ECE 358: Computer Networks

Fall 2014

Project 1: M/D/1 and M/D/1/K Queue Simulation

Date of Submission: October 5, 2014

Submitted by:

20414514:	Louisa Chong:	l5chong@uwaterloo.ca
20434209:	Simarpreet Singh	s244sing@uwaterloo.ca

Marks received:

Marked By:

Table of Contents

I. M/D/1 Queue

QUESTION 1:

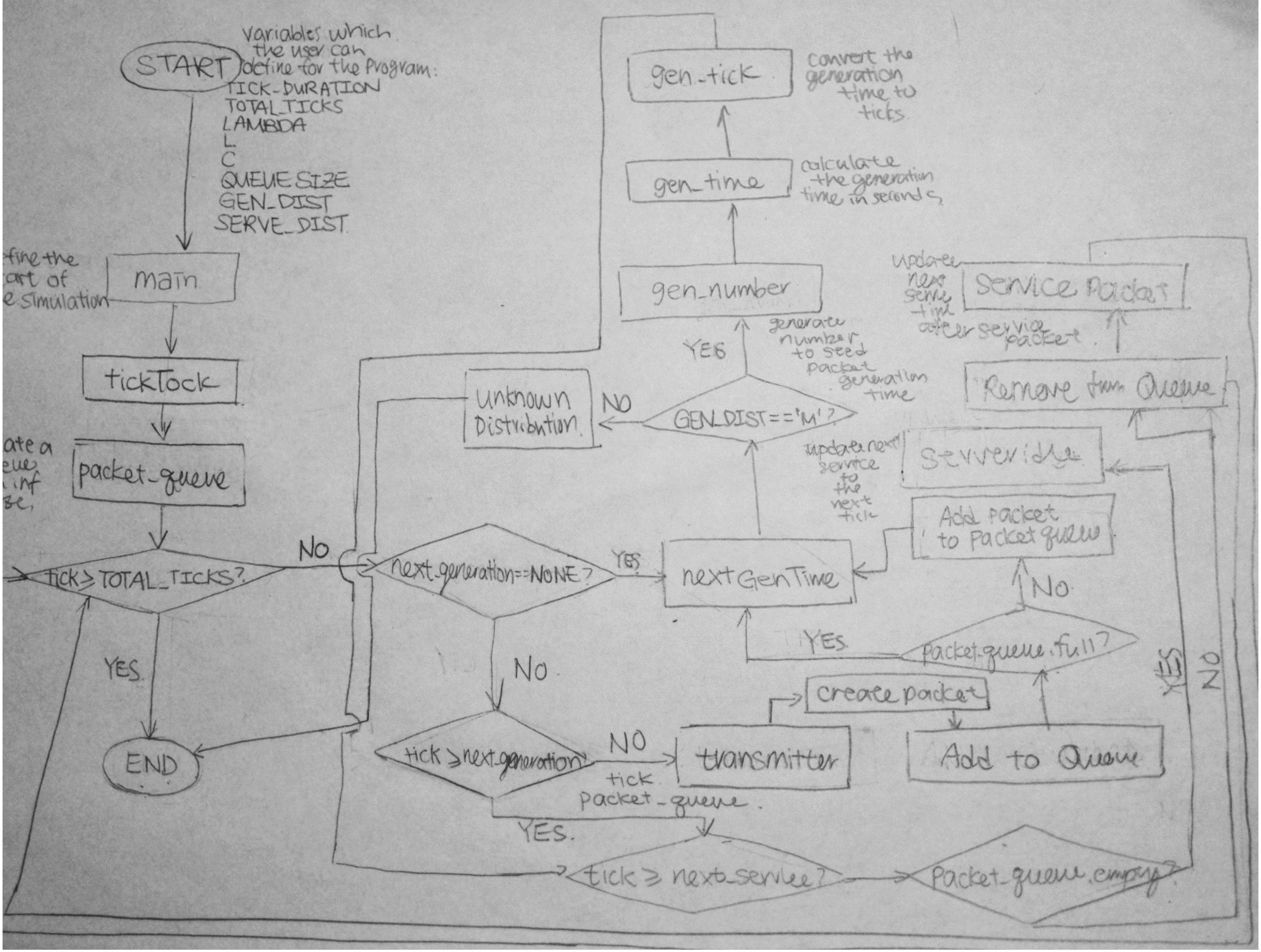
Description:

The program we wrote accepts eight variables. The first variable is the tick duration, which defines the time that a tick represents in real time. The second variable is the total ticks, this variable represents the total number of ticks we want our program to run for. Lambda is the variable, which define the average number of packets generated per second. The variable L is the length of packet and C defines the rate which the server can service the packets. These variables are used to determine the time, which the server takes to service the packet. The sixth variable is the queue size; this is a buffer where the transmitter will try to send a packet to and where the server will receive a packet. Lastly, the generation distribution and service distribution which define the rate of which packet is generated and serviced.

Using these variables, the simulation will start from main. Main function will call the tick tock, which defines how many discrete ticks that the function will run for. The tick tock functions runs a for loop, for a total of total ticks specified. In each tick, the tick tock function will check if it is time to generate a new packet or receive a packet from the buffer. In order to perform the transmission and service, the tick tock function will call a transmit function and service function. When the generation time is reached, the tick tock function will call the transmit function and a new packet will be generated and add to the buffer. After a new packet is generated, we will recalculate a new generation time. Similarly, when the service time is reached, the tick tock function will call the service function and serve the packet at the top of the buffer. After receiving the packet from the buffer, we will recalculate a new service time.

The calculation of the generation time and the service time are extracted into functions. To calculate the generation time, we first determine what distribution we will be using. For this lab, we first verify the generation distribution we are using is the Markovian distribution. Knowing the distribution, we calculate the interval, which we want to generate the new packet. With the interval and the current time, we will return the next generation time. To calculate the generation time, we will determine the distribution we will be using. If the distribution is deterministic, we calculate the service interval using the rate of which the server is serving the packets and the packet length. With the interval and the current time, we will return the next service time.

In the following flow chart, it shows the overall flow of the program.


START
variables which the user can define for the Program:
TICK-DURATION
TOTAL_TICKS
LAMBDA
L
C
QUEUE SIZE
GEN_DIST
SERVE_DIST
main
fine the
art of
e simulation
tickTock
packet_queue
ate a
eue,
inf
se,
tick ≥ TOTAL