Report on term project

"Simulation of Packet Queuing System with Partial TCP functionality"

A term project report submitted in fulfillment
of the project requiremtnts for
CpE 356 Computer Networks I

Submitted to
Prof. Hamed Alazmi

Prepared by
Sulaiman A. Alkandari
209111338

Computer Engineering Department
Kuwait University, Kuwait
May 14, 2014

Table of Contents

ABSTRACT

When one needs to decide which is the best low-cost router queuing functionality for any given environment, an analysis should be carried out in accordance to maintain the budget. This analysis would be most accurate if it is based on a simulation of the system and the environment. Consequently, a given specification is required to be simulated and predicted, therefore, the results would give a good indication of how well it will practically fit. The results predict that the probability of packet-loss events is very insignificant, and therefore, the system can be almost fully utilized with a small probability of queuing inefficiency in accordance to the lost packets. In addition, the simulator statistical results that has been obtained estimates that the theory applies substantially well for the followed specification. As a final referee, a better understanding of the code and future plans hold to simulate the results again but for a dynamic number of senders and real-time simulation.

1. Introduction

When one needs to decide which is the best low-cost router queuing functionality for any given environment, an analysis should be carried out in accordance to maintain the budget. This analysis would be most accurate if it is based on a simulation of the system and the environment. Consequently, a given specification is required to be simulated and predicted, therefore, the results would give a good indication of how well it will operate.

The main idea is to implement the communication and data flow of a TCP connection with multiple senders and only one receiver. The sending specification demands sending packets with Poisson random variables and serves them to the router with exponential distribution random values. The router implements the *m/m/1/N* algorithm.

Later on in this report, the computed statistical values of different queuing parameters from the simulation results will be exemplified and compared with the calculations of corresponding theoretical equations. A better explanation of the values will be demonstrated. Consequently, a conclusion will be drawn after studying the results.

2. Theoretical Approach

2.1. System Components and Behavior

A queuing system model has been implemented with fixed buffer size. The queue has been implemented to handle multiple packets received from multiple senders. Received packets would be delivered to only one receiver. In more precise words, an *m/m/1/N* system has been implemented; where the *m* parameters stand for the rate of packets arrival with Poisson rate lambda. The received packets would be served exponentially in the queue with rate mu. The third and forth parameters, *1* and *N*, stand for having only one router/queue and the size of the queue/buffer, respectively.

The previously mentioned queuing system would be implemented with a simulated protocol, TCP, over a simulation of the transport layer. More precisely, generation of packets and retransmission is to be simulated on the sender/receiver side, while the queue, in the simulated network layer, plays a crucial role in delivering the transmitted packets.

As for the congestion control, TCP *Tahoe* has been implemented, where the congestion window size would drop to one if packet-loss occurs (Kurose & Ross, 2013, P. 276). Otherwise, it would increase exponentially until the congestion widow size reaches the threshold value. Afterwards, the increment pattern will become linear as the size of window will be continually incremented by one.

2.2. Factors of System Performance Calculation

According to these general formulas provided below, one can derive the waiting time for the packet in both system and queue as well as the number of the already existing waiting packets.

This ratio should always be less than or equal to 1:

$$\wp = \frac{lambda}{mu}$$

The probability that the queue/server is idle and holds no packets is:

$$P_0 = \frac{1-\wp}{1-\wp^{N+1}}$$

Therefore, the utilization of the queue/server is described when the server is busy:

$$Utilization = 1 - P_0$$

After deriving and tracing the system at each point the packet arrives, a general equation can be given to find the probability that there is *n* packets in the queue:

$$P_n = \wp^n \times P_0$$

With that, we are able to derive the expected number of packets in both the system and the queue by the following equations respectively:

$$Ls = \frac{\wp}{(1-\wp)\times(1-\wp^{N+1})} \times [1 + N \times \wp^{N+1} - (N+1) \times \wp^N]$$

and

$$Lq = \lambda \times Wq$$

However, the *lambda effective* is represented as:

$$\lambda_{eff} = \lambda \times (1 - P_N)$$

Accordingly, expected waiting time in both the system and in the queue can be derived, respectively, as follows:

$$Ws = \frac{Ls}{\lambda_{eff}}$$

and

$$Wq = Ws - \frac{1}{mu}$$

3. Simulation Approach

3.1. Implementation structure and Programming

The system implements generation of packets with multiple attributes to store several data fields needed for statistical purposes. Hence, the senders, receiver, and the router shall store generated and delivered packets in their temporary queues.

Four classes have been constructed: Sender, Receiver, Packet, and Router. However, the simulation part has been done separately in the main part of the code. Python language is the language used for programming this simulator. According to the structure, the design can be depicted by the illustration below where the arrows indicate a sequence of packets being transmitted.

