

UNIVERSITY OF VICTORIA

# AIRLINE CHECK-IN COUNTER SIMULATION

---

**DISHA GARG (V00847833)**



**DEPARTMENT OF COMPUTER SCIENCE**

**CSC-546**

**12/9/2015**

## Table of Contents

Project Description.....	2
Problem Mapping .....	2
Simulation Goals & Parameters .....	2
Methodology.....	3
Tool Used .....	3
M/M/1 Modelling .....	4
Simulation set-up .....	4
Collecting the Statistics: .....	7
Analysis: .....	7
M/G/1 Modelling .....	12
Setting up Simulation:.....	12
Collecting the Statistics:.....	13
Analysis: .....	14
Test for Random Number Generator .....	18
Chi-Square Test For uniformity:.....	19
Extended Queuing (Priority Queue).....	20
Analysis: .....	21
Conclusion.....	27

## Project Description

A Queuing Model is a suitable model to represent a service oriented problem where customers arrive randomly to receive some services.

The objective of this project is to model an Airline Counter Check-in scenario from the real world with simulating and analysing the effect of priority queues on the boarding procedures for economy, business and first class passengers. We are assuming here that the system here is having an infinite capacity and the passengers are assumed to be 0.89 million (i.e., 890,000). This project has been divided into three parts to compare all their possible behaviours that affect the performance of the system. The first part here is to analyse a scenario where the arrival rate of the passengers is Poisson distributed and the service time given at the security check terminal is exponentially distributed. This type of queue where a single line exists for both types of passengers is known as the basic M/M/1 queue. The second part however, models the service time with the uniform distribution provided at the counter check-in. This type of queue is M/G/1 queue. Finally in the last part of this report, we analysed the model as a priority queue with the three queues which are all provided with a different priority to first, economic and business class passengers with a check-in counter as a server.

## Problem Mapping

In the first part, which involves an M/M/1 queue, both the inter-arrival time and the service time are expressed as an exponential distribution. However, in the second part uniform distribution is used to express the service time. And in the last part, the percentages of the passengers going in the three parallel queues of first class passengers, economic and business class are assigned as 15%, 25% and 60% respectively. All the queues in this model are have been assigned an infinite capacity. These three models will have 10 replications each with the same parameters but different random number seeds and for the 5 specified server utilization as = 0.1, 0.2, 0.4, 0.6 and 0.8.

## Simulation Goals & Parameters

The fundamental goal of this project is to simulate the functional behaviour of the Airline check-in counter in order to determine and analyse the various performance metrics. The reason behind modelling these three scenarios is to determine the performance of the system under different distribution types and queues. Moreover in the extended part, we will examine the effect of the priority queues on the check-in service and will analyse the time that the passengers spend in the line and in the whole system using a one and two lines.

Following summarized are the parameters which we will be evaluating in this project:

Metric	Definition
$L$	Long-run time-average number of customers in the system.
$L_Q$	Long-run time-average number of customers in queue.
$w$	Long-run average-time spent in system per customer.
$w_Q$	Long-run average-time spent in queue per customer.
$\rho$	Server utilization.

Table 1: Metrics and their definitions

## Methodology

### Tool Used

The tool which we are using to simulate this project is OMNeT++ (version 4.6). The OMNeT++ IDE comes with several projects that showcase the types of simulations that can be built using OMNeT++. An OMNeT++ model consists of modules that communicate with message passing.

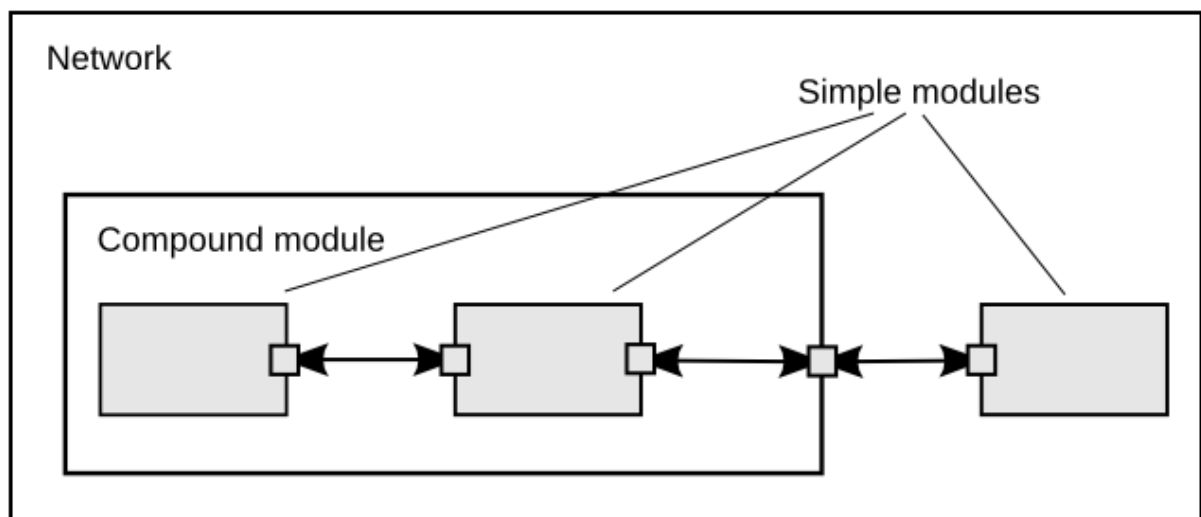







Figure 1: The figure shows Simple and Compound modules.

The functionality can be added by creating NED and ini files for creating and configuring models respectively. The metrics can be collected by running batch executions, and then analysing simulation results.

Following are the four simple modules which are used in OMNeT++ to model our M/M/1, M/G/1 and 3-Queue Priority system:

 <p>Source</p>	<p>It is a module that generates jobs. We can specify the number of jobs to be generated.</p> <p>Note: Inter-arrival time is emitted here.</p>
---	--

 Queue	It is a queue with a built-in server.  Note: System Length ( $L$ ), Queue Length ( $L_Q$ ) and Server Utilization ( $\rho$ ) are emitted here.
 Sink	It destroys (or optionally keep) the packets and collects statistics.  Note: Total System Time ( $w$ ), Total Queue Time ( $w_Q$ ) and Service Time are emitted here.
 Classifier	It sends the packets to different outputs depending on messages type or priority.
 Connection	All these blocks are connected through these unidirectional arrows.

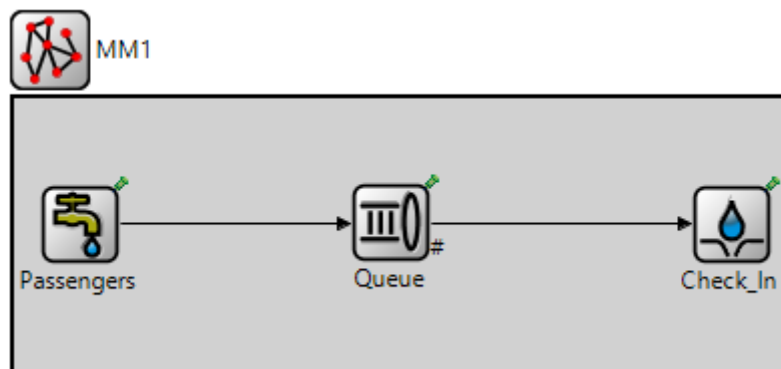
*Table 2: Simple Modules*

## M/M/1 Modelling

### Simulation set-up

In order to set up the simulation we need to create the NED, ini, .cc, .h files.

(a) NED File: The structure of a simulation model is described in the NED language. NED stands for Network Description. It lets the user declare simple modules, and connect and assemble them into compound modules.



*Figure 2: Diagram of an NED file with M/M/1 model.*

```

import org.omnetpp.queueing.Queue;
import org.omnetpp.queueing.Sink;
import org.omnetpp.queueing.Source;
import org.omnetpp.queueing.Server;

//
// This simple queueing network only contains a source, a FIFO queue and a sink.
//
network MM1
{
    parameters:
        @display("i=block/network2");
    submodules:
        Check_In: Sink {
            @display("p=358,63");
        }
        Passengers: Source {
            @display("p=33,63");
        }
        Queue: Queue {
            @display("p=188,62");
        }
    connections:
        Passengers.out --> Queue.in++;
        Queue.out --> Check_In.in++;
}

```

Figure 3: Source code in MM1.ned file

The diagram shown in figure 2 is designed with .ned (Network Description) code as shown in figure 3. For M/M/1, we made three blocks: Passengers, Queue and Check\_In with their position parameter defined and the connections are made between these blocks. So, the generated passengers at the Source (Passengers) are sent to Queue where they wait for their service to be executed and if the server is idle, then the passengers can get served from the server in the Queue block and then once completed, the passengers can be send to Sink (Check\_In) and released from the Check-in Counter system.

(b) ini File: Configuration and input data for the simulation are in a configuration file usually called omnetpp.ini.

The **repeat()** function is used to get metrics for server utilization  $\rho = 0.1, 0.2, 0.4, 0.6$  and  $0.8$  with 10 different seeds. And to get these specified values for server utilization, we need to keep a constant service time and a variable inter-arrival time as 10 s, 5 s, 2.5 s, 1.67 s and 1.25 s.

```

[Config MM1]
description = "a single M/M/1 queue"
network = MM1
**.numJobs = 890000
**.interArrivalTime = exponential(${exp= 10, 5, 2.5, 1.67, 1.25}s)
**.serviceTime = exponential(1s)
**.capacity = -1
repeat=10

```

Figure 4: Source code in omnetpp.ini file

Now, in order to make a connection of our model to the in-built package (i.e., queueinglib) containing all the definitions of basic blocks like Source, Sink, Queue and PassiveQueue etc. The configurations specified as above are included under [General] section, by which we give the path of “queueinglib” in our model.

```
[General]
ned-path = ../../queueinglib
```

Figure 5: [General] section of omnetpp.ini file

We can also make a direct reference to the already made project as:

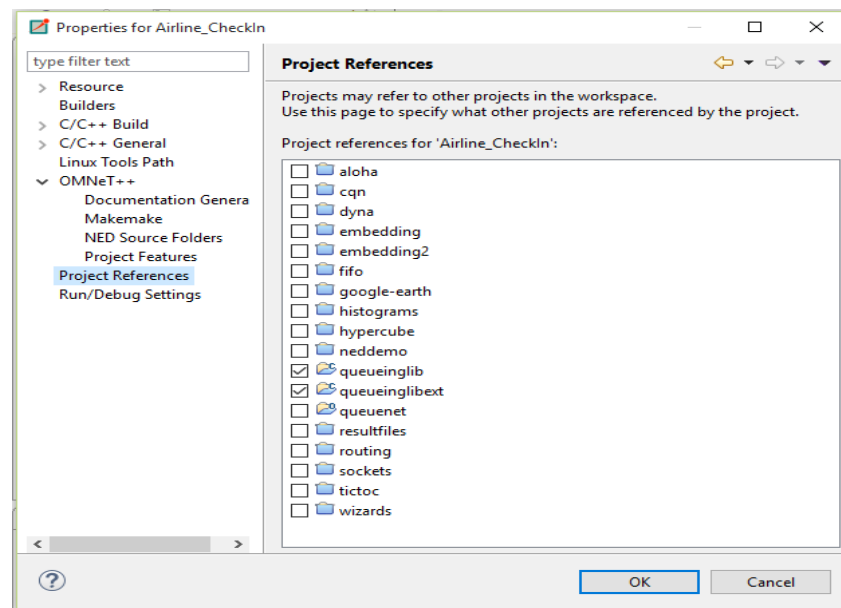


Figure 6: Selection of Project References

The simulations are made faster by running the processes in parallel as shown in figure 7.

