ECS 152A Phase I

3.7

The Lambda list is [0.1,0.25,0.4,0.55,0.65,0.80,0.90], Service/Transmission Rate And Max\_Buffer equal to 100000(which is close to infinity)

Result is:

-------------------------------------------------

Lambda is: 0.65Max\_Buffer: 100000

Mean\_Queue\_Length: 1.52102503827655

Total Package Loss: 0

Utilization: 0.64661

-------------------------------------------------

Lambda is: 0.8Max\_Buffer: 100000

Mean\_Queue\_Length: 3.5173245862081965

Total Package Loss: 0

Utilization: 0.79569

-------------------------------------------------

Lambda is: 0.9Max\_Buffer: 100000

Mean\_Queue\_Length: 8.678097369645666

Total Package Loss: 0

Utilization: 0.90254

-------------------------------------------------

---------------Starting Phase I---------------

Lambda is: 0.1Max\_Buffer: 100000

Mean\_Queue\_Length: 0.06037289138798461

Total Package Loss: 0

Utilization: 0.10137

-------------------------------------------------

Lambda is: 0.25Max\_Buffer: 100000

Mean\_Queue\_Length: 0.2047513589691967

Total Package Loss: 0

Utilization: 0.24835

-------------------------------------------------

Lambda is: 0.4Max\_Buffer: 100000

Mean\_Queue\_Length: 0.4684664416947881

Total Package Loss: 0

Utilization: 0.40005

-------------------------------------------------

Lambda is: 0.55Max\_Buffer: 100000

Mean\_Queue\_Length: 0.9219481110605371

Total Package Loss: 0

Utilization: 0.54925

-------------------------------------------------

3.8

The Mathematically formulae for Utilization and Mean Queue Length are:

Mathematically Result for Part 3.7 is:

-------------------------------------------------

Lambda is: 0.8

Mathematically Mean\_Queue\_Length: 3.20

Mathematically Utilization: 0.8

-------------------------------------------------

Lambda is: 0.9

Mathematically Mean\_Queue\_Length: 8.10

Mathematically Utilization: 0.9

-------------------------------------------------

-------------------------------------------------

Lambda is: 0.1

Mathematically Mean\_Queue\_Length: 0.0111

Mathematically Utilization: 0.1

-------------------------------------------------

Lambda is: 0.25

Mathematically Mean\_Queue\_Length: 0.0833

Mathematically Utilization: 0.25

-------------------------------------------------

Lambda is: 0.4

Mathematically Mean\_Queue\_Length: 0.2667

Mathematically Utilization: 0.4

-------------------------------------------------

Lambda is: 0.55

Mathematically Mean\_Queue\_Length: 0.6722

Mathematically Utilization: 0.55

-------------------------------------------------

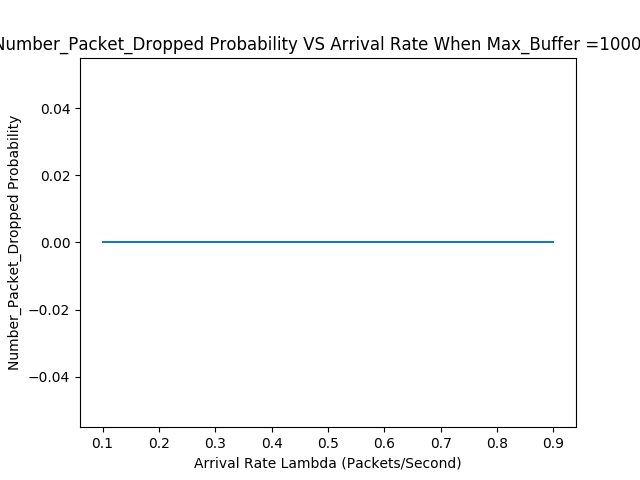
Lambda is: 0.65

Mathematically Mean\_Queue\_Length: 1.2071

Mathematically Utilization: 0.65

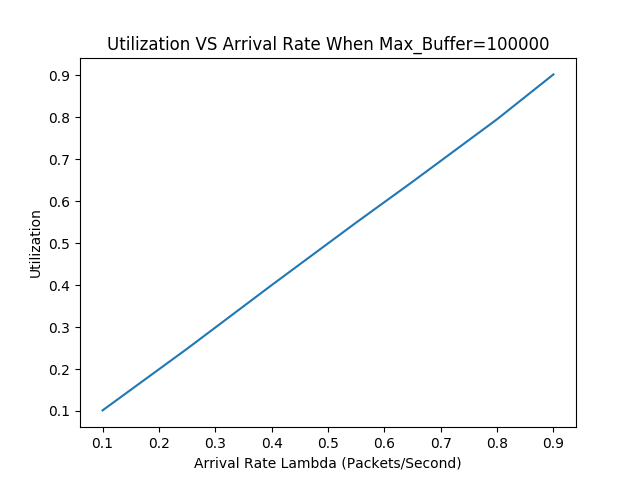
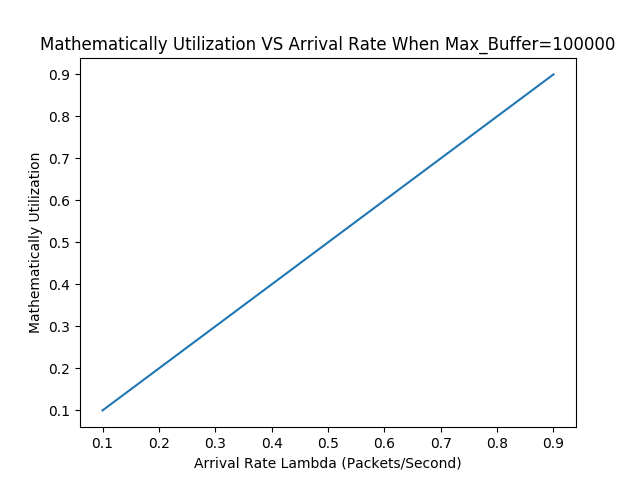
-------------------------------------------------

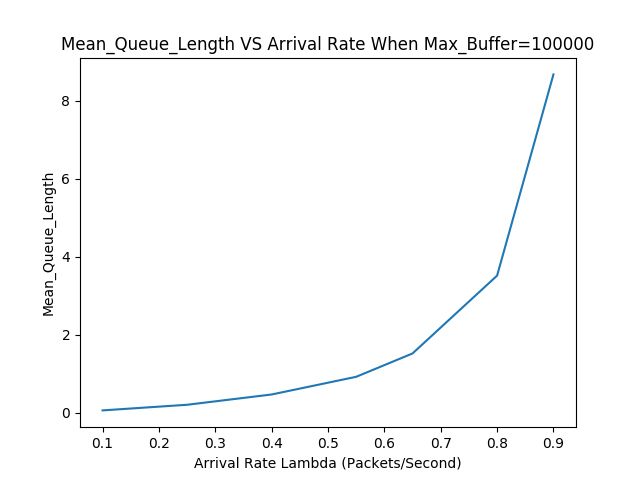
* The Packet dropped plot for simulation of Part 3.7 is the plot below:

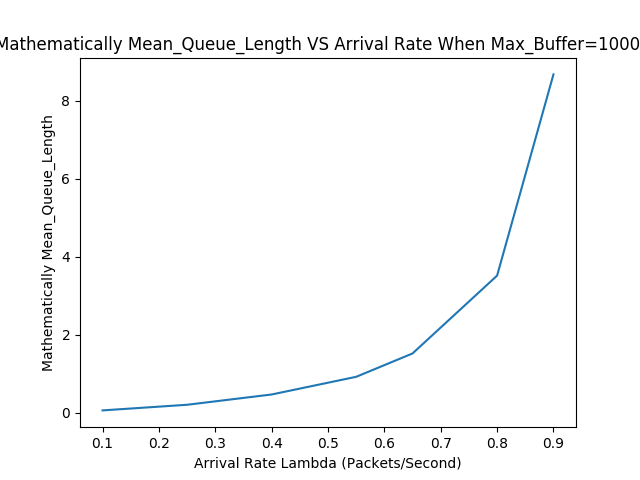


This plot make sense because when the Maximum Buffer size is close to infinity, there’s no packet loss at all.

* The comparison of Simulated and Mathematically figures are given by below plot







The Mathematically Utilization and Simulation Utilization is approximately the same.

The Mathematically Mean Queue Length and Simulation Mean Queue Length, however, have some difference. The Mathematically Mean Queue Length is always smaller than the Simulation Mean Queue Length. Overall, the shape of both plot is quite similar.