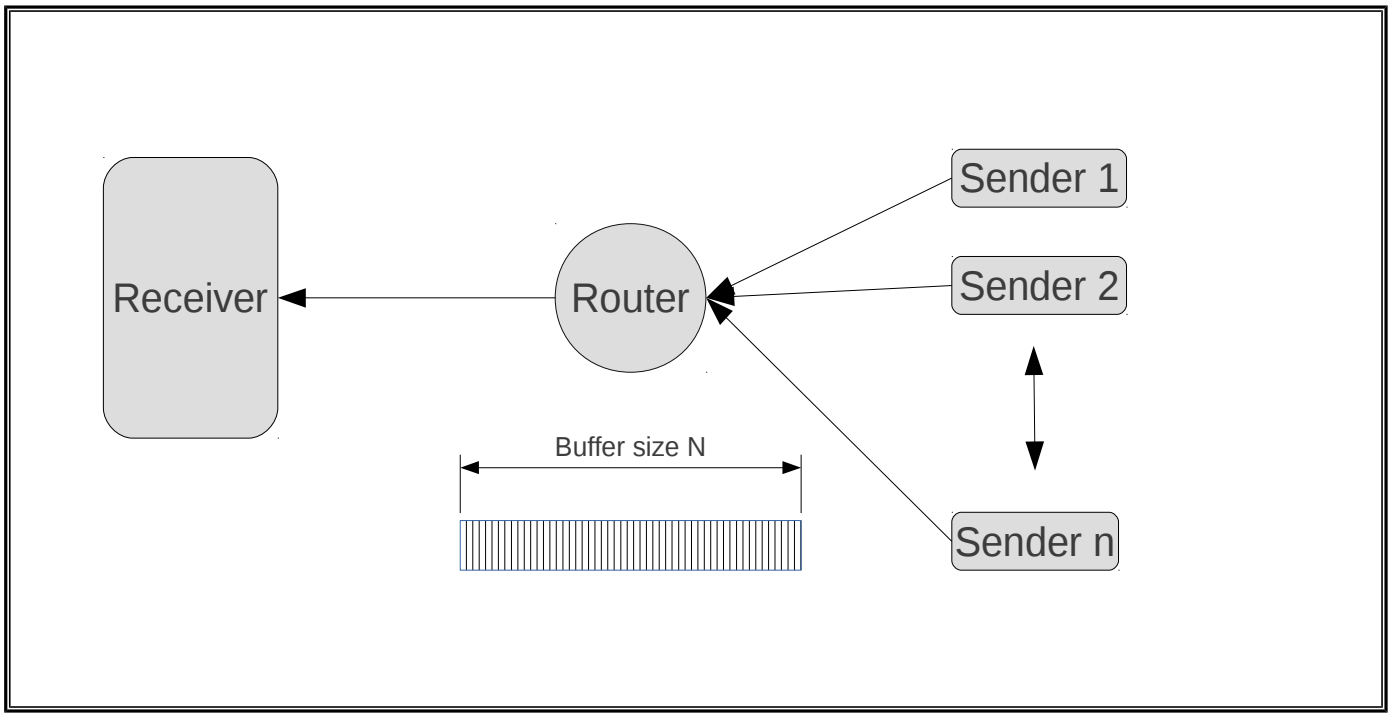

Figure 1: The Overall Design of the System

3.2. Simulation Results

The results are projected in two methods: plotting and statistical average values. The plots give approximate estimations of the real average statistical values which are provided in the following sections. The following criteria specify the starting points for the simulated results.

```
lam = 8.0
meu = 10.0
rough = lam / meu
N = 35
```