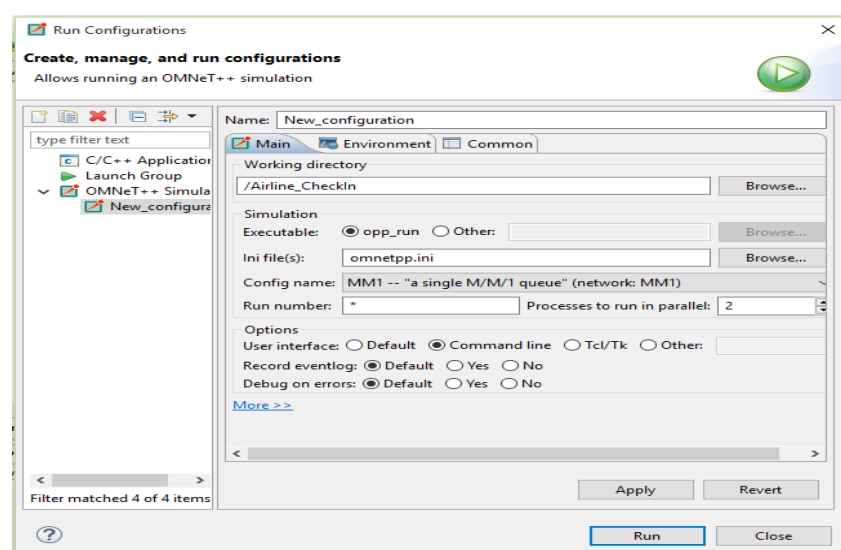


Figure 7: Running configurations

### Collecting the Statistics:

We can find the statistical data of all the parameters from *Analysis* (\*.anf) file under the scalars tab. Data can be easily exported into csv formats and their mean values can be compared with the theoretical values.

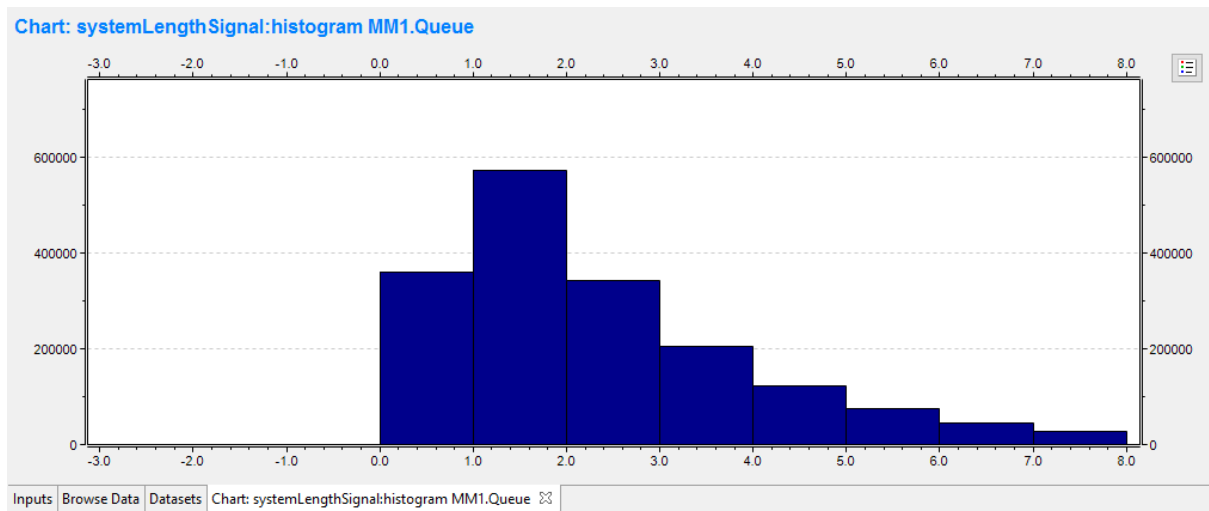
The theoretical values are calculated using the formulae of M/M/1 as shown in Table 3.

$L = \frac{\lambda}{\mu - \lambda} = \frac{\rho}{1 - \rho}$	$L_Q = \frac{\lambda^2}{\mu(\mu - \lambda)} = \frac{\rho^2}{1 - \rho}$
$w = \frac{1}{\mu - \lambda} = \frac{1}{\mu(1 - \rho)}$	$w_Q = \frac{\lambda}{\mu(\mu - \lambda)} = \frac{\rho}{\mu(1 - \rho)}$
$\bar{Y} \pm t_{\alpha/2, n-1} S / \sqrt{n}$	$L = \lambda w \quad \rho = \lambda / \mu$

Table 3: Formulae to calculate the theoretical values of performance parameters

### Analysis:

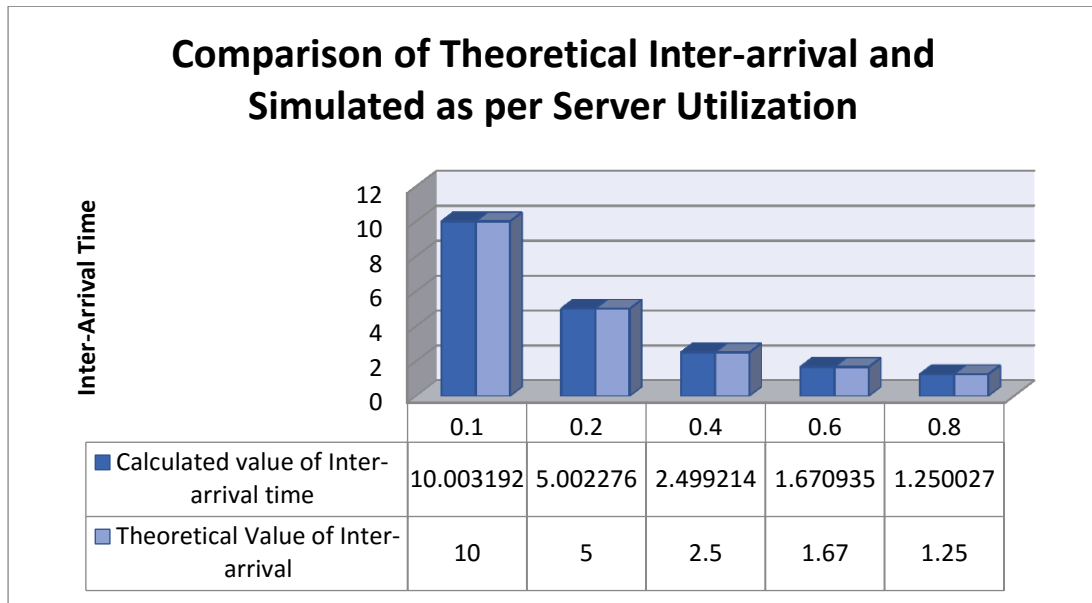
#### System State Histogram for Utilization of 0.6:



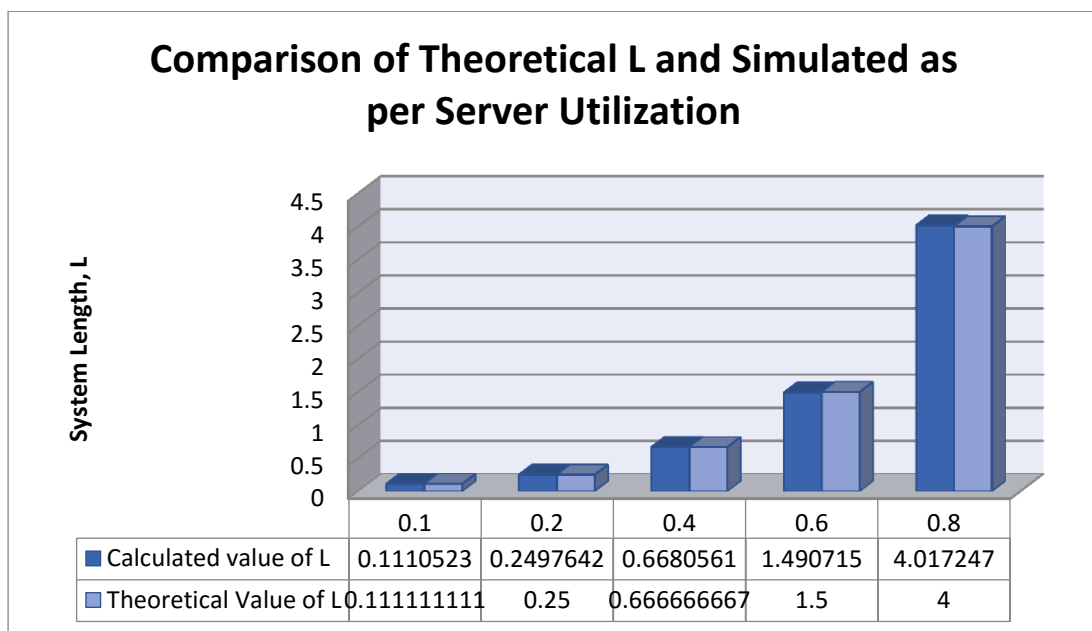
Histogram 1: System state histogram of M/M/1 for Server Utilization =0.6



*Comparison of Theoretical and simulated Parameters as per Server utilization:*

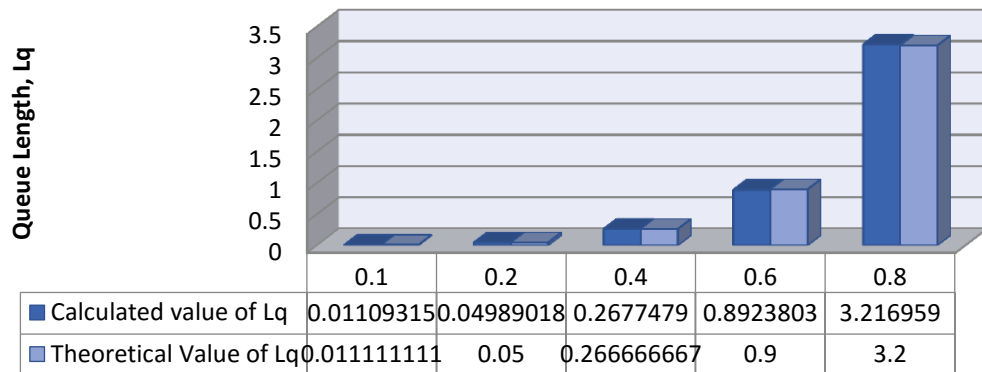


*Graph 1: Comparison of Theoretical Inter-arrival and Simulated as per Server Utilization*