3.9

Then, we are going to vary the Max\_Buffer = [1,20,50]

-------------------------------------------------

Lambda is: 0.2Max\_Buffer: 20

Mean\_Queue\_Length: 0.14891364621666583

Total Package Loss: 0

Utilization: 0.19837

-------------------------------------------------

Lambda is: 0.4Max\_Buffer: 20

Mean\_Queue\_Length: 0.4610344312123175

Total Package Loss: 0

Utilization: 0.40138

-------------------------------------------------

Lambda is: 0.6Max\_Buffer: 20

Mean\_Queue\_Length: 1.2421347662404432

Total Package Loss: 0

Utilization: 0.60297

-------------------------------------------------

Lambda is: 0.8Max\_Buffer: 20

Mean\_Queue\_Length: 3.495253637632051

Total Package Loss: 163

Utilization: 0.80272

-------------------------------------------------

Lambda is: 0.9Max\_Buffer: 20

Mean\_Queue\_Length: 6.067291585778468

Total Package Loss: 1216

Utilization: 0.90145

-------------------------------------------------

-------------------------------------------------

Lambda is: 0.2Max\_Buffer: 1

Mean\_Queue\_Length: 0.10357178892140047

Total Package Loss: 2053

Utilization: 0.19822

-------------------------------------------------

Lambda is: 0.4Max\_Buffer: 1

Mean\_Queue\_Length: 0.21367585756616944

Total Package Loss: 8509

Utilization: 0.39822

-------------------------------------------------

Lambda is: 0.6Max\_Buffer: 1

Mean\_Queue\_Length: 0.29908463783496947

Total Package Loss: 18036

Utilization: 0.60304

-------------------------------------------------

Lambda is: 0.8Max\_Buffer: 1

Mean\_Queue\_Length: 0.37810368264182476

Total Package Loss: 30319

Utilization: 0.80187

-------------------------------------------------

Lambda is: 0.9Max\_Buffer: 1

Mean\_Queue\_Length: 0.4097710433959111

Total Package Loss: 36779

Utilization: 0.89755

-------------------------------------------------

And Mathematical Statistic in below box

**Mathematical**

-------------------------------------------------

Lambda is: 0.2

Mathematically Mean\_Queue\_Length: 0.05

Mathematically Utilization: 0.2