Data 1: Values of Arrival and Service Rates, and Buffer Size

3.2.1. Congestion Window Size

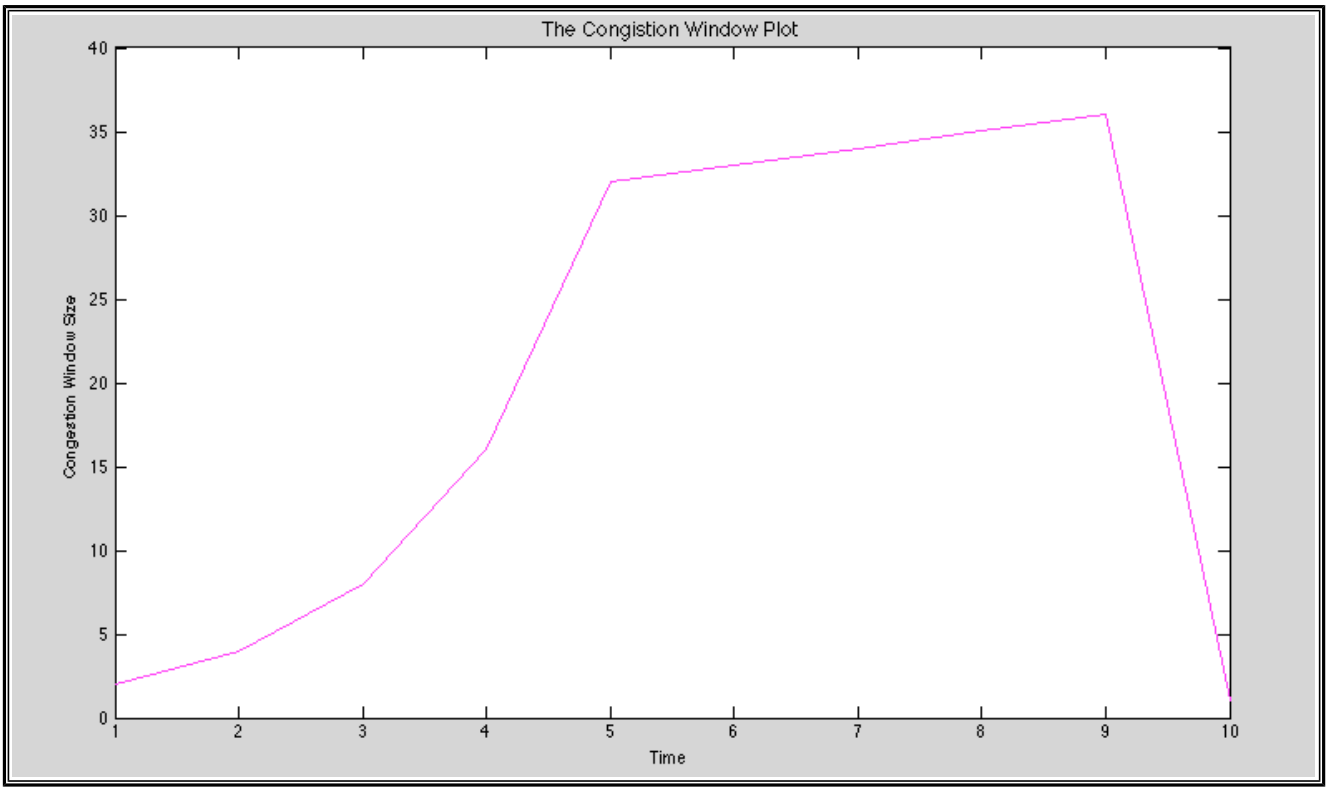

Figure 2: Congestion Window Plot

3.2.2. Utilization in Queue

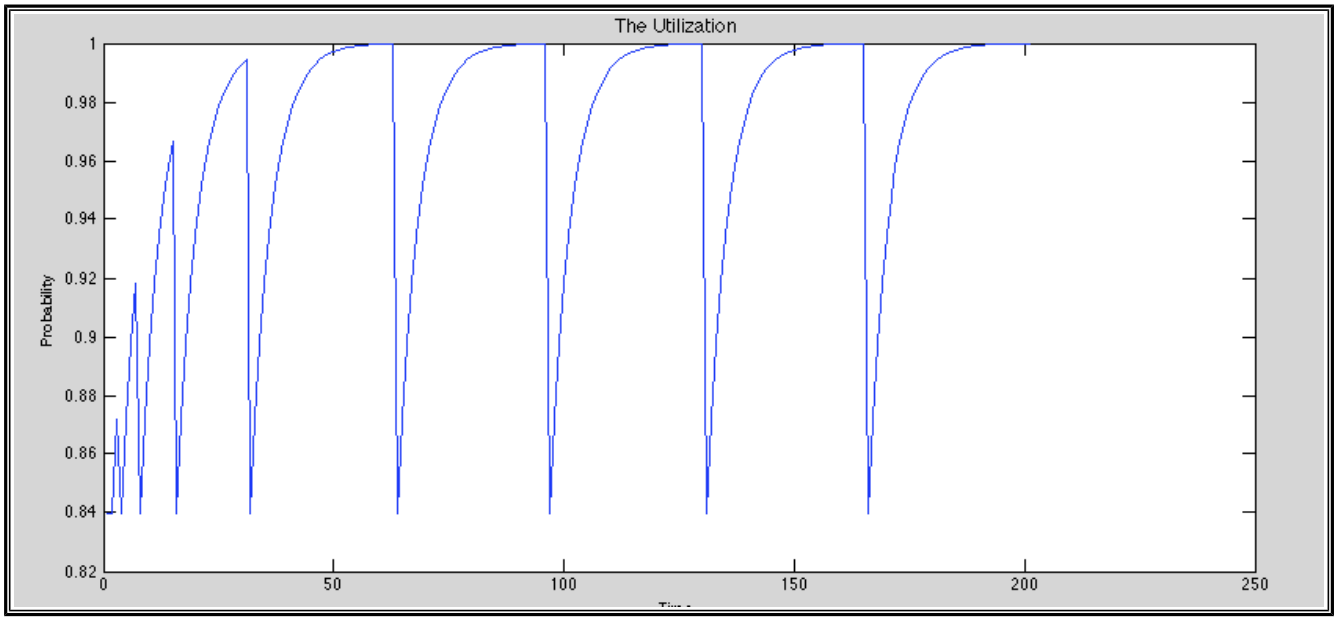

Figure 3: Utilization of Queue

3.2.3. Delay in Queue

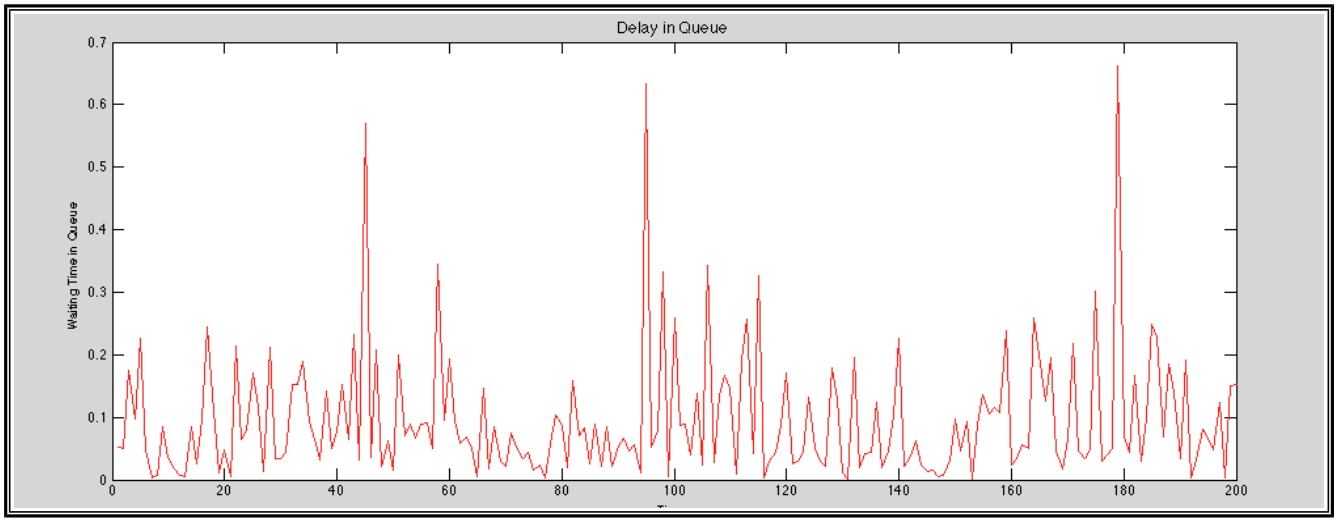

Figure 4: Delay in Queue

3.2.4. General Statistics

The probability that the server is idle: 0.200247894887
The probability that the server is full and cannot take any user: 8.12301965273e-05
The expected packets in system: 3.9883135396
The value of lambda effective: 7.99935015843
The expected waiting time in system: 0.498579692176
The expected waiting time in queue: 0.398579692176
The expected packets in queue: 3.18863753741
The dropping probability: 0.00497512437811
The expected utilization: 0.799752105113

Data 2: The Theoretical Statistics of the Queue

3.3. Observation and Discussion

Regarding the previously plotted and printed-out statistics, a comparison would be carried in order to examine, verify, and clarify the results.