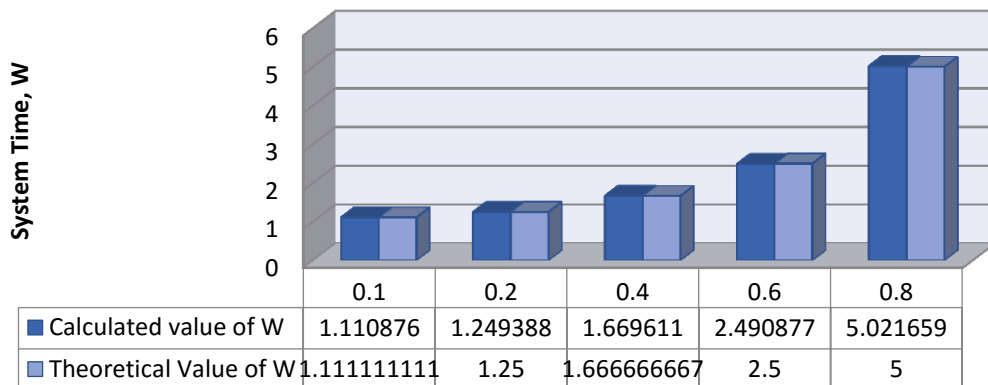
*Graph 2: Comparison of Theoretical L and Simulated L as per Server Utilization*

### Comparison of Theoretical Lq and Simulated as per Server Utilization



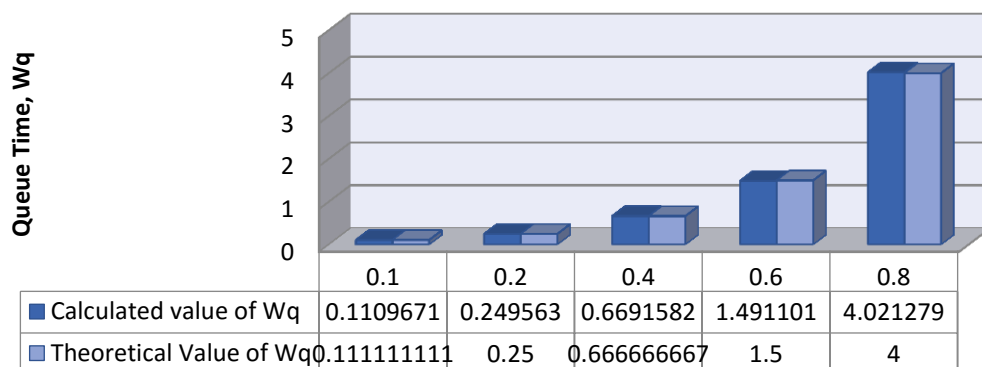
Graph 3: Comparison of Theoretical Lq and Simulated Lq as per Server Utilization

### Comparison of Theoretical W and Simulated as per Server Utilization



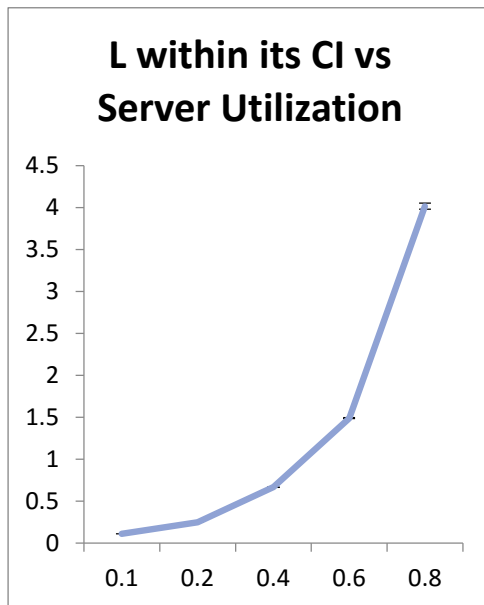
Graph 4: Comparison of Theoretical W and Simulated as per Server Utilization

### Comparison of Theoretical Wq and Simulated as per Server Utilization

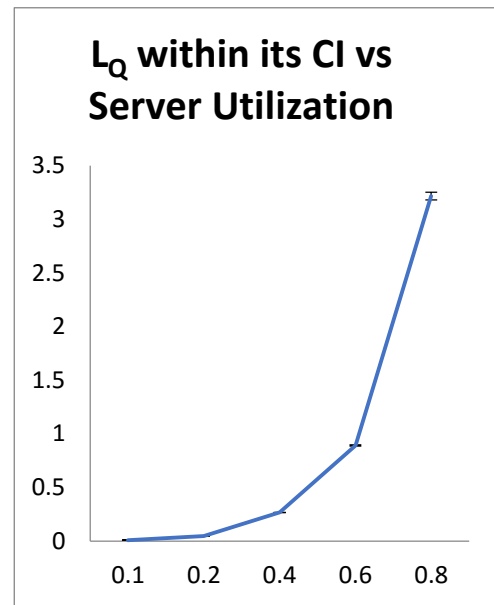


Graph 5: Comparison of Theoretical Wq and Simulated as per Server Utilization

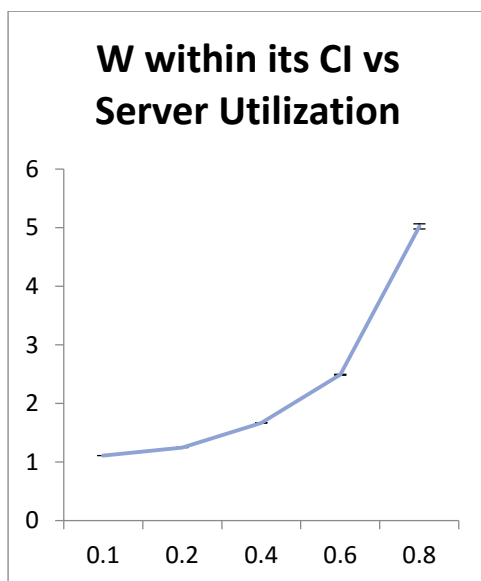
*Plotting Server Utilization versus Parameters within CI (Confidence Intervals):*



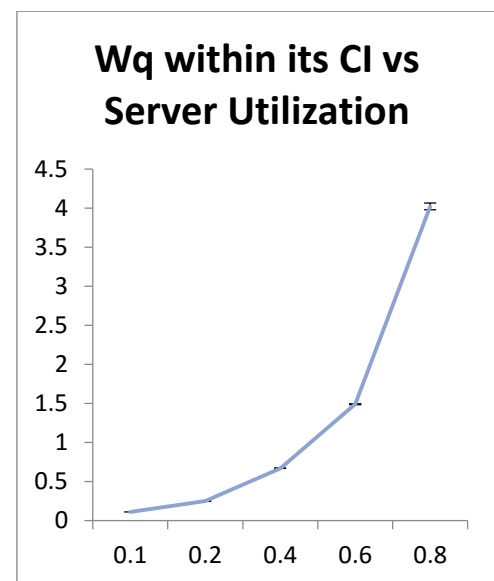
*Graph 6: L versus rho*



*Graph 7:  $L_Q$  versus rho*



*Graph 8: W versus rho*



*Graph 9:  $W_q$  versus rho*

*Data Analysis for M/M/1:*

Here we will be computing the Confidence Intervals for all the Statistics collected from the M/M/1 model. The Confidence Intervals with the Actual and Theoretical data can be viewed in the table below.

When Inter-arrival time is 10	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	0.1110523	0.111111111	0.000142229	0.111194529	0.110910071
	Lq	0.01109315	0.011111111	5.81536E-05	0.011151304	0.011034996
	W	1.110876	1.111111111	0.001225255	1.112101255	1.109650745
	Wq	0.1109671	0.111111111	0.000582491	0.111549591	0.110384609
	Rho	0.09995892	0.1	0.000103864	0.100062784	0.099855056
When Inter-arrival time is 5	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	0.2497642	0.25	0.00045395	0.25021815	0.24931025
	Lq	0.04989018	0.05	0.000265659	0.050155839	0.049624521
	W	1.249388	1.25	0.001335598	1.250723598	1.248052402
	Wq	0.249563	0.25	0.001158717	0.250721717	0.248404283
	Rho	0.1998738	0.2	0.000232276	0.200106076	0.199641524
When Inter-arrival time is 2.5	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	0.6680561	0.666666667	0.001286543	0.669342643	0.666769557
	Lq	0.26774790	0.266666667	0.000905657	0.268653557	0.266842243
	W	1.669611	1.666666667	0.002429601	1.672040601	1.667181399
	Wq	0.6691582	0.666666667	0.002036352	0.671194552	0.667121848
	Rho	0.4003073	0.4	0.000510286	0.400817586	0.399797014
When Inter-arrival time is 1.67	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	1.490715	1.5	0.005614257	1.496329257	1.485100743
	Lq	0.89238030	0.9	0.004873851	0.897254151	0.887506449
	W	2.490877	2.5	0.007588553	2.498465553	2.483288447
	Wq	1.491101	1.5	0.007086872	1.498187872	1.484014128
	Rho	0.5983325	0.6	0.000819992	0.599152492	0.597512508
When Inter-arrival time is 1.25	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	4.017247	4	0.036506326	4.053753326	3.980740674
	Lq	3.21695900	3.2	0.035504639	3.252463639	3.181454361
	W	5.021659	5	0.044044633	5.065703633	4.977614367
	Wq	4.021279	4	0.043106638	4.064385638	3.978172362
	Rho	0.8002829	0.8	0.001124376	0.801407276	0.799158524

Table 4: Data Analysis of the M/M/1 Parameters

## M/G/1 Modelling

### Setting up Simulation:

In contrast to M/M/1 modelling, in the M/G/1 model, the service times have General Distribution which can be uniform, triangular, weibull etc. Here, we have chosen a uniformly distributed service time. Here, Source (Passengers) is generating the jobs (passengers) with Poisson distributed inter-arrival and these passengers are independently generated.

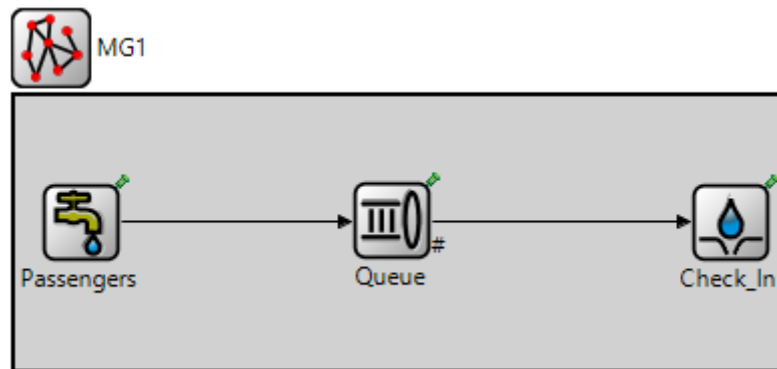


Figure 8: Diagram of an NED file with M/G/1 model.

```
import org.omnetpp.queueing.Queue;
import org.omnetpp.queueing.Sink;
import org.omnetpp.queueing.Source;
import org.omnetpp.queueing.Server;
//
// This simple queueing network only contains a source, a FIFO queue and a sink.
//
network MG1
{
    parameters:
        @display("i=block/network2");
    submodules:
        Check_In: Sink {
            @display("p=357,63");
        }
        Passengers: Source {
            @display("p=33,63");
        }
        Queue: Queue {
            @display("p=188,62");
        }
    connections:
        Passengers.out --> Queue.in++;
        Queue.out --> Check_In.in++;
}
```

Figure 9: Source code in MG1.ned file

For M/G/1 also, we made three building blocks: Passengers, Queue and Check-In with the connections in between.