-------------------------------------------------

Lambda is: 0.4

Mathematically Mean\_Queue\_Length: 0.2667

Mathematically Utilization: 0.4

-------------------------------------------------

Lambda is: 0.6

Mathematically Mean\_Queue\_Length: 0.9

Mathematically Utilization: 0.6

-------------------------------------------------

Lambda is: 0.8

Mathematically Mean\_Queue\_Length: 3.2

Mathematically Utilization: 0.8

-------------------------------------------------

Lambda is: 0.9

Mathematically Mean\_Queue\_Length: 8.1

Mathematically Utilization: 0.9

-------------------------------------------------

-------------------------------------------------

Lambda is: 0.2Max\_Buffer: 50

Mean\_Queue\_Length: 0.1410146173688736

Total Package Loss: 0

Utilization: 0.19771

-------------------------------------------------

Lambda is: 0.4Max\_Buffer: 50

Mean\_Queue\_Length: 0.48191175735771746

Total Package Loss: 0

Utilization: 0.40026

-------------------------------------------------

Lambda is: 0.6Max\_Buffer: 50

Mean\_Queue\_Length: 1.235029416176397

Total Package Loss: 0

Utilization: 0.60001

-------------------------------------------------

Lambda is: 0.8Max\_Buffer: 50

Mean\_Queue\_Length: 3.5835247415988416

Total Package Loss: 0

Utilization: 0.80108

-------------------------------------------------

Lambda is: 0.9Max\_Buffer: 50

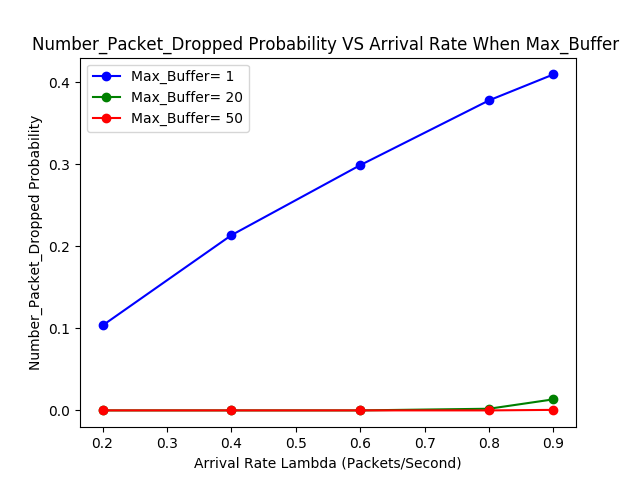
Mean\_Queue\_Length: 7.993275492907485

Total Package Loss: 57

Utilization: 0.89672

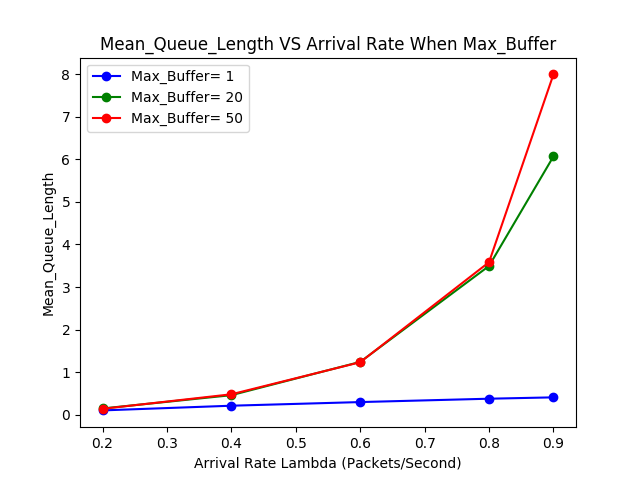
-------------------------------------------------

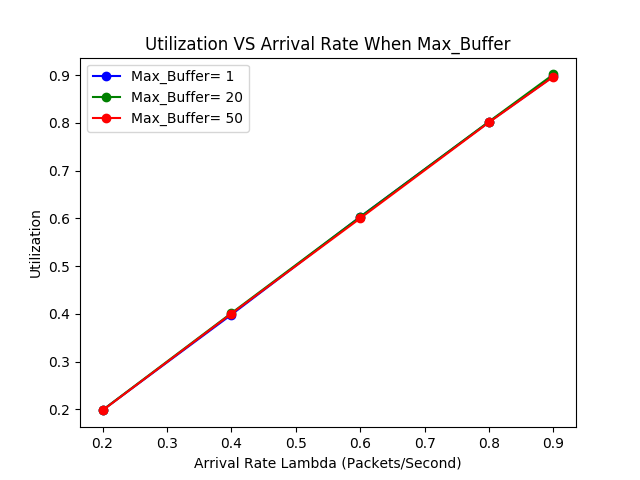
* The Packet dropped plot for simulation of Part 3.9 is the plot below:



This plot shows that the probability of dropping a packet is decreasing when Max\_Buffer increase, which means more rooms in the queue to avoid packet loss. It also makes sense that the probability of dropping a packet increase when the arrival rate increase.

* The Simulated Utilization and Mean Queue Length figures are given by below plot:





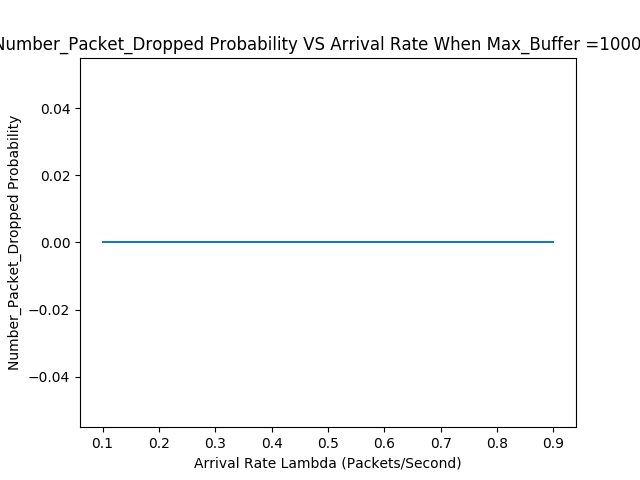
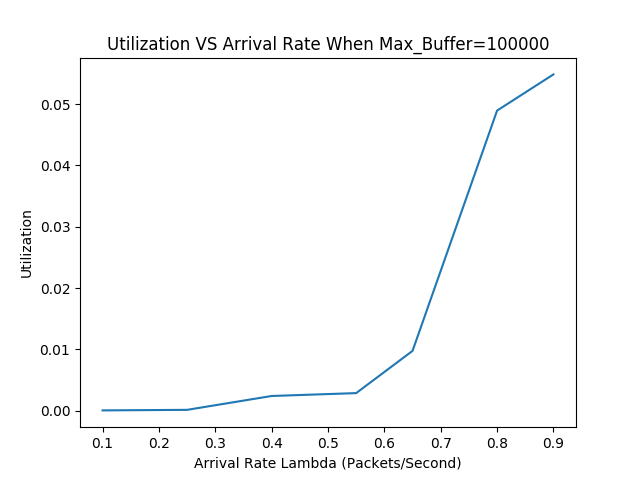
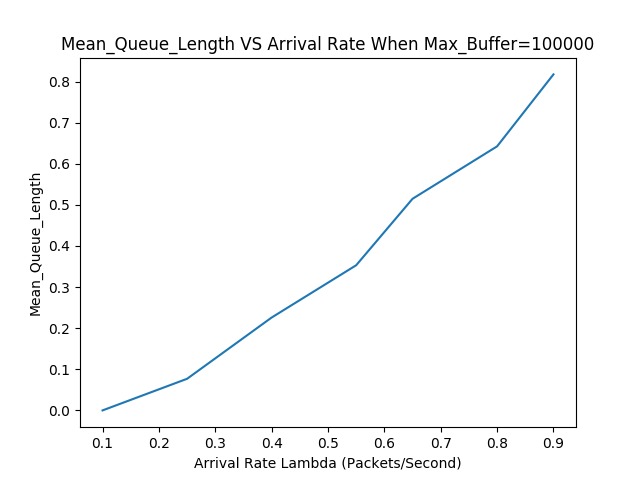
Here, the Utilization for both Mathematical formulae and Simulation are still approximate the same. It increases approximately linearly with increase in arrival rate.

However, the Mean Queue Length are different from case in 3.7. When the Max\_Buffer = 50, its shape looks similar to what we have in 3.7. However, when Max\_Buffer size decrease, the line goes down. This decrease actually corresponds to the packet loss in the plot above. That is, the decrease in Mean Queue Length was partially because the increase in packet loss probability.

Extra Credit

Parameter α is called shape parameter that determines heavy-tailed characteristics.

After using the Pareto Distribution for Arrival Time, the three statistics are shown in the following plots.



The difference between plots in 3.7 and 3.9 are the difference in Mean Queue Length and Utilization. The Mean Queue Length seems to have linear relationship with the arrival rate Moreover, the Utilization is no longer have linear relationship with arrival rate, but have a sharp increase when arrival rate is larger than 0.6. Also, the Utilization decreased significantly from 3.7. This shows the Self-similar traffic which is more realistic in the real life. The traffic has similar statistical properties at a range of timescales.

Appendix

# Haozhe Gu 999200555

# ECS 152A Project I Phase I

import simpy

import matplotlib.pyplot as plt

import math

import random

import numpy as np

Lambda = [0.1,0.25,0.4,0.55,0.65,0.80,0.90]

Mu = 1

Max\_Buffer = [1,20,50]

Max\_Time = 100000

##def Negative\_Exponentially\_Distributed\_Time( rate)

print("---------------Starting Phase I---------------\n")

class Global\_Event\_list(object):

def \_\_init\_\_(self,env,Lambda,Mu,Max\_Buffer,mode):

self.env = env

self.Lambda = Lambda

self.Packet\_Loss = 0

self.Total\_Packet = 0

self.Sum\_Queue\_Length = 0

self.mode = mode

self.Queue = Queue(self.env,Mu,Max\_Buffer)

# Start the run process everytime an instance is created.

self.action = env.process(self.run())

def run(self):

while True:

if(self.Total\_Packet!=0):

self.Sum\_Queue\_Length = self.Sum\_Queue\_Length+self.Queue.Length

self.Total\_Packet = self.Total\_Packet+1

if(self.Queue.Length < self.Queue.Capacity):

self.Queue.Length = self.Queue.Length +1

else:

self.Packet\_Loss = self.Packet\_Loss+1

if(self.mode==1):