As for implementation considerations, the methods and equations provided in the theoretical part demonstrate how exactly the program computes the stats. However, since the theory holds that the router will receive packets from multiple senders, it is intuitive that multiple packets will be received and stored temporarily in the router at nearly the same time. Consequently, the *n* senders will be equal to one in this implementation for simplicity purposes, and according to the TCP requirement, multiple packets will be sent at the same time with different rates. Therefore, the theory will still apply for this case.

3.3.1. Utilization

The expected utilization in the statistics is expected to be averaged to 0.8, while the value in the graph gives an estimation between 0.82 and 1 which is almost accurate to the average value calculated theoretically.

The graph repeats itself due to packet-loss events when the utilization reaches 1. The graph below illustrates this event.

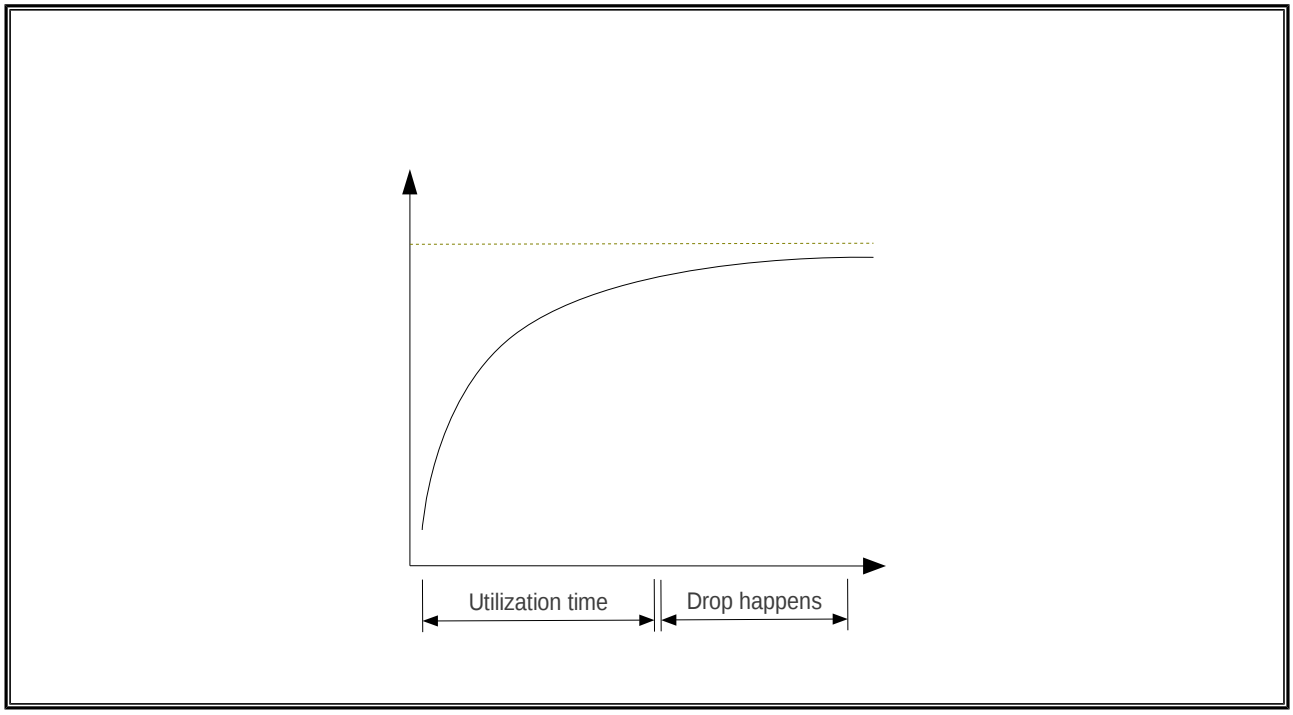

Figure 5: Theoretical Utilization of Queue

3.3.2. Delay

Delay, or expected waiting time in queue, is the time needed for the packet to leave the queue. Once the simulator generates the packet, it calculates a time for its arrival to the queue, and checks whether there are packets already waiting in the queue or not. According the that, it calculates its estimated time for the end of service. This time is the said to be the delay in the queue.

According to the theoretical results, the expected average waiting time in queue equals to 0.4 time unit. However, the values in the Delay graph gives estimated values between 0.05 and 0.6 which is close to the theoretical value.

4. Future Modification

The code demonstrates the simplest design for the TCP with *m/m/1/N* implementation for the queue where the number of senders is 1 ($n = 1$). Therefore, using dynamic list to hold multiple Sender objects would be a good engineering practice in this case.

As for the time estimated in the code, it has been represented as a function of Poisson, and exponential distribution values. Furthermore, this simulator would give better results if a real time is being used instead of approximate simulation of time.

5. Conclusion

Theoretical and experimental results can give better understanding to the problem of implementing a router with a specified queue for company or individual usage. Thus, when simulating a set of criteria of the given system, one can predict the loss of packets events that may occur during the process of sending packets from one host to another. We, as engineers, would conclude and decide on which is the best low-cost required buffer of a queuing system for such use.