These passengers are served with the service rate with mean,  $\mu = 1$  sec. The inter-arrival rate is denoted by  $\lambda$ . The statistics are collected for server utilization  $\rho = 0.1, 0.2, 0.4, 0.6$  and  $0.8$  with 10 different seeds that are generated using the **repeat()** function in omnetpp.ini file as shown in figure 10.

```
[Config MG1]
description = "a single M/G/1 queue"
network = MG1
**.numJobs = 890000
**.interArrivalTime = exponential($exp= 10, 5, 2.5, 1.67, 1.25}s)
**.serviceTime = uniform(0s,2s)
**.capacity = -1
repeat=10
```

Figure 10: Source Code in omnetpp.ini file

For M/G/1 execution, we need to write the above shown configuration in the omnetpp.ini file. We run the M/G/1 for 890000 jobs which are generated with the different mean inter-arrival time for each run as 10 sec, 5 sec, 2.5 sec, 1.67 sec and 1.25 sec. All the jobs are run for 10 different seeds for each value of utilization.

#### Collecting the Statistics:

All the statistical data for  $L$ ,  $L_q$ ,  $W$ ,  $W_q$  are collected through *Analysis* (.anf) file under scalar tab. These data are then exported to csv formats and their mean values are calculated for comparison with the theoretical values.

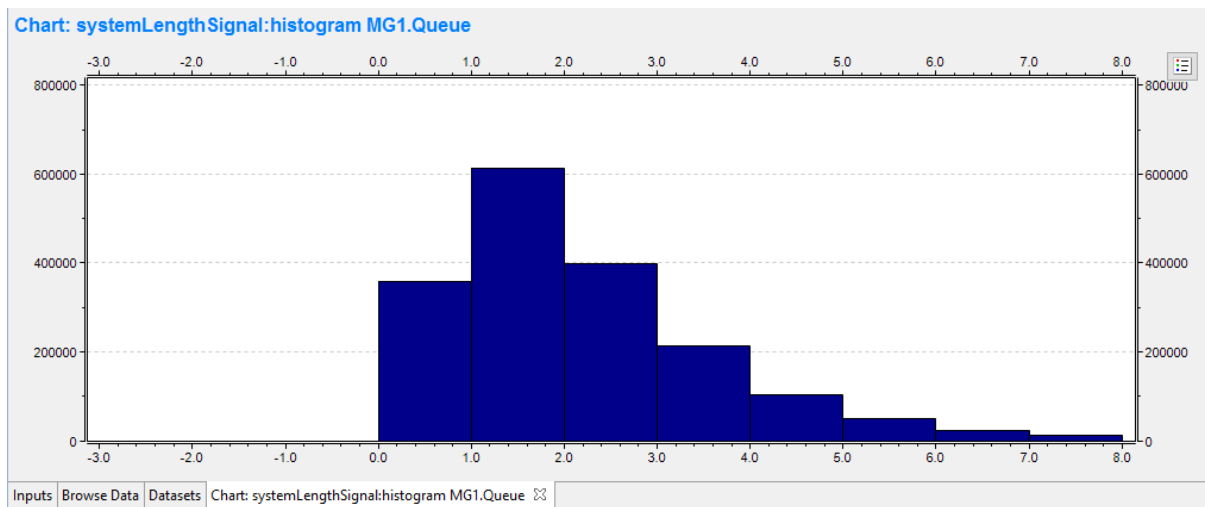
The theoretical values are calculated using the formulae of M/M/1 as shown in Table 5.

$L = \rho + \frac{\lambda^2(1/\mu^2 + \sigma^2)}{2(1-\rho)} = \rho + \frac{\rho^2(1 + \sigma^2\mu^2)}{2(1-\rho)}$	$L_q = \frac{\rho^2(1 + \sigma^2\mu^2)}{2(1-\rho)}$
$w = \frac{1}{\mu} + \frac{\lambda(1/\mu^2 + \sigma^2)}{2(1-\rho)}$	$w_q = \frac{\lambda(1/\mu^2 + \sigma^2)}{2(1-\rho)}$
$\bar{Y} \pm t_{\alpha/2, n-1} S / \sqrt{n}$	$\rho = \lambda / \mu$

Table 5: Formulae to calculate the theoretical values of performance parameters

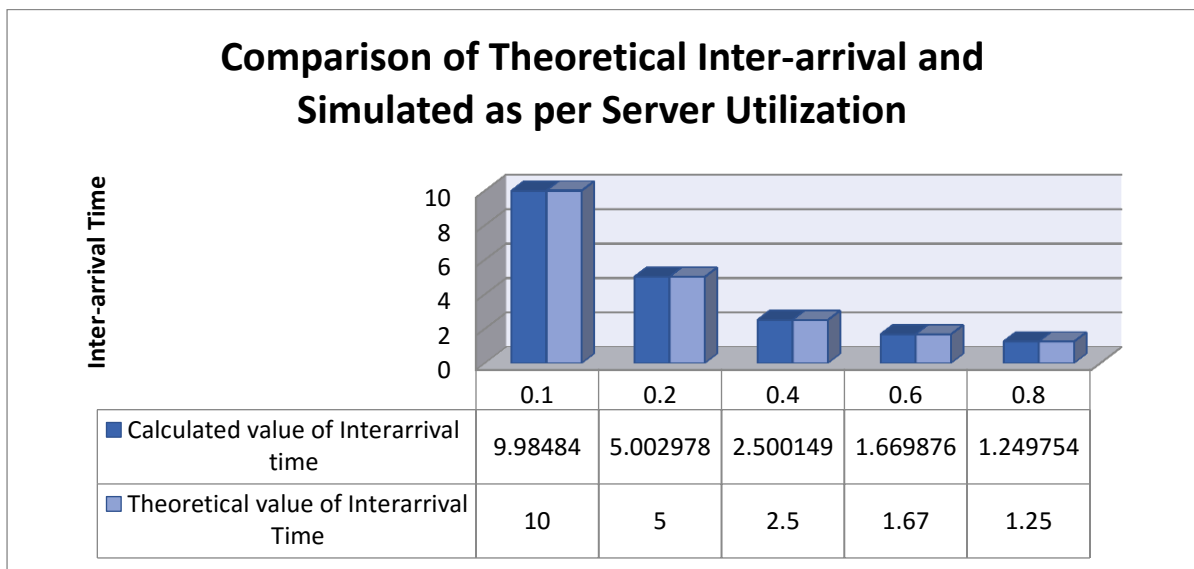
## Analysis:

### System State Histogram for Utilization of 0.6:

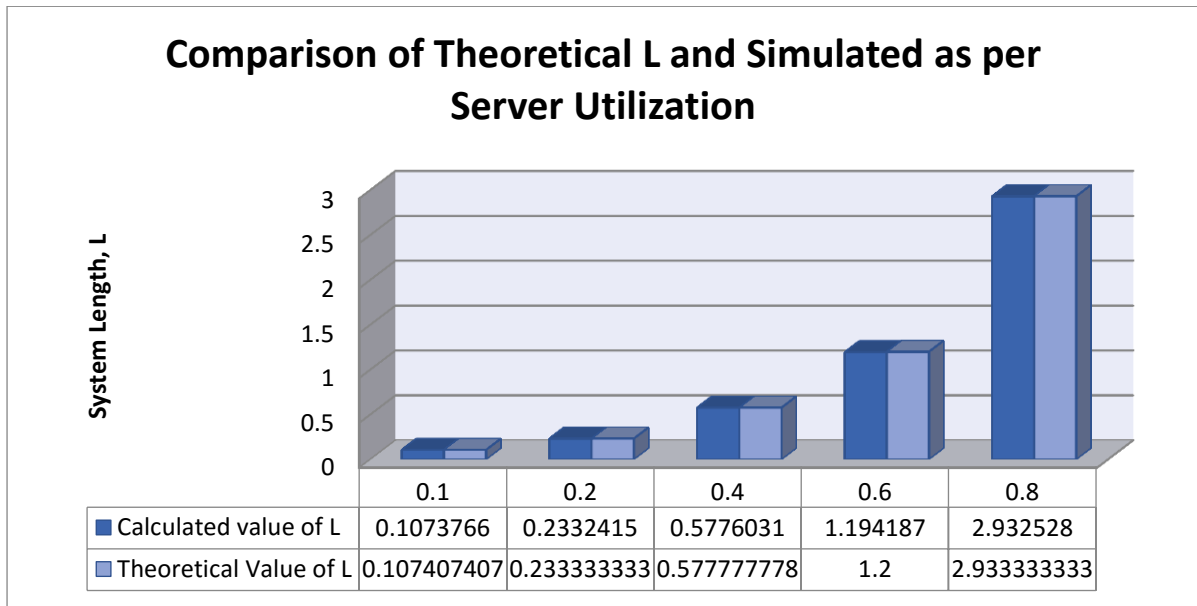


Histogram 2: System state histogram of M/G/1 for Server Utilization =0.6

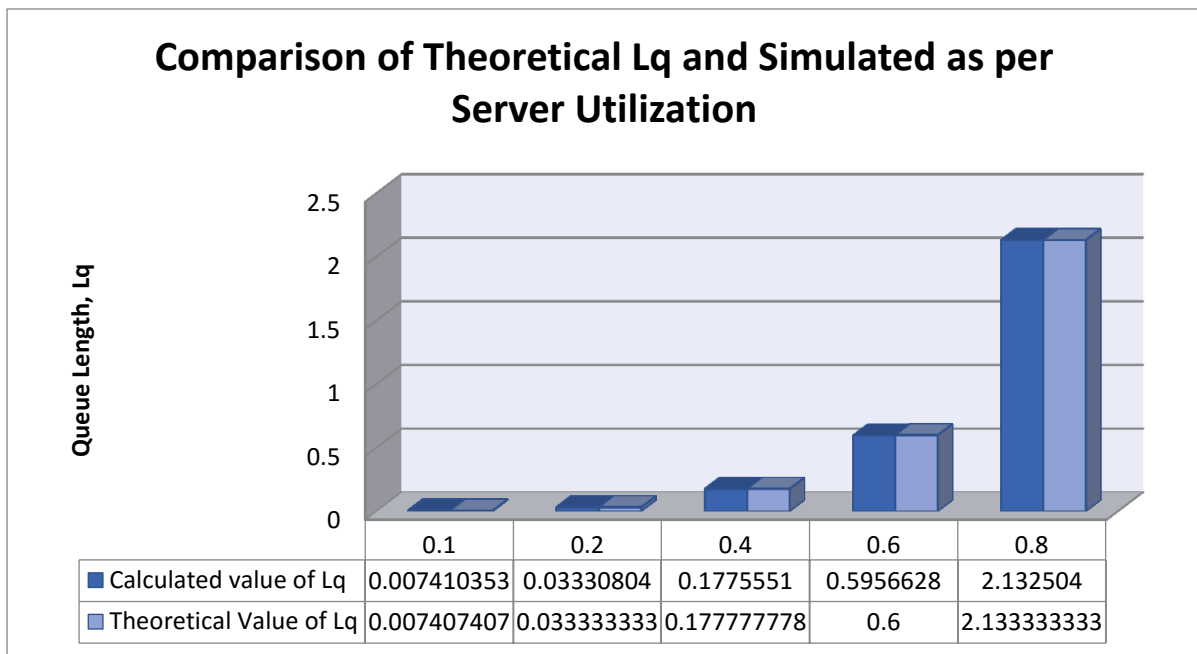
### Comparison of Theoretical and simulated Parameters as per Server utilization:



Graph 10: Comparison of Theoretical Inter-arrival and Simulated as per Server Utilization

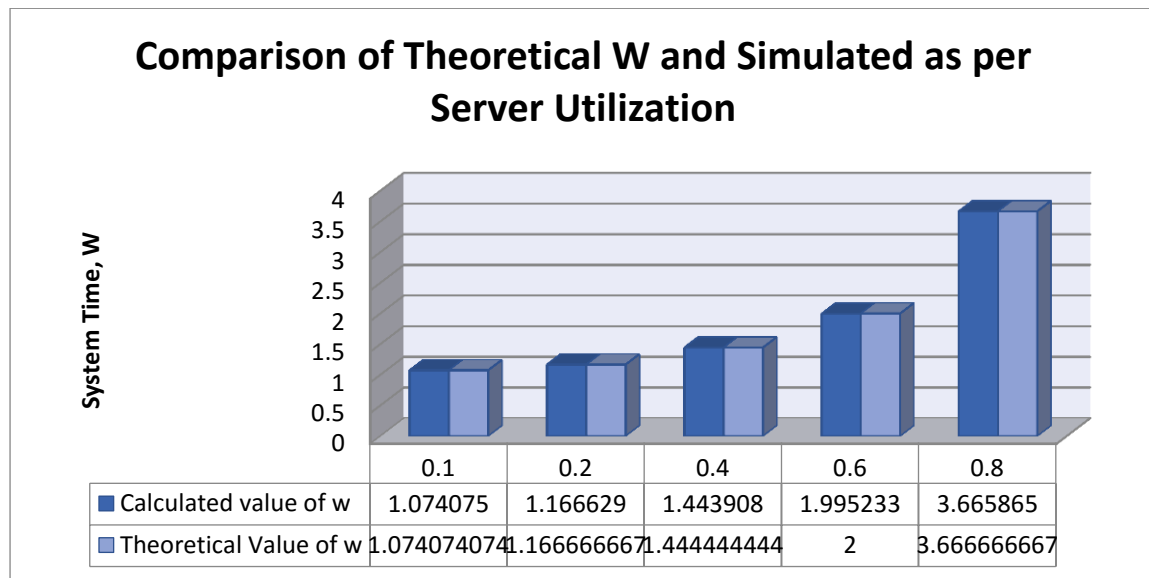


*Graph 11: Comparison of Theoretical L and Simulated as per Server Utilization*

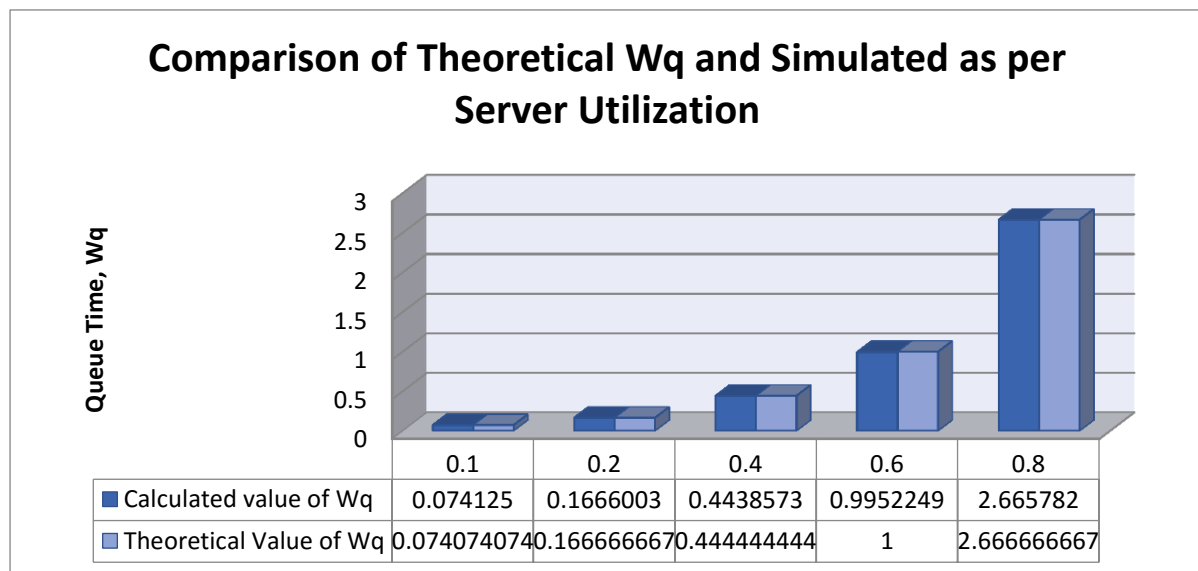


*Graph 12: Comparison of Theoretical Lq and Simulated as per Server Utilization*



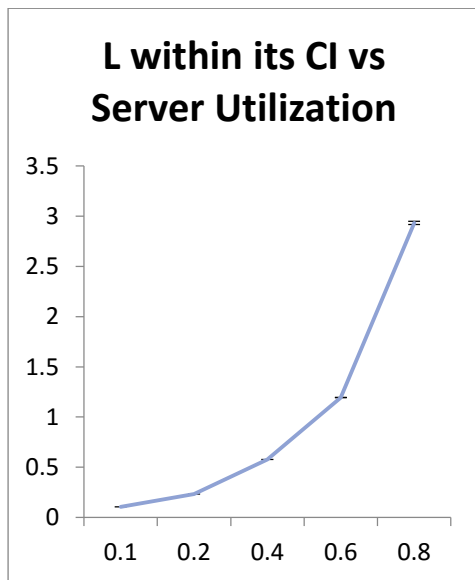


Graph 13: Comparison of Theoretical W and Simulated as per Server Utilization

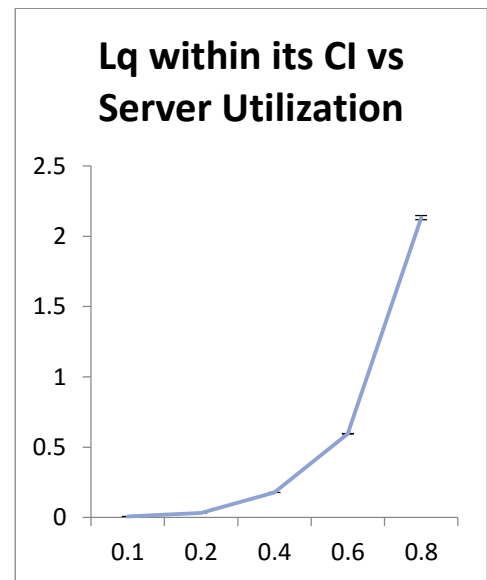


Graph 14: Comparison of Theoretical Wq and Simulated as per Server Utilization

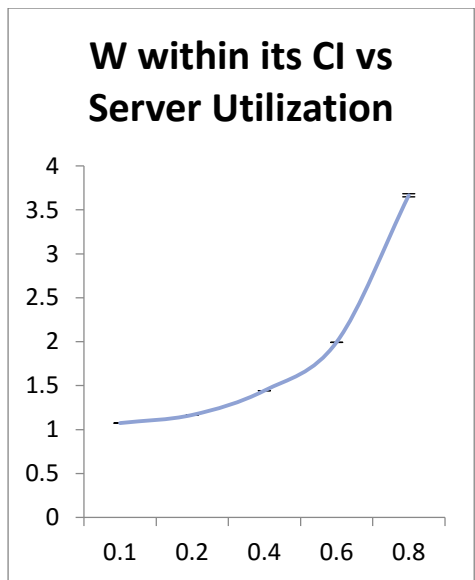
Plotting Server Utilization vs Parameters within CI (Confidence Intervals):



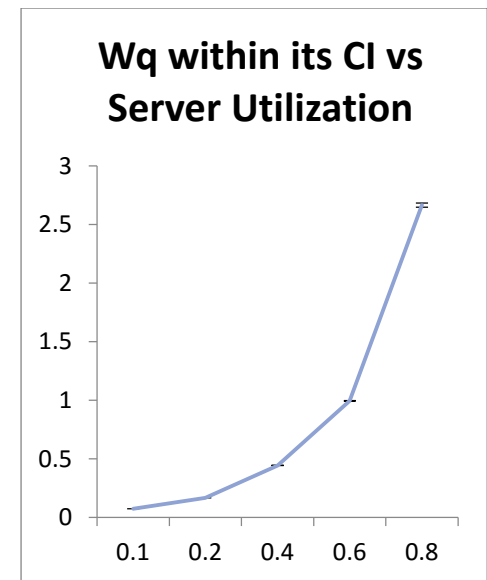
Graph 15: L versus rho



Graph 16: Lq versus rho



Graph 17: W versus rho



Graph 18: Wq versus rho

### Data Analysis for M/G/1:

Here we will be computing the Confidence Intervals for all the Statistics collected from the M/G/1 model. The Confidence Intervals with the Actual and Theoretical data can be viewed in the table below.