Time = random.expovariate(self.Lambda)

else:

Time = np.random.pareto(self.Lambda)

yield env.timeout(Time)

class Queue(object):

def \_\_init\_\_(self,env,Mu,Max\_Buffer):

self.env = env

self.Capacity = Max\_Buffer

self.Length = 0

self.Mu = Mu

self.action = env.process(self.run())

def run(self):

while True:

if (self.Length>0): #Service the first Packet in the Queue

self.Length = self.Length-1

Time = random.expovariate(self.Mu)

yield env.timeout(Time)

else: #IDLE,No Packet,Let the time run

yield env.timeout(1)

#MAIN

#PART1

j=100000

Utilization = []

Mean\_Queue\_Length=[]

Number\_Packet\_Dropped=[]

for i in Lambda:

env = simpy.Environment()

Test = Global\_Event\_list(env,i,Mu,j,0)

env.run(until=Max\_Time)

Utilization.append(Test.Total\_Packet/Max\_Time)

Mean\_Queue\_Length.append(Test.Sum\_Queue\_Length/Test.Total\_Packet)

Number\_Packet\_Dropped.append(Test.Packet\_Loss/Test.Total\_Packet)

print("Lambda is: "+str(i)+"Max\_Buffer: "+str(j)+"\n")

print("Mean\_Queue\_Length: "+str(Test.Sum\_Queue\_Length/Test.Total\_Packet)+"\n")

print("Total Package Loss: "+str(Test.Packet\_Loss)+"\n")

print("Utilization: "+str(Test.Total\_Packet/Max\_Time)+"\n-------------------------------------------------\n")

fig = plt.figure()

plt.plot(Lambda,Utilization)

plt.xlabel("Arrival Rate Lambda (Packets/Second)")

plt.ylabel("Utilization")

plt.title("Utilization VS Arrival Rate When Max\_Buffer="+str(j))

plt.savefig("Utilization\_Max\_Buffer="+str(j)+ ".png")

fig = plt.figure()

plt.plot(Lambda,Mean\_Queue\_Length)

plt.xlabel("Arrival Rate Lambda (Packets/Second)")

plt.ylabel("Mean\_Queue\_Length")

plt.title("Mean\_Queue\_Length VS Arrival Rate When Max\_Buffer="+str(j))

plt.savefig("Mean\_Queue\_Length\_Max\_Buffer="+str(j)+ ".png")

fig = plt.figure()

plt.plot(Lambda,Number\_Packet\_Dropped)

plt.xlabel("Arrival Rate Lambda (Packets/Second)")

plt.ylabel("Number\_Packet\_Dropped Probability")

plt.title("Number\_Packet\_Dropped Probability VS Arrival Rate When Max\_Buffer ="+str(j))

plt.savefig("Number\_Packet\_Dropped\_Max\_Buffer="+str(j)+ ".png")

#Part2

Lambda1 = [0.2,0.4,0.6,0.8,0.9]

j=1

Utilization1 = []

Mean\_Queue\_Length1=[]

Number\_Packet\_Dropped1=[]

for i in Lambda1:

env = simpy.Environment()

Test = Global\_Event\_list(env,i,Mu,j,0)

env.run(until=Max\_Time)

Utilization1.append(Test.Total\_Packet/Max\_Time)

Mean\_Queue\_Length1.append(Test.Sum\_Queue\_Length/Test.Total\_Packet)

Number\_Packet\_Dropped1.append(Test.Packet\_Loss/Test.Total\_Packet)

print("Lambda is: "+str(i)+"Max\_Buffer: "+str(j)+"\n")

print("Mean\_Queue\_Length: "+str(Test.Sum\_Queue\_Length/Test.Total\_Packet)+"\n")

print("Total Package Loss: "+str(Test.Packet\_Loss)+"\n")

print("Utilization: "+str(Test.Total\_Packet/Max\_Time)+"\n-------------------------------------------------\n")

j=20

Utilization2 = []

Mean\_Queue\_Length2=[]

Number\_Packet\_Dropped2=[]

for i in Lambda1:

env = simpy.Environment()

Test = Global\_Event\_list(env,i,Mu,j,0)

env.run(until=Max\_Time)

