Omnet Simulation Setup:

I utilized the provided FIFO queue sample project as my starting point. I needed to make the following changes to ensure I had a working simulation for the task:

1. **FifoNet.ned**

   This file defines the "FifoNet" setup. I created three identical queues consisting of a source node, a FIFO server, and a sink node, in which all traffic flows through the three nodes; this setup is shown in Figure #1.

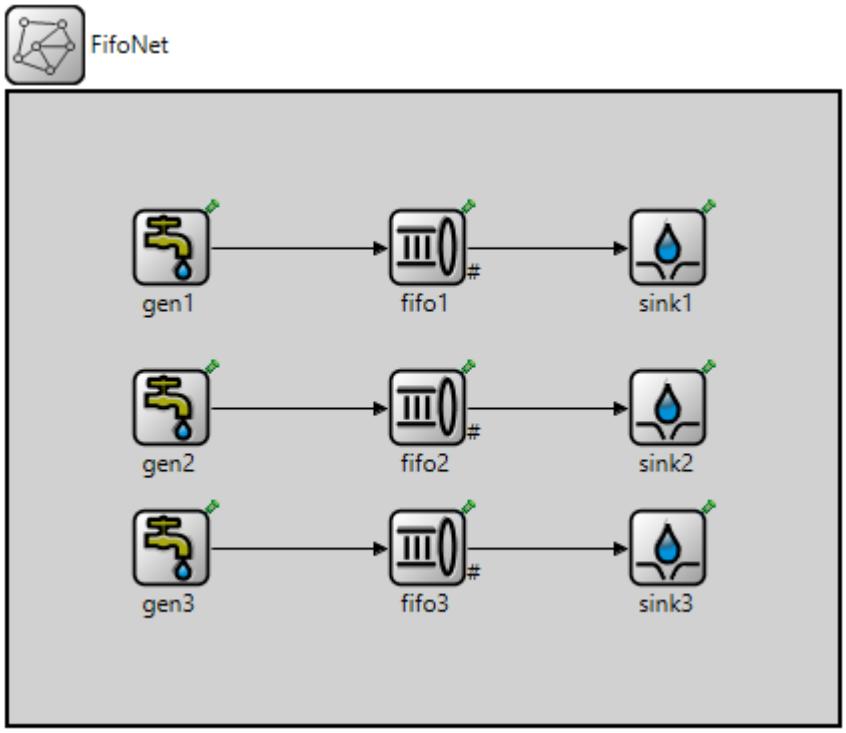

Figure 1: Diagram showing the three queue setup.

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

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

2. **omnetpp.ini**

   The modifications to this file specified each queue's individual parameters. Each queue had a **service time** of $t_s = 0.75\,seconds$. I differentiated the arrival rates of each queue by specifying the interarrival time from each generator/source. These times are shown below:

Queue #:	Interarrival Time (seconds)
1	1.0
2	0.50
3	0.25

Table 1: Interarrival Times

The final change was to set the **sim-time-limit** to one hour.

Results & Analysis:

Utilization:

Given the interarrival times and service times of each queue, I calculated values for λ, μ, and ρ for each queue using the below relationships:

$$\lambda = \frac{1}{Interarrival\,Time} \qquad \mu = \frac{1}{Service\,Time} \qquad \rho = \frac{\lambda}{\mu}$$

Note, each queue has a service time of 0.75 seconds, and the interarrival times stated in Table #1

Queue #:	λ	μ	ρ
1	1.00	$1.\bar{3}\bar{3}$	0.75
2	2.00	$1.\bar{3}\bar{3}$	1.50
3	4.00	$1.\bar{3}\bar{3}$	3.00

Table 2: Utilization

From these results, I would expect Queue #1 to have a negligible delay. Because the equation used to calculate ρ breaks down with $\rho > 1$, the calculated utilization of Queues #2 and #3 is unattainable. I would expect Queues #2 and #3 to be fully utilized, with $\rho_{effective} = 1$.[1] However, Queues #2 and #3 will have a queue length which will grow to infinity. This makes sense given the meaning of each variable: for Queue #1 - packets arrive slower than they are serviced, for the other two queues - packets arrive faster than they can be serviced.
The utilization is directly related to the number of packets which the queue processes. This quantity, E[X], can be found analytically, using the equation below.