Acknowledgments

I would like to express my special thanks to my colleges Mohammed Mustafa, Homoud Alsaleh, and Hamad Alghanim for their great help and tremendous advices.

References

Kurose, J. F. and Ross, K. W. (2013). "Computer Networking: *A Top-Down Approach*".6th ed.

APPENDIX A: Simulator code

#The Code:

```
#Sulaiman A. Alkandari
#209111338

import random
import numpy

#.............................................................class
packet
class Packet():
        #initiating instance variables:
        source = 0
        destination = 0
        ACK = False

        def __init__(self, arrival_time, service_start_time, service_time):
                self.arrival_time = arrival_time
                self.service_start_time = service_start_time
                self.service_time = service_time
                self.service_end_time = service_start_time + service_time
                self.delay = self.service_end_time - arrival_time

        #initiating setters:
        def set_source(self, value):
                self.source = value

        def set_destination(self, value):
                self.destination = value

        def set_ACK(self, value):
                self.ACK = value

        #initiating getters:
        def get_source(self):
                return self.source

        def get_destination(self):
                return self.destination

        def get_ACK(self):
                return self.ACK

#.....................................................................cla
ss sender

class Sender():
        # initiating the instance variables:
        retransmit = False
        address = 0
        out_queue = []
        in_queue = []
```

```
window_size = 1
window_increment = 1
threshold = 16
check_inorder_ACK = [ 0 for i in range(100)]

#initialize a constructor:
def __init__(self, address):
        self.address = address
        self.window_size = 1
        self.threshold = 16
#end of constructor

#generate setters:
def set_retransmit(self, value):
        self.retransmit = value

def set_address(self, value):
        self.address = value

def set_out_queue(self, value):
        self.out_queue.append(value)

def set_in_queue(self, value):
        self.in_queue.append(value)

def set_window_size(self, value):
        self.window_size = value

def set_threshold(self, value):
        self.threshold = value

def set_window_increment(self, value):
        self.window_increment = value

#generate getters:
def get_retransmit(self):
        return self.retransmit

def get_address(self):
        return self.address

def get_out_queue(self):
        return self.out_queue

def get_in_queue(self):
        return self.in_queue

def get_window_size(self):
        return self.window_size

def get_threshold(self):
        return self.threshold

def get_window_increment(self):
        return self.window_increment
```

```
#.....................................................................c
lass Receiver
class Receiver():

        #initiating instance variables:
        ACK_value = 0
        receiver_queue = []

        # initiating setters
        def set_ACK_value(self, value):
                self.ACK_value = value

        def set_receiver_queue(self, value):
                self.receiver_queue.append(value)

        #initiating getters:
        def get_ACK_value(self):
                return self.ACK_value

        def get_receiver_queue(self):
                return self.receiver_queue

#.....................................................................class
Router
class Router():

        #initiating instance variables
        number_of_lost_packets = 0
        expected_packets_in_queue = []
        droped_packets_queue = []
        average_delay_time_in_queue = []
        expected_utilization = []
        router_queue = []

        #generate setters:
        def set_number_of_lost_packets(self, value):
                self.number_of_lost_packets = value

        def set_expected_packets_in_queue(self, value):
                self.expected_packets_in_queue.append(value)

        def set_droped_packets_queue(self, value):
                self.droped_packets_queue.append(value)

        def set_average_delay_time_in_queue(self, value):
                self.average_delay_time_in_queue.append(value)

        def set_expected_utilization(self, value):
                self.expected_utilization.append(value)

        def set_router_queue(self, value):
                self.router_queue.append(value)

        #generate getters:
```

```
        def get_number_of_lost_packets(self):
                return self.number_of_lost_packets

        def get_expected_packets_in_queue(self):
                return self.expected_packets_in_queue

        def get_droped_packets_queue(self):
                return self.droped_packets_queue

        def get_average_delay_time_in_queue(self):
                return self.average_delay_time_in_queue

        def get_expected_utilization(self):
                return self.expected_utilization

        def get_router_queue(self):
                return self.router_queue


#the simulation
part.....................................................................

sender = Sender(1)
router = Router()
receiver = Receiver()
sender.set_threshold(16)

# lambda, and meu:
lam = 8.0
meu = 10.0
rough = lam / meu
probability_zero = (1.0 - rough) / (1.0 - pow(rough, 30))
total_packets = 0.0
droped_packets = 0.0

time = 0
end_time = 5000
arrival_time = 0
N = 35

while time <= end_time:
        #generating packets step
        ##############################################
        for i in range(0, sender.get_window_size()):
                if len(receiver.get_receiver_queue()) == 0:
                        arrival_time += numpy.random.poisson(lam)
                        service_start_time = arrival_time
                        service_time = random.expovariate(meu)
                        sender.set_in_queue(Packet(arrival_time,
service_start_time, service_time))
                        probability_n = pow(rough,
len(sender.get_in_queue())) * probability_zero
                        router.set_expected_utilization(1.0 -
probability_n)
                        total_packets += 1
```

```
                if len(receiver.get_receiver_queue()) != 0:
                        if len(sender.get_in_queue()) == 0:
                                arrival_time += numpy.random.poisson(lam)
                                service_start_time = max(arrival_time,