Utilization2.append(Test.Total\_Packet/Max\_Time)

Mean\_Queue\_Length2.append(Test.Sum\_Queue\_Length/Test.Total\_Packet)

Number\_Packet\_Dropped2.append(Test.Packet\_Loss/Test.Total\_Packet)

print("Lambda is: "+str(i)+"Max\_Buffer: "+str(j)+"\n")

print("Mean\_Queue\_Length: "+str(Test.Sum\_Queue\_Length/Test.Total\_Packet)+"\n")

print("Total Package Loss: "+str(Test.Packet\_Loss)+"\n")

print("Utilization: "+str(Test.Total\_Packet/Max\_Time)+"\n-------------------------------------------------\n")

j=50

Utilization3 = []

Mean\_Queue\_Length3=[]

Number\_Packet\_Dropped3=[]

for i in Lambda1:

env = simpy.Environment()

Test = Global\_Event\_list(env,i,Mu,j,0)

env.run(until=Max\_Time)

Utilization3.append(Test.Total\_Packet/Max\_Time)

Mean\_Queue\_Length3.append(Test.Sum\_Queue\_Length/Test.Total\_Packet)

Number\_Packet\_Dropped3.append(Test.Packet\_Loss/Test.Total\_Packet)

print("Lambda is: "+str(i)+"Max\_Buffer: "+str(j)+"\n")

print("Mean\_Queue\_Length: "+str(Test.Sum\_Queue\_Length/Test.Total\_Packet)+"\n")

print("Total Package Loss: "+str(Test.Packet\_Loss)+"\n")

print("Utilization: "+str(Test.Total\_Packet/Max\_Time)+"\n-------------------------------------------------\n")

print(Utilization1)

print(Utilization2)

print(Utilization3)

fig = plt.figure()

line1 = plt.plot(Lambda1,Utilization1,'bo-',label = "Max\_Buffer= 1")

line2 = plt.plot(Lambda1,Utilization2,'go-',label = "Max\_Buffer= 20")

line3 = plt.plot(Lambda1,Utilization3,'ro-',label = "Max\_Buffer= 50")

lines = line1+line2+line3

labels = [l.get\_label() for l in lines]

plt.legend(labels,loc='upper left')

plt.xlabel("Arrival Rate Lambda (Packets/Second)")

plt.ylabel("Utilization")

plt.title("Utilization VS Arrival Rate When Max\_Buffer")

plt.savefig("Utilization\_Max\_Buffer.png")

fig = plt.figure()

line1 = plt.plot(Lambda1,Mean\_Queue\_Length1,'bo-',label = "Max\_Buffer= 1")

line2 = plt.plot(Lambda1,Mean\_Queue\_Length2,'go-',label = "Max\_Buffer= 20")

line3 = plt.plot(Lambda1,Mean\_Queue\_Length3,'ro-',label = "Max\_Buffer= 50")

lines = line1+line2+line3

labels = [l.get\_label() for l in lines]

plt.legend(labels,loc='upper left')

#plt.plot(Lambda,Mean\_Queue\_Length)

plt.xlabel("Arrival Rate Lambda (Packets/Second)")

plt.ylabel("Mean\_Queue\_Length")

plt.title("Mean\_Queue\_Length VS Arrival Rate When Max\_Buffer")

plt.savefig("Mean\_Queue\_Length\_Max\_Buffer.png")

fig = plt.figure()

line1 = plt.plot(Lambda1,Number\_Packet\_Dropped1,'bo-',label = "Max\_Buffer= 1")

line2 = plt.plot(Lambda1,Number\_Packet\_Dropped2,'go-',label = "Max\_Buffer= 20")

line3 = plt.plot(Lambda1,Number\_Packet\_Dropped3,'ro-',label = "Max\_Buffer= 50")

lines = line1+line2+line3

labels = [l.get\_label() for l in lines]

plt.legend(labels,loc='upper left')

#plt.plot(Lambda,Number\_Packet\_Dropped)

plt.xlabel("Arrival Rate Lambda (Packets/Second)")

plt.ylabel("Number\_Packet\_Dropped Probability")

plt.title("Number\_Packet\_Dropped Probability VS Arrival Rate When Max\_Buffer ")

plt.savefig("Number\_Packet\_Dropped\_Max\_Buffer.png")