IF$\rho < 1$

$$E[X] = \frac{\text{simtime}}{\lambda}$$

IF$\rho \geq 1$

$$E[X] = \frac{\text{simtime}}{\mu}$$

Queue #:	ρ	**E[X]**
1	0.75	3600
2	1.50	4800
3	3.00	4800

Table 3: Expected Packets Processed

[1] $\rho_{effective}$ is the actual utilization of the server, calculated as $1 - p_{idle}|_{p_{idle}=0} = 1$

System Delay

Expected Results

Because the queue has no variation in service time or interarrival rate (for a given trial), the expected system delay can be found by analyzing a sequence of arrivals because they will have the same relationships as an arithmetic sequence. I will walk through this analysis for each of the three cases:

Case 1: Interarrival Time = 1s

Packet #:	Arrival Time (s)	Finish Time (s)	Expected Delay
1	0	0.75	0.75
2	1	1.75	0.75
3	2	2.75	0.75

Table 4: Expected Packet Delay with $\lambda = 1\frac{packets}{second}$

From this data, the following relationships can be established for packet n:

$$\begin{aligned} \text{Packet Delay}_n &= 0.75s \\ \text{Finish Time}_n &= (n-1) + 0.75s \end{aligned}$$

Case 2: Interarrival Time = 0.50s

Packet #:	Arrival Time (s)	Finish Time (s)	Expected Delay
1	0	0.75	0.75
2	0.5	1.50	1.00
3	1.0	2.25	1.25

Table 5: Expected Packet Delay with $\lambda = 2\frac{packets}{second}$

The following equation thus defines the finish time and packet delay for packet n:

$$\begin{aligned} \text{Finish Time}_n &= n \cdot 0.75 \\ \text{Packet Delay}_n &= 0.75 + (n-1) \cdot 0.25 \end{aligned}$$

Thus, given the expectation of servicing 4800 packets, the final packet would have the following finish time and delay:

$$\begin{aligned} \text{Finish Time}_{4800} &= 4800 \cdot 0.75 \\ &= 3600s \\ \text{Packet Delay}_{4800} &= 0.75 + (4799) \cdot 0.25 \\ &= 1200.5s \end{aligned}$$

Because the delays grow linearly, the mean delay will be the average of the first and last packet's delays:

$$\begin{aligned} \text{Average Delay}_{Case\,2} &= \frac{0.75 + 1200.5}{2} \\ &= 600.625s \end{aligned}$$

Case 3: Interarrival Time = 0.25s

Packet #:	Arrival Time (s)	Finish Time (s)	Expected Delay
1	0	0.75	0.75
2	0.25	1.50	1.25
3	0.50	2.25	1.75

Table 6: Expected Packet Delay with $\lambda = 4\frac{packets}{second}$

The following equations thus define the finish time and packet delay for packet n:

$$\text{Finish Time}_n = n \cdot 0.75$$
$$\text{Packet Delay}_n = 0.75 + (n-1) \cdot 0.50$$

Thus, given the expectation of servicing 4800 packets, the final packet would have the following finish time and delay:

$$\begin{aligned}\text{Finish Time}_{4800} &= 4800 \cdot 0.75 \\ &= 3600s \\ \text{Delay}_{4800} &= 0.75 + (4799) \cdot 0.50 \\ &= 2400.25s\end{aligned}$$

Because the delays grow linearly, the mean delay will be the average of the first and last packet's delays:

$$\begin{aligned}\text{Average Delay}_{Case\,3} &= \frac{0.75 + 2400.25}{2} \\ &= 1200.50s\end{aligned}$$

Simulation Results

Figure #2 shows the instantaneous system delay of each queue. Packets in Queue #1 experienced Packet Delay = $0.75s$, while the packet delay steadily increased for packets in Queues 2 and 3.