(receiver.get_receiver_queue()[-1].service_end_time))
                                service_time = random.expovariate(meu)
                                sender.set_in_queue(Packet(arrival_time,
service_start_time, service_time))
                                probability_n = pow(rough,
len(sender.get_in_queue())) * probability_zero
                                router.set_expected_utilization(1.0 -
probability_n)
                                total_packets += 1
                        else:
                                arrival_time += numpy.random.poisson(lam)
                                service_start_time = max(arrival_time,
sender.get_in_queue()[-1].service_end_time)
                                service_time = random.expovariate(meu)
                                sender.set_in_queue(Packet(arrival_time,
service_start_time, service_time))
                                probability_n = pow(rough,
len(sender.get_in_queue())) * probability_zero
                                router.set_expected_utilization(1.0 -
probability_n)
                                total_packets += 1
        ##############################################

        for j in range(0, N):
                if len(sender.get_in_queue()) != 0:

router.set_router_queue(sender.get_in_queue().pop(0))
                else:
                        break
        #.............


router.set_expected_packets_in_queue(len(router.get_router_queue()))

        if len(sender.get_in_queue()) != 0:
                sender.set_retransmit(True)
                router.set_droped_packets_queue(len(sender.get_in_queue()))
                sender.set_window_size(1)
                sender.set_window_increment(1)
                droped_packets += len(sender.get_in_queue())
        else:
                if sender.get_window_size() <= sender.get_threshold():
                        sender.set_window_size(pow(2,
sender.get_window_increment()))
                        router.set_droped_packets_queue(0)
                else:
                        sender.set_window_size(sender.get_window_size() +
1)
                        router.set_droped_packets_queue(0)
        #.............
```

```
        for z in range(0, N):
                if len(router.get_router_queue()) != 0:

receiver.set_receiver_queue(router.get_router_queue().pop(0))

        for i in range(0, len(sender.get_in_queue())):
                sender.get_in_queue().pop(0)

        #this is to check the progress on window size:
        sender.set_out_queue(sender.get_window_size())
        sender.set_window_increment(sender.get_window_increment() + 1)
        #incrementing the time of arrival by poisson(lambda)
        time += arrival_time
#end while loop

#calculating the statistics
ls = ((rough) / ((1.0 - rough) * (1.0 - pow(rough, (N + 1))))) * ((1.0 + (N
* pow(rough, (N + 1)))) - ((N + 1) * pow(rough, N)))
probability_N = pow(rough, N) * probability_zero
lam_eff = lam * (1 - probability_N)
ws = ls / lam_eff
wq = ws - (1.0/meu)
lq = lam * wq
dp = droped_packets / total_packets
u = 1.0 - probability_zero

#..........testing my results.................#
print '.........................((arrival time)).........................'
for i in receiver.get_receiver_queue(): print i.arrival_time
print '.........................((delay)).........................'
for i in receiver.get_receiver_queue(): print i.delay
print '.........................((lost packets)).........................'
for i in router.get_droped_packets_queue(): print i
print '.........................((window
progress)).........................'
for i in sender.get_out_queue(): print i
print '.........................((expected packets in
queue)).........................'
for i in router.get_expected_packets_in_queue(): print i
print '.........................((service end
time)).........................'
for i in receiver.get_receiver_queue(): print i.service_end_time
print '.........................((Expected
utilization)).........................'
for i in router.get_expected_utilization(): print i

print '\n\n\n'

#printing the statistics:
print 'The statistics: \n'
print 'The probability that the server is idle: ' + str(probability_zero)
print 'The probability that the server is full and cannot take any user: '
+ str(probability_N)
print 'The expected packets in system: ' + str(ls)
print 'The value of lambda effective: ' + str(lam_eff)
```

```
print 'The expected waiting time in system: ' + str(ws)
print 'The expected waiting time in queue: ' + str(wq)
print 'The expected packets in queue: ' + str(lq)
print 'The dropping probability: ' + str(dp)
print 'The expected utilization: ' + str(u)

#writing the data on a matlab file:
with open('Simulation_Results.m','w') as mat:
        mat.write('arrival_time = [')
        for i in receiver.get_receiver_queue():
mat.write(str(i.arrival_time) + ' ')
        mat.write(']; \n')

        mat.write('delay = [')
        for i in receiver.get_receiver_queue(): mat.write(str(i.delay) + '
')
        mat.write(']; \n')

        mat.write('congestion_window_size = [')
        for i in sender.get_out_queue(): mat.write(str(i) + ' ')
        mat.write(']; \n')

        mat.write('service_end_time = [')
        for i in receiver.get_receiver_queue():
mat.write(str(i.service_end_time) + ' ')
        mat.write(']; \n')

        mat.write('service_start_time = [')
        for i in receiver.get_receiver_queue():
mat.write(str(i.service_start_time) + ' ')
        mat.write(']; \n')

        mat.write('expected_utilization = [')
        for i in router.get_expected_utilization(): mat.write(str(i) + ' ')
        mat.write(']; \n')

        mat.write('subplot(1, 3, 1); \n')
        mat.write('plot(congestion_window_size, \'Color\', \'m\'); \n')
        mat.write('xlabel(\'Time\') \n')
        mat.write('ylabel(\'Congestion Window Size\') \n')
        mat.write('title(\'The Congistion Window Plot\', \'FontSize\',
12) \n')
        mat.write('subplot(1, 3, 2); \n')
        mat.write('plot(expected_utilization, \'Color\', \'b\'); \n')
        mat.write('xlabel(\'Time\') \n')
        mat.write('ylabel(\'Probability\') \n')
        mat.write('title(\'The Utilization\', \'FontSize\', 12) \n')
        mat.write('subplot(1, 3, 3); \n')
        mat.write('plot(delay, \'Color\', \'r\'); \n')
        mat.write('xlabel(\'Time\') \n')
        mat.write('ylabel(\'Waiting Time in Queue\') \n')
        mat.write('title(\'Delay in Queue\', \'FontSize\', 12) \n')
```

