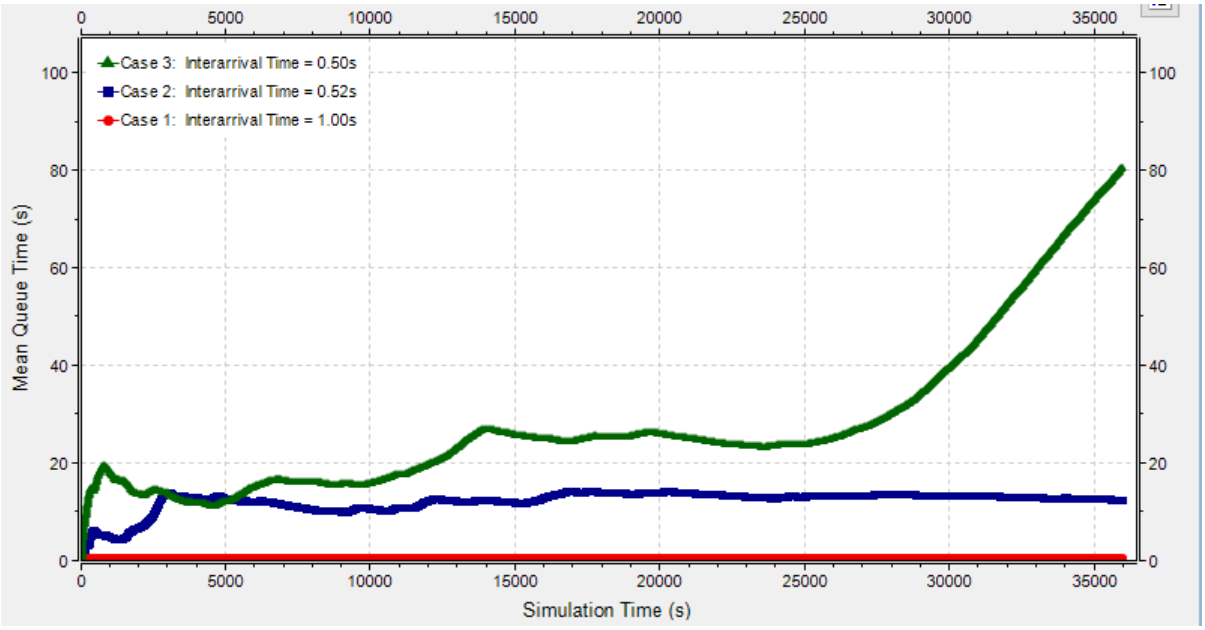

Figure 6: Fifo2 Mean Queue Time

Appendix B: Simulation Files

```
[General]
network = TandemSatellite
sim-time-limit = 10h
cpu-time-limit = 300s
#debug-on-errors = true
#record-eventlog = true

[Config TandemQueue]
**.gen.sendIaTime = exponential( ${m=1.0, 0.52, 0.50}s )
**.fifo*.serviceTime = exponential(0.5s)
**.satellite.propDelay = 0.1401s

[Config Tandem1]
description = "Arrival rate of 1s"
**.gen.sendIaTime = exponential(1s)
**.fifo*.serviceTime = exponential(0.5s)
**.satellite.propDelay = 0.1401s

[Config Tandem2]
description = "Arrival rate of 0.52s"
**.gen.sendIaTime = exponential(0.52 s)
**.fifo*.serviceTime = exponential(0.5s)
**.satellite.propDelay = 0.1401s

[Config Tandem3]
description = "Arrival rate of 0.5s"
**.gen.sendIaTime = exponential(0.5 s)
**.fifo*.serviceTime = exponential(0.5s)
**.satellite.propDelay = 0.1401s
```

Simulation Initialization File - omnetpp.ini

```
network TandemSatellite
{
    @display("bgb=528,254");
    submodules:
        gen: Source {
            parameters:
                @display("p=45,136");
        }
        fifo1: Fifo {
            parameters:
                @display("p=160,136");
        }
        fifo2: Fifo {
            parameters:
                @display("p=360,136");
        }
        satellite: Satellite {
            parameters:
                @display("p=260,51");
        }
        sink: Sink {
            parameters:
                @display("p=475,136");
        }
    connections:
        gen.out --> fifo1.in;
        fifo1.out --> { delay = satellite.propDelay; } --> satellite.in;
        satellite.out --> { delay = satellite.propDelay; } --> fifo2.in;
        fifo2.out --> sink.in;
}
```

Simulation Setup - Tandem_Satellite.ned