When Inter-arrival time is 10	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	0.1073766	0.107407407	7.88918E-05	0.107455492	0.107297708
	Lq	0.007410353	0.007407407	2.17431E-05	0.007432096	0.00738861
	W	1.074075	1.074074074	0.000570699	1.074645699	1.073504301
	Wq	0.074125	0.074074074	0.000206368	0.074331368	0.073918632

	Rho	0.09996606	0.1	6.58851E-05	0.100031945	0.099900175
When Inter-arrival time is 5	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	0.23343725	0.233333333	0.000287682	0.233724932	0.233149568
	Lq	0.03330804	0.033333333	0.000115912	0.033423952	0.033192128
	W	1.166629	1.166666667	0.00063489	1.16726389	1.16599411
	Wq	0.1666003	0.166666667	0.000458024	0.167058324	0.166142276
	Rho	0.199933	0.2	0.000189207	0.200122207	0.199743793
When Inter-arrival time is 2.5	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	0.5776031	0.577777778	0.000850107	0.578453207	0.576752993
	Lq	0.1775551	0.177777778	0.000554754	0.178109854	0.177000346
	W	1.443908	1.444444444	0.001377088	1.445285088	1.442530912
	Wq	0.4438573	0.444444444	0.001213217	0.445070517	0.442644083
	Rho	0.4000472	0.4	0.00039446	0.40044166	0.39965274
When Inter-arrival time is 1.67	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	1.194187	1.2	0.002227017	1.196414017	1.191959983
	Lq	0.5956628	0.6	0.001900124	0.597562924	0.593762676
	W	1.995233	2	0.003237953	1.998470953	1.991995047
	Wq	0.9952249	1	0.002974813	0.998199713	0.992250087
	Rho	0.598524	0.6	0.000482969	0.599006969	0.598041031
When Inter-arrival time is 1.25	Metric	Actual Value	Theoretical Value	Confidence +/-	Confidence Interval	
	L	2.932528	2.933333333	0.014779265	2.947307265	2.917748735
	Lq	2.132504	2.133333333	0.014503364	2.147007364	2.118000636
	W	3.665865	3.666666667	0.018166316	3.684031316	3.647698684
	Wq	2.665782	2.666666667	0.017915853	2.683697853	2.647866147
	Rho	0.8000198	0.8	0.000464181	0.800483981	0.799555619

Table 6: Data Analysis for M/G/1

## Test for Random Number Generator

When a random number generator is devised, one needs to test its property. The two properties in which we are concerned most are uniformity and independence. Here, we have to test these properties for the random number generator of OMNeT++.

We can do this by using a chi-square test.

The whole process is described here:

1. Generating random numbers by applying a function in the source code:  
We will first generate the random numbers by adding a corresponding function in the Source.cc file of queuinglib as shown in the figure below.

```

random = registerSignal("random_number");
// Generation of random numbers.
gen = cModule::getRNG(0);

```

Figure 11: Random number generation in Source.cc

Then we will emit this signal to get all the random numbers.

2. A configuration for the generation of 1000 random numbers has to be written in the omnetpp.ini file. The number of random numbers to be taken has to be mentioned in the parameter "\*\*.numJobs". The code for omnetpp.ini configuration should be written as:

```

[Config Random_Number]
description = "a random number generator"
network = MM1
**.numJobs = 1000
**.interArrivalTime = exponential(10s)
**.serviceTime = exponential(1s)
**.capacity = -1
repeat=5

```

Figure 12: Configuration written in omnetpp.ini file

3. After executing the omnetpp.ini file, we will collect the data (1000 random numbers) specified as vector quantities, from the Analysis (.anf) file. And export the data into a .csv file.
4. Now, the chi-square test can be performed on the data exported.

#### Chi-Square Test For uniformity:

For this test, if we have 1000 numbers (raw data) then we should take the number of intervals as  $\sqrt{1000}$ (approx).

So, after taking the number of bins as 32, we have to fetch the number of observed random numbers in every interval.

Number of Interval	Intervals		Observed, $O_i$	Expected, $E_i$	$(O_i - E_i)^2$	$\frac{(O_i - E_i)^2}{E_i}$
1	0	0.03125	23	31.25	68.0625	2.178
2	0.03125	0.0625	34	31.25	7.5625	0.242
3	0.0625	0.09375	26	31.25	27.5625	0.882
4	0.09375	0.125	21	31.25	105.0625	3.362
5	0.125	0.15625	31	31.25	0.0625	0.002
6	0.15625	0.1875	31	31.25	0.0625	0.002
7	0.1875	0.21875	31	31.25	0.0625	0.002
8	0.21875	0.25	36	31.25	22.5625	0.722
9	0.25	0.28125	31	31.25	0.0625	0.002
10	0.28125	0.3125	42	31.25	115.5625	3.698
11	0.3125	0.34375	35	31.25	14.0625	0.45
12	0.34375	0.375	28	31.25	10.5625	0.338

13	0.375	0.40625	26	31.25	27.5625	0.882
14	0.40625	0.4375	26	31.25	27.5625	0.882
15	0.4375	0.46875	32	31.25	0.5625	0.018
16	0.46875	0.5	26	31.25	27.5625	0.882
17	0.5	0.53125	30	31.25	1.5625	0.05
18	0.53125	0.5625	26	31.25	27.5625	0.882
19	0.5625	0.59375	23	31.25	68.0625	2.178
20	0.59375	0.625	39	31.25	60.0625	1.922
21	0.625	0.65625	28	31.25	10.5625	0.338
22	0.65625	0.6875	39	31.25	60.0625	1.922
23	0.6875	0.71875	32	31.25	0.5625	0.018
24	0.71875	0.75	31	31.25	0.0625	0.002
25	0.75	0.78125	25	31.25	39.0625	1.25
26	0.78125	0.8125	40	31.25	76.5625	2.45
27	0.8125	0.84375	40	31.25	76.5625	2.45
28	0.84375	0.875	31	31.25	0.0625	0.002
29	0.875	0.90625	44	31.25	162.5625	5.202
30	0.90625	0.9375	29	31.25	5.0625	0.162
31	0.9375	0.96875	29	31.25	5.0625	0.162
32	0.96875	1	35	31.25	14.0625	0.45
Chi-square, $\chi^2_0 =$						33.984

Table 7: Chi-square Test for uniformity of 1000 random numbers

Therefore, the chi-square value = 33.984

And the critical value of chi-square can be computed as:

Degrees of freedom =  $n-1 = 32 - 1 = 31$

Level of significance,  $\alpha = 0.05$

Critical Chi-square value,  $\chi^2_{0.05,31} = 44.985$

As,

$$\chi^2_0 = 33.984 < \chi^2_{0.05,31} = 44.985$$

Therefore, the ( $H_0$ ) Hypothesis of this data of random numbers confirming uniformity is not rejected.

## Extended Queuing (Priority Queue)

For the extension of our project we would like to introduce the concept of priority in our model. Since a particular priority will be assigned to all of the queues, the server will fetch the jobs using the fetching algorithm.

In this scenario, passengers travelling from First Class will get a higher priority than the passengers from Business Class and the Business Class ones will get a higher priority than the Economy ones. There's only one check-in counter open. It serves the queues in priority order -- economy class passengers are only served if there're no business class or first class passengers.

The Source (Passengers) block is still generating the passengers at an inter-arrival time of 10s, 5s, 2.5s, 1.67s, and 1.25s. Again, the idea behind this is to vary server utilization. The

main difference of this model is the use of more than 1 queue that too Passive Queues (Queues without server) and a classifier. A Binomial distribution is used by the classifier to simulate the separation of passengers into business and economy, in order to provide a varied priority to every associated queue. The architecture for this model will be as shown in figure 11.

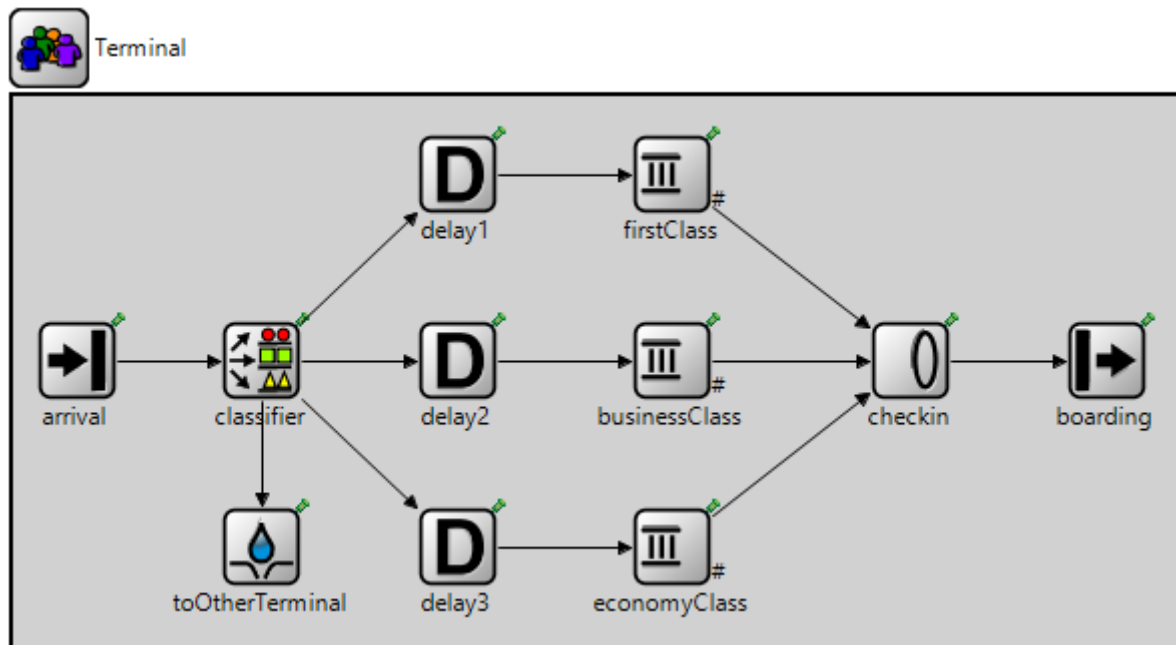
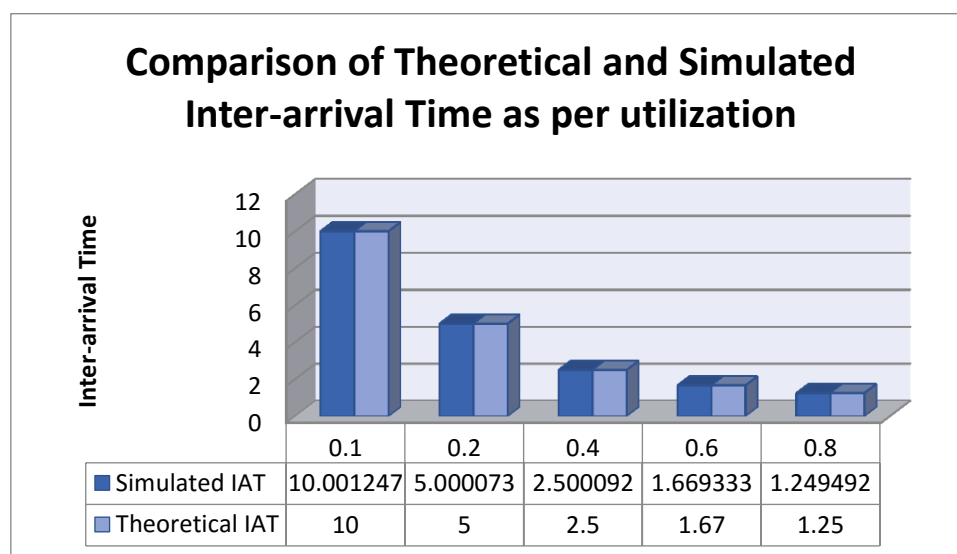


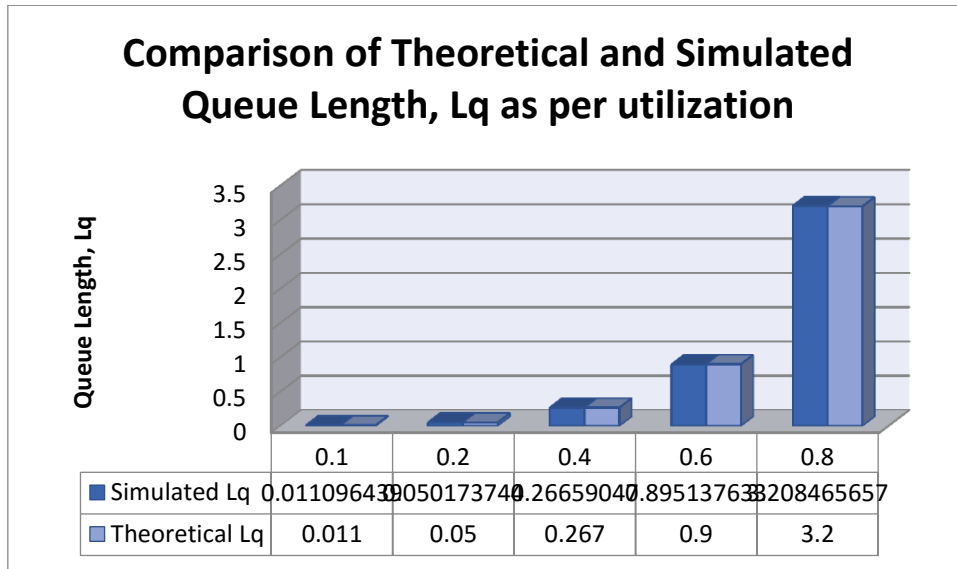
Figure 13: Diagram of an NED file with Priority queue.