APPENDIX B: Statistics

Arrival Time	Delay	Service End Time	Utilization
9	0.0866174085	9.0866174086	0.8398016841
19	0.0194799387	19.0194799387	0.8398016841
31	0.0235306334	31.0235306334	0.8718413473
41	0.1586664723	41.1586664723	0.8398016841
50	0.1198240319	50.1198240319	0.8718413473
58	0.0502115458	58.0502115458	0.8974730778
64	0.1959063175	64.1959063175	0.9179784623
68	0.0229653805	68.0229653805	0.8398016841
73	0.0284245448	73.0284245448	0.8718413473
76	0.171790947	76.171790947	0.8974730778
86	0.1083625442	86.1083625442	0.9179784623
99	0.2466788692	99.2466788692	0.9343827698
101	0.1213078557	101.121307856	0.9475062158
112	0.0215680205	112.021568021	0.9580049727
123	0.0698307293	123.069830729	0.9664039781
128	0.0258096064	128.025809606	0.8398016841
137	0.0898239064	137.089823906	0.8718413473
139	0.1431653955	139.143165396	0.8974730778
144	0.0110466901	144.01104669	0.9179784623
149	0.0436585701	149.04365857	0.9343827698
159	0.0574428373	159.057442837	0.9475062158
174	0.1796548757	174.179654876	0.9580049727
183	0.0751770287	183.075177029	0.9664039781
192	0.0222210905	192.02222109	0.9731231825
203	0.1414379513	203.141437951	0.978498546
209	0.1038077647	209.103807765	0.9827988368
218	0.0449065332	218.044906533	0.9862390694
225	0.0753964839	225.075396484	0.9889912556
233	0.4348984729	233.434898473	0.9911930044
242	0.0521215026	242.052121503	0.9929544036
252	0.0331946135	252.033194614	0.9943635228
263	0.0954865028	263.095486503	0.8398016841
272	0.1516138415	272.151613842	0.8718413473
281	0.1056109508	281.105610951	0.8974730778
288	0.1132074972	288.113207497	0.9179784623
300	0.0101658874	300.010165887	0.9343827698
308	0.1055376524	308.105537652	0.9475062158
319	0.0422860898	319.04228609	0.9580049727
330	0.1998097632	330.199809763	0.9664039781
340	0.0144123999	340.0144124	0.9731231825
348	0.0045057215	348.004505722	0.978498546
360	0.290600428	360.290600428	0.9827988368
370	0.1819576047	370.181957605	0.9862390694
380	0.0961364264	380.096136426	0.9889912556
385	0.0059531197	385.00595312	0.9911930044
389	0.0624174559	389.062417456	0.9929544036
394	0.0268647407	394.026864741	0.9943635228
397	0.2075888103	397.20758881	0.9954908183

405	0.0189054704	405.01890547	0.9963926546
414	0.0084398343	414.008439834	0.9971141237
426	0.5640018251	426.564001825	0.997691299
438	0.2348771422	438.234877142	0.9981530392
451	0.0764568617	451.076456862	0.9985224313
461	0.0651736035	461.065173604	0.9988179451
471	0.2150203342	471.215020334	0.9990543561
475	0.1036895784	475.103689578	0.9992434848
483	0.0875374294	483.087537429	0.9993947879
499	0.0558436308	499.055843631	0.9995158303
504	0.0378899593	504.037889959	0.9996126642
511	0.0999684518	511.099968452	0.9996901314
517	0.1310791719	517.131079172	0.9997521051
525	0.0077412652	525.007741265	0.9998016841
532	0.103849707	532.103849707	0.9998413473
544	0.1863922275	544.186392228	0.8398016841
555	0.1578858679	555.157885868	0.8718413473
565	0.0183298188	565.018329819	0.8974730778
572	0.0019723193	572.001972319	0.9179784623
579	0.0649658744	579.064965874	0.9343827698
584	0.2618086002	584.2618086	0.9475062158
594	0.0036810087	594.003681009	0.9580049727
599	0.0625054036	599.062505404	0.9664039781
605	0.0111114806	605.011111481	0.9731231825
613	0.0658391677	613.065839168	0.978498546
617	0.0502937491	617.050293749	0.9827988368
627	0.149240707	627.149240707	0.9862390694
635	0.0421868856	635.042186886	0.9889912556
647	0.0040091136	647.004009114	0.9911930044
655	0.1460534604	655.14605346	0.9929544036
663	0.1760630387	663.176063039	0.9943635228
678	0.0163121916	678.016312192	0.9954908183
686	0.0042875631	686.004287563	0.9963926546
694	0.1576181252	694.157618125	0.9971141237
706	0.0074359465	706.007435946	0.997691299
712	0.0883069531	712.088306953	0.9981530392
720	0.1207221133	720.120722113	0.9985224313
730	0.1892677486	730.189267749	0.9988179451
739	0.0183548603	739.01835486	0.9990543561
747	0.2742171518	747.274217152	0.9992434848
753	0.0640618439	753.064061844	0.9993947879
761	0.1091863756	761.109186376	0.9995158303
763	0.1815979615	763.181597961	0.9996126642
770	0.0048930836	770.004893084	0.9996901314
779	0.1205738928	779.120573893	0.9997521051
785	0.0144015836	785.014401584	0.9998016841
795	0.0049634446	795.004963445	0.9998413473
799	0.0530439167	799.053043917	0.9998730778
806	0.0046387864	806.004638786	0.8398016841

