ECE358: Computer Networks

Winter 2018

Project 1: Queue Simulation

Date of submission:

Submitted by: YuCheng Liu

Student ID: y698liu

Student name : Liu, YuCheng>

Waterloo Email address: y698liu@uwaterloo.ca

Marks received:

Marked by:

Table of Contents

[Distribution Simulation 3](#_Toc505004196)

[Question 1: How would you generate an exponential random variable with parameter from U (0,1)? 3](#_Toc505004197)

[M/M/1 Queue 3](#_Toc505004198)

# **Distribution Simulation**

## *Question 1: How would you generate an exponential random variable with parameter from U (0,1)?*

def checkMeanVariance(Lambda):

randomTime = [nextTime(Lambda) for i in xrange(1000)]

mean = sum(randomTime) / 1000

variance = [x-mean for x in randomTime];

variance = sum([x\*x for x in variance])/1000

expectedMean = 1/Lambda

expectedVariance = expectedMean/Lambda

print("Mean:" + str(mean) + " compare to " + str(expectedMean)+ "\n" + "Variance:" + str(variance)+ " compare to " + str(expectedVariance))

def nextTime(rateParameter):

return -math.log(1.0 - random.random()) / rateParameter

For Lambda = 75:

Lab1 $ python Lab1.py

Mean:0.0132434053893 compare to 0.0133333333333

Variance:0.000188891068463 compare to 0.000177777777778

Lab1 $ python Lab1.py

Mean:0.013032477334 compare to 0.0133333333333

Variance:0.000171351837958 compare to 0.000177777777778

Lab1 $ python Lab1.py

Mean:0.0132905588025 compare to 0.0133333333333

Variance:0.000173888385356 compare to 0.000177777777778

Lab1 $ python Lab1.py

Mean:0.0138144162272 compare to 0.0133333333333

Variance:0.000174434791397 compare to 0.000177777777778

Lab1 $ python Lab1.py

Mean:0.0132648395681 compare to 0.0133333333333

Variance:0.000161259948369 compare to 0.000177777777778

**M/M/1 Queue**

*Question 2: Build your simulator for this queue and explain in words what you have done.* Show your code in the report.

Before creating a DES, I have analyzed the system and created a few object class to make to system easier to understand and implement.

class packet:

def \_\_init\_\_(self,arrivalTime,packetSize,serviceTime=0,new\_dpTime=0):

self.arrivalTime = arrivalTime

self.packetSize = packetSize

self.serviceTime = serviceTime

self.departureTime = new\_dpTime

class observer:

def \_\_init\_\_(self,new\_observeTime):

self.observeTime = new\_observeTime

class event:

def \_\_init\_\_(self,new\_type,new\_time):

self.type = new\_type

self.time = new\_time

The packet class represents the packet in the system and has arrivalTime, packetSize, serviceTime, and departureTime for attributes.

The observer is an object used to check the state of the system and the observeTime will be generated according to a Poisson distribution.

The event class in a generalized class to keep track and trigger actions depending on the type of the events.

In order to create a DES for a simple queue with an infinite buffer, I have separated the work into different steps.

Step1(Generating the list of observers):

I created a function that generates a set of random observation times according to a Poisson distribution with input parameter α and T. The function will continuously generate observer until the arrival time of the observer is larger than the total simulation time T. In the end, the function will return a list of observers and the arrival time of those observers will be a Poisson distribution.

def generateObserverList(T,Alpha):

arrivalTime = nextTime(Alpha) #generate an arrival time

newObserver = observer(arrivalTime) # create a observer for the new arrival time

observerList = [newObserver]

while(arrivalTime < T): # check if this arrival is still in the time period

nextarrival = nextTime(Alpha) #generate an arrival time

arrivalTime += nextarrival #increment the time counter

newObserver = observer(arrivalTime) #create a observer for the new arrivaltime

observerList += [newObserver] #add the new observer to the observer list

return observerList

Step 2(Generating the list of packets):