#### Analysis:

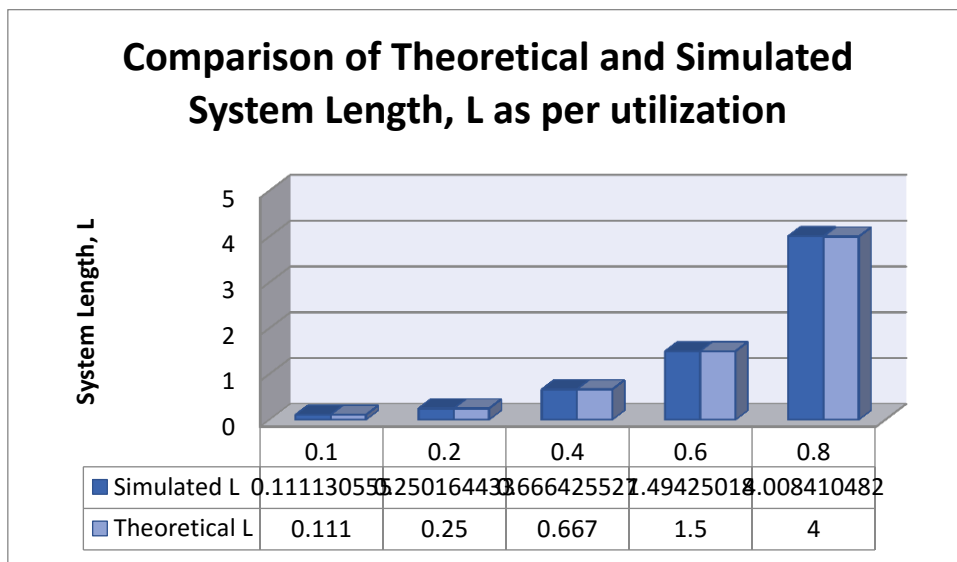
Comparison of Theoretical and simulated Parameters as per Server utilization:



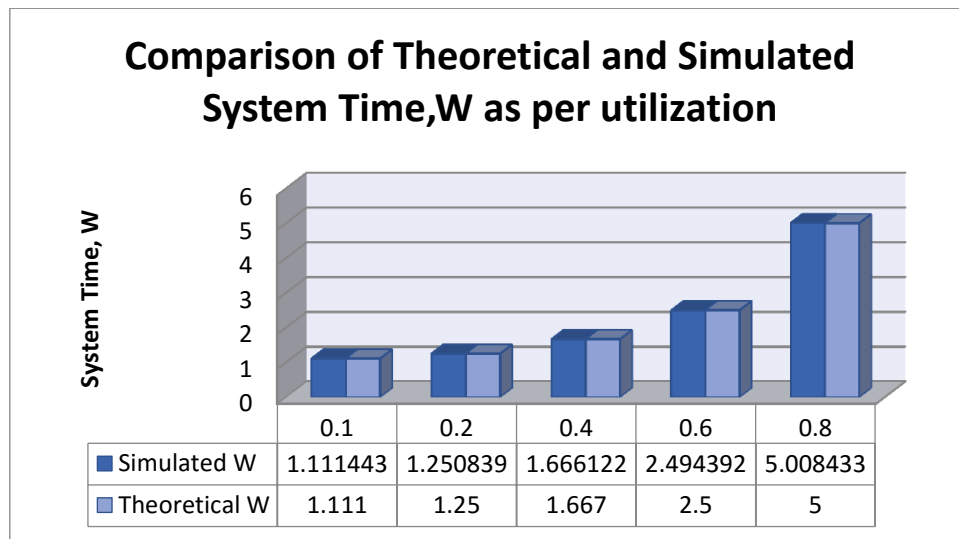
Graph 19: Comparison of Theoretical and Simulated Inter-arrival Time as per utilization



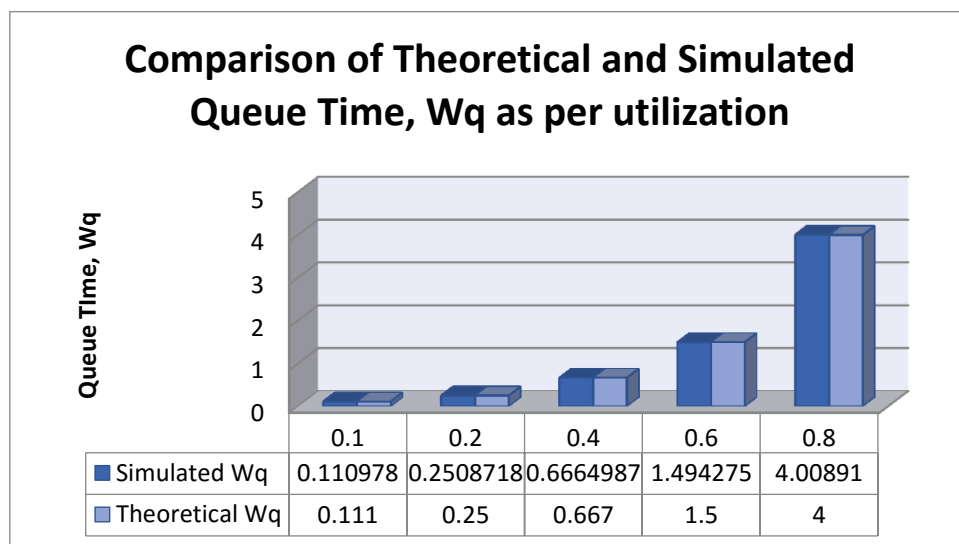
Graph 20: Comparison of Theoretical and Simulated System Length,  $L$  as per utilization



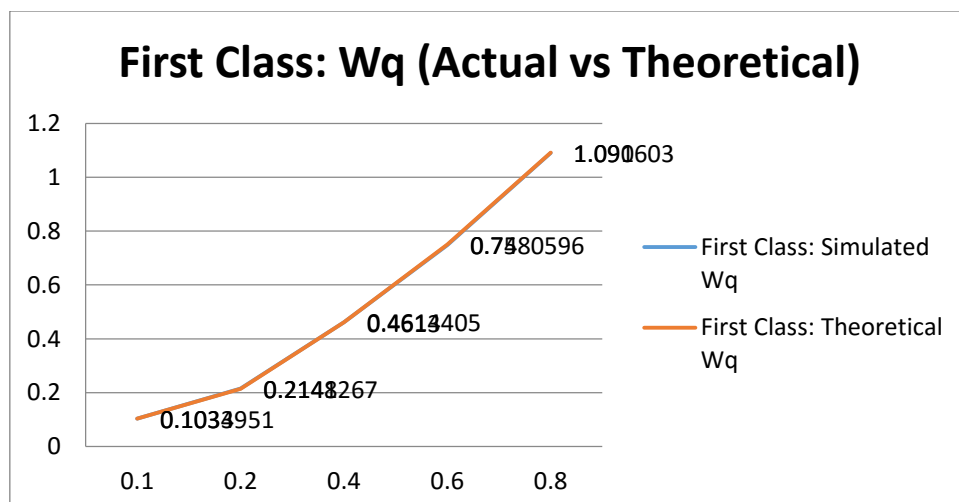
Graph 21: Comparison of Theoretical and Simulated Queue Length,  $L_q$  as per utilization



Graph 22: Comparison of Theoretical and Simulated System Time,  $w$  as per utilization

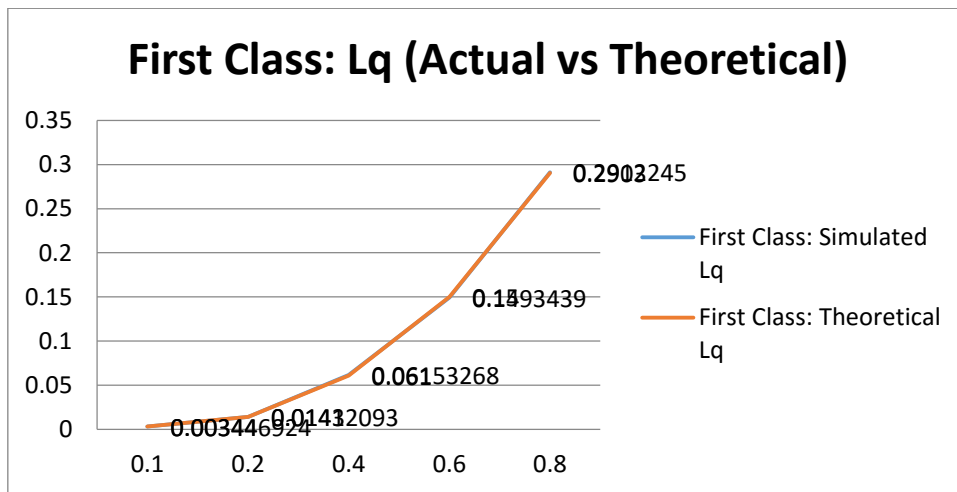


Graph 23: Comparison of Theoretical and Simulated Queue Time,  $w_q$  as per utilization