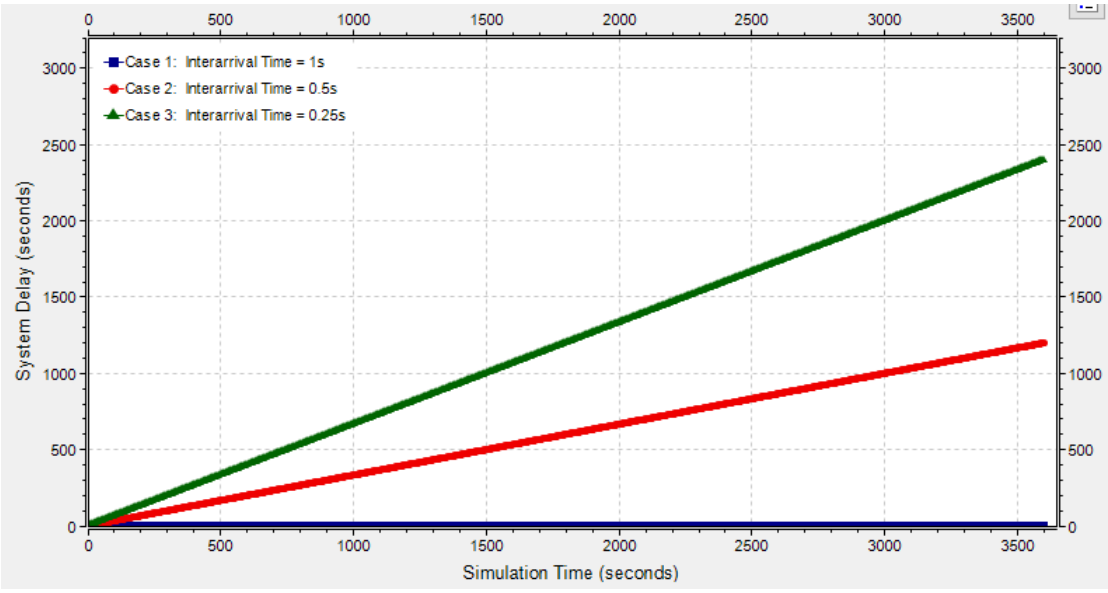

Figure 2: Instantaneous System Delay

Table #7 shows the simulation data for the packet delay for each queue.

Queue #:	Average Packet Delay (s)	Std. Dev
1	0.75	0.0
2	600.63	346.45
3	1200.50	692.89

Table 7: Average Packet Delays

These results matched the expected results given analysis of the arrival and service rates. The system delays for Queue #2 and Queue #3 increased linearly as the queue length increased. The standard deviation indicates how quickly the average packet delay increases for each case.
Queue #1 serviced each packet which arrived, servicing a total of 3600 packets. Despite the differences in arrival rates between Queue #2 and #3, they both serviced 4800 packets. One can see the relationship between the arrival rates, service rates, and throughput from these results. When each packet that enters the system is serviced, as in Queue #1, $throughput = \lambda$; when the queue grows infinitely, as in Queues #2 and #3, $throughput = \mu$.

Queue Length & Queue Time

Expected Results

Because the arrival rates and service times are constant, the queue will grow at the following rate:

$$\text{Inst. Queue Length} = t \cdot (\frac{1}{\lambda} - \frac{1}{\mu}) \tag{2}$$

$$\text{Mean Queue Length} = \frac{1}{2} \cdot (\text{Inst. Queue Length}_{final} - 0) = \frac{1}{2} \cdot \text{Inst. Queue Length}_{final} \tag{3}$$

Using Equation #'s 2 and 3, I calculated the final queue length and mean queue length for each queue, given a final time of 3600 seconds.[2]

Queue #	Final Queue Length (packets)	Mean Queue Length (packets)
1	0	0
2	2400	1200
3	9600	4800

Table 8: Expected Queue Lengths

The queue time is directly related to the queue length, at the time of a packets arrival. Section details the delay of the final packet serviced in each queue, which would each have a queue time = packet delay $- 0.75s$. The mean queue time should be half of the queue time of the final packet.