819	0.085198801	819.085198801	0.8718413473
825	0.0842903134	825.084290313	0.8974730778
834	0.1924086857	834.192408686	0.9179784623
841	0.2353459047	841.235345905	0.9343827698
847	0.0358451471	847.035845147	0.9475062158
854	0.1108677516	854.110867752	0.9580049727
862	0.1303982947	862.130398295	0.9664039781
867	0.011727198	867.011727198	0.9731231825
873	0.0291747648	873.029174765	0.978498546
880	0.1580302533	880.158030253	0.9827988368
887	0.0533272348	887.053327235	0.9862390694
899	0.012556815	899.012556815	0.9889912556
910	0.0001353861	910.000135386	0.9911930044
918	0.1009626653	918.100962665	0.9929544036
928	0.2738080569	928.273808057	0.9943635228
934	0.027617799	934.027617799	0.9954908183
942	0.0741545169	942.074154517	0.9963926546
946	0.0404286134	946.040428613	0.9971141237
950	0.0632991943	950.063299194	0.997691299
958	0.0107624998	958.0107625	0.9981530392
968	0.0137112583	968.013711258	0.9985224313
974	0.3141442307	974.314144231	0.9988179451
983	0.1918297627	983.191829763	0.9990543561
994	0.1710830276	994.171083028	0.9992434848
1001	0.03711064	1001.03711064	0.9993947879
1007	0.1477050162	1007.14770502	0.9995158303
1016	0.1324749901	1016.13247499	0.9996126642
1027	0.0389383785	1027.03893838	0.9996901314
1036	0.016794432	1036.01679443	0.9997521051
1040	0.0441587127	1040.04415871	0.9998016841
1054	0.0722491946	1054.07224919	0.9998413473
1058	0.0661575682	1058.06615757	0.9998730778
1060	0.0685459329	1060.06854593	0.9998984623
1070	0.0107931317	1070.01079313	0.8398016841
1076	0.0576121404	1076.05761214	0.8718413473
1084	0.0614041211	1084.06140412	0.8974730778
1090	0.1754345258	1090.17543453	0.9179784623
1098	0.1405052618	1098.14050526	0.9343827698
1106	0.0198894293	1106.01988943	0.9475062158
1118	0.0429045968	1118.0429046	0.9580049727
1128	0.1443477462	1128.14434775	0.9664039781
1136	0.1474260117	1136.14742601	0.9731231825
1138	0.1299132976	1138.1299133	0.978498546
1151	0.2736201819	1151.27362018	0.9827988368
1161	0.0358722607	1161.03587226	0.9862390694
1170	0.2428416745	1170.24284167	0.9889912556
1179	0.0267621977	1179.0267622	0.9911930044
1189	0.0759814918	1189.07598149	0.9929544036
1201	0.0439965842	1201.04399658	0.9943635228

1205	0.041562805	1205.0415628	0.9954908183
1217	0.0054111608	1217.00541116	0.9963926546
1227	0.0544248966	1227.0544249	0.9971141237
1234	0.1215713132	1234.12157131	0.997691299
1237	0.143938468	1237.14393847	0.9981530392
1249	0.0234668371	1249.02346684	0.9985224313
1257	0.0065858744	1257.00658587	0.9988179451
1271	0.3731277037	1271.3731277	0.9990543561
1281	0.2282509548	1281.22825095	0.9992434848
1286	0.0122058584	1286.01220586	0.9993947879
1296	0.0141293943	1296.01412939	0.9995158303
1307	0.2278770456	1307.22787705	0.9996126642
1318	0.0423303551	1318.04233036	0.9996901314
1325	0.0864851617	1325.08648516	0.9997521051
1331	0.1804012437	1331.18040124	0.9998016841
1341	0.1132109137	1341.11321091	0.9998413473
1350	0.1725030649	1350.17250306	0.9998730778
1362	0.0294873875	1362.02948739	0.9998984623
1372	0.1162149707	1372.11621497	0.9999187698
1380	0.0095743311	1380.00957433	0.8398016841
1385	0.0118144033	1385.0118144	0.8718413473
1394	0.0378879997	1394.037888	0.8974730778
1401	0.1164164598	1401.11641646	0.9179784623
1409	0.1466791936	1409.14667919	0.9343827698
1411	0.3675039882	1411.36750399	0.9475062158
1418	0.0687151192	1418.06871512	0.9580049727
1426	0.0338215254	1426.03382153	0.9664039781
1438	0.4585301755	1438.45853018	0.9731231825
1450	0.1201987663	1450.12019877	0.978498546
1453	0.1226855124	1453.12268551	0.9827988368
1460	0.0039563089	1460.00395631	0.9862390694
1469	0.1499988013	1469.1499988	0.9889912556
1481	0.0744684151	1481.07446842	0.9911930044
1490	0.0725100141	1490.07251001	0.9929544036
1504	0.0176413049	1504.0176413	0.9943635228
1515	0.0639566228	1515.06395662	0.9954908183
1521	0.103641443	1521.10364144	0.9963926546
1526	0.0527047632	1526.05270476	0.9971141237
1534	0.2010263052	1534.20102631	0.997691299
1545	0.0242470919	1545.02424709	0.9981530392
1553	0.0969861822	1553.09698618	0.9985224313
1562	0.0841763007	1562.0841763	0.9988179451
1567	0.0441919588	1567.04419196	0.9990543561
1577	0.0542613097	1577.05426131	0.9992434848
1583	0.057948294	1583.05794829	0.9993947879
1590	0.043539957	1590.04353996	0.9995158303
1595	0.0523027548	1595.05230275	0.9996126642
1603	0.0457277993	1603.0457278	0.9996901314
1613	0.0386452245	1613.03864522	0.9997521051

1622	0.0826931355	1622.08269314	0.9998016841
1627	0.2469505731	1627.24695057	0.9998413473
1638	0.388428104	1638.3884281	0.9998730778
1646	0.0064649159	1646.00646492	0.9998984623
1652	0.0164331942	1652.01643319	0.9999187698
			0.9999350158