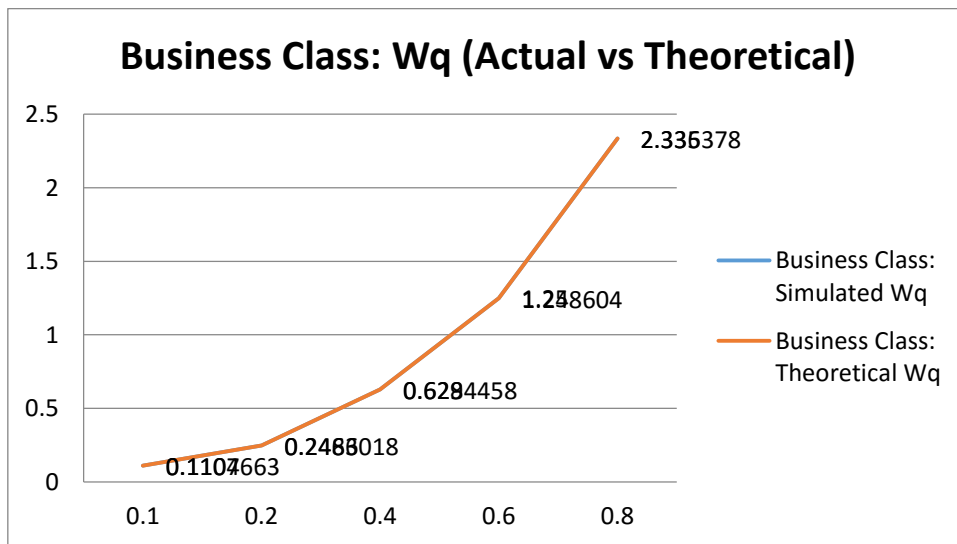


Graph 24: First Class:  $W_q$  (Actual vs Theoretical)

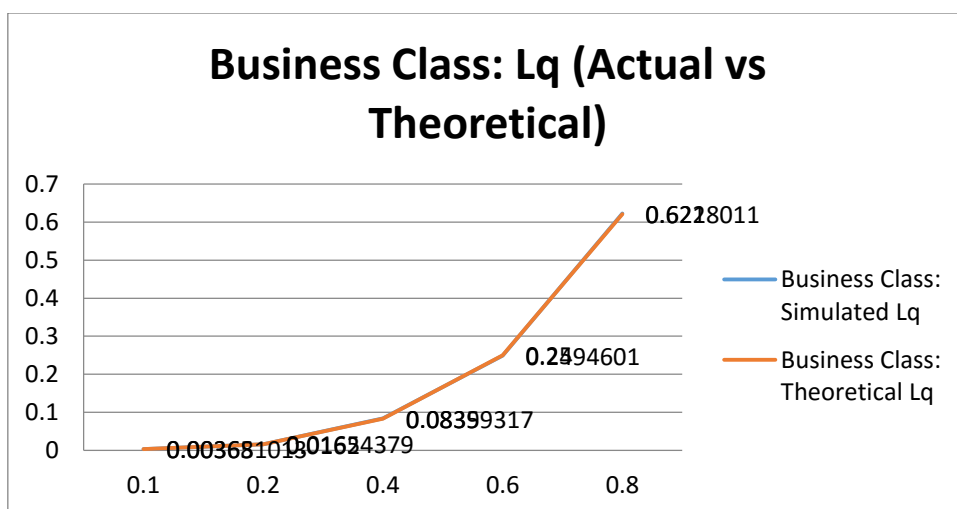




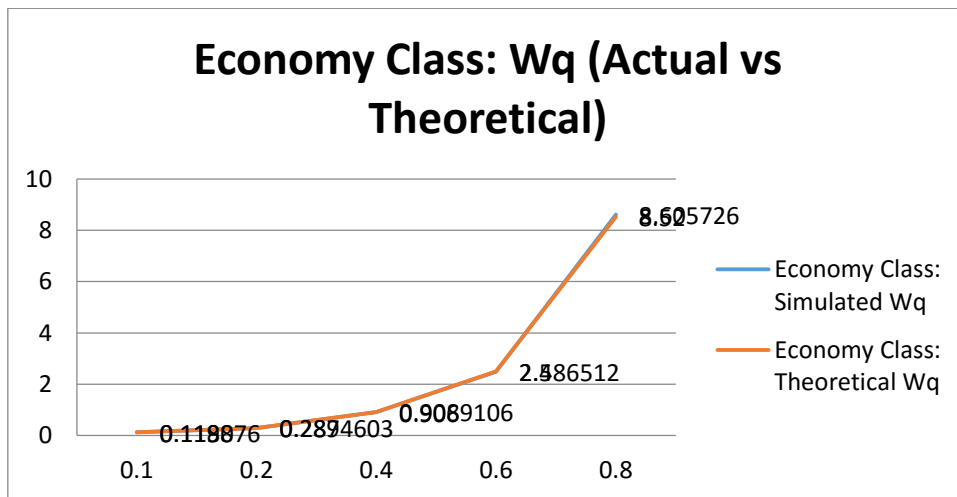
Graph 25: First Class: Wq (Actual vs Theoretical)



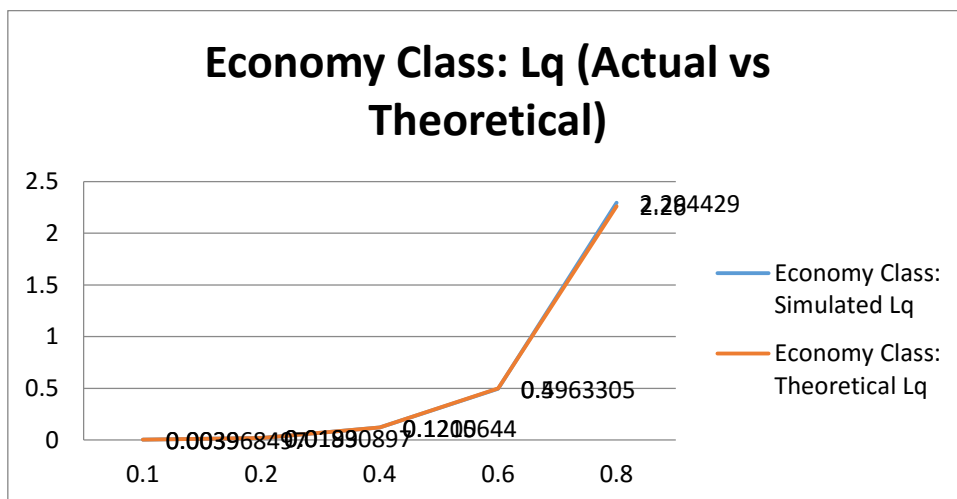
Graph 26: Business Class: Wq (Actual vs Theoretical)



Graph 27: Business Class: Lq (Actual vs Theoretical)

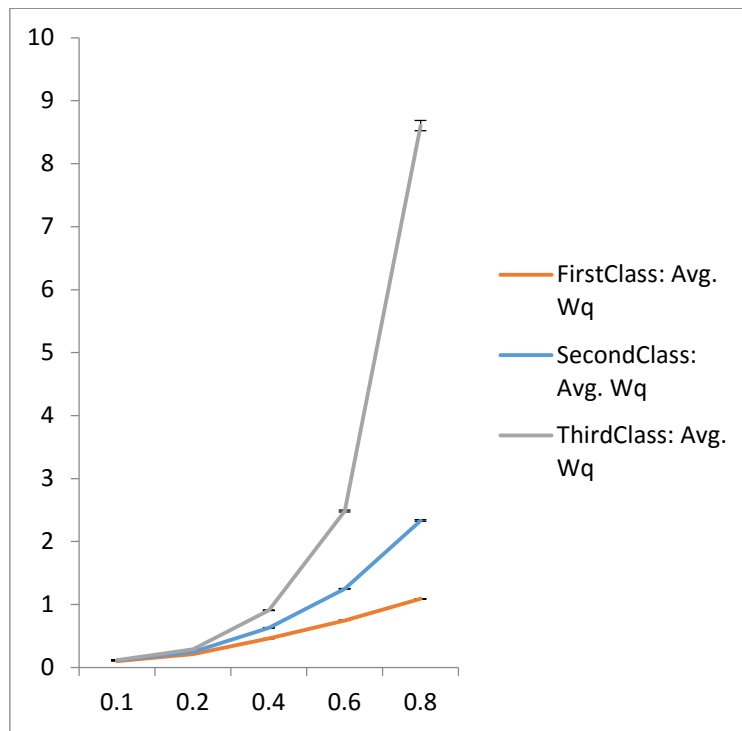


Graph 28: Economy Class: Wq (Actual vs Theoretical)

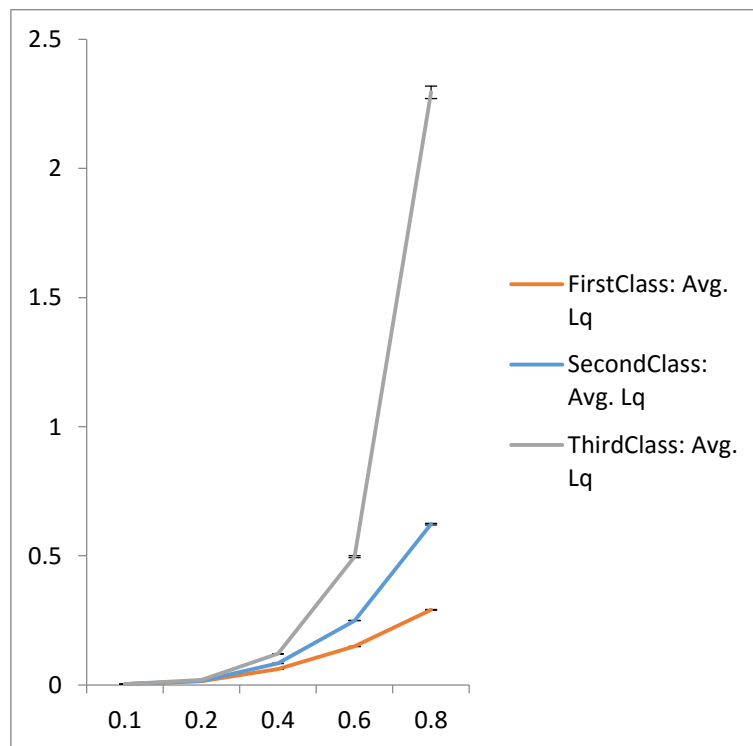


Graph 29: Economy Class: Lq (Actual vs Theoretical)

Plotting Server Utilization versus Parameters within CI (Confidence Intervals):



Graph 30: Comparison of Server Utilization versus  $wq$  (within CI) in all the queues



Graph 31: Comparison of Server Utilization versus  $Lq$  in all the queues

## Conclusion

Three queuing models have been simulated and compared for Airline Check-in counter with prediction of queuing theory. Good agreement between queuing theory and the simulation results have been shown and confirmed with the help of graphs and histograms, using confidence intervals.

We have assumed the System's capacity to be infinite. And the number of passengers (jobs) to be 890000. Some performance measures, such as mean system length, queue length, total system time, total queue time, the server utilization, etc. are obtained.

After performing all the three queuing models M/M/1 queue, M/G/1 queue and Priority queue model, we infer that the Length of the queue,  $L_q$  and the Time spent in the queue,  $w_q$  in the case of First Class passengers in the Priority queue is significantly lower than in the other queues.

For a better simulation of the actual model, a large amount of data and simulations are required. And here considering 890000 passengers, the data obtained from the simulation is much closer to the theoretical data. This concludes that we are serving enough number of passengers from the Source and we are performing just enough number of replications to better replicate the real system. Therefore, this simulation closely replicates the real system.