Simulation Results

Tables #3 and #4 below show the raw data for the queue length and queue time for each queue.[3]

Queue #	Count	Mean	Std. Dev
1	1	0.0	0.00
2	12001	1200.0	692.91
3	19201	4800.3	2771.50

Table 9: Queue Length

Queue #	Count	Mean	Std. Dev
1	3601	0.0	0.00
2	4801	600.0	346.52
3	4801	1200.0	693.04

Table 10: Queue Time

The data shown above is indicative of the system delays experienced by each packet. When the server can keep up with/stay ahead of the queue, arriving packets have no queuing delay: there is no queue! However, once the queue forms, if the arrival rate remains faster than the service rate, the queue simply continues to grow, causing increasing system delays for each packet. These results match the expected results from the prior analysis.

[2] If $\frac{1}{\lambda} - \frac{1}{\mu} < 0$, this would indicate a shortening of the queue. For this project, this result indicates the queue length is always 0.
[3] The count seems increased by 1, but is caused by the packets arriving at the end of the simulation, that had not been serviced.

Conclusions

This project demonstrated the effects of using constant arrival and service times, which leads to constant arrival and service rates. Because any queue length, queue time, or packet delay increases follow a linear pattern, they can be analyzed using arithmetic relationships. The simulation results directly matched those gained through analytic methods.

As long as the service rate is faster than the arrival rate, the system will never queue. However, with arrival rates faster than the service rate, the system will become unstable, with the queue growing indefinitely. The queue lengths of the unstable systems grow with respect to their arrival rates, with a corresponding increase in system delay as the queue length grows. For any systems which always maintain a queue (such as #s 2 and 3), their servers will be fully utilized. Thus, for a system with constant rates, the optimal configuration would be with $\mu = \lambda \rightarrow \rho = 1$, which maintains full utilization without a queue.

Appendix A: Graphs of Queue Length and Time

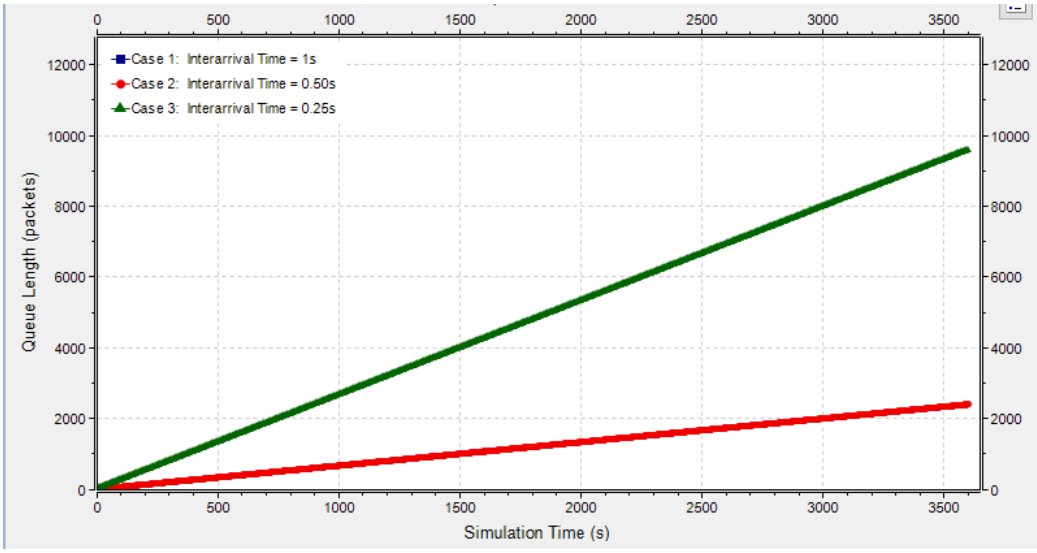

Figure 3: Instantaneous Queue Length

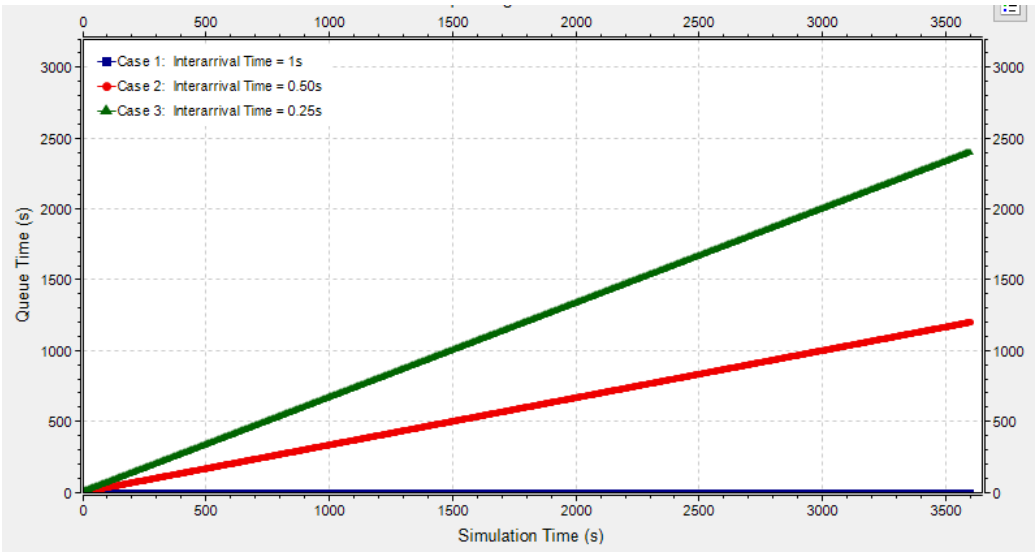

Figure 4: Inst. Queue Time

Appendix B: Edited Files

```
// FifoNet.ned:
network FifoNet
{
    submodules:
        gen1: Source {
            parameters:
                @display("p=81,77");
        }
        gen2: Source {
            parameters:
                @display("p=81,157");
        }
        gen3: Source {
            parameters:
                @display("p=81,227");
        }
        fifo1: Fifo {
            parameters:
                @display("p=209,77");
        }
        fifo2: Fifo {
            parameters:
                @display("p=209,157");
        }
        fifo3: Fifo {
            parameters:
                @display("p=209,227");
        }
        sink1: Sink {
            parameters:
                @display("p=329,77");
        }
        sink2: Sink {
            parameters:
                @display("p=329,157");
        }
        sink3: Sink {
            parameters:
                @display("p=329,227");
        }
    connections:
        gen1.out --> fifo1.in;
        fifo1.out --> sink1.in;

        gen2.out --> fifo2.in;
        fifo2.out --> sink2.in;

        gen3.out --> fifo3.in;
        fifo3.out --> sink3.in;
}
```

Defining the network in FifoNet.ned

```
// omnetpp.ini
[General]
description = "3 Seperate Arrival times, same service times"
network = FifoNet
sim-time-limit = 1h
cpu-time-limit = 300s
#debug-on-errors = true
#record-eventlog = true
**.gen1.sendIaTime = 1s
**.fifo1.serviceTime = 0.75s

**.gen2.sendIaTime = 0.50s
**.fifo2.serviceTime = 0.75s

**.gen3.sendIaTime = 0.25s
**.fifo3.serviceTime = 0.75s
```

Omnetpp.ini edited to run the three queues with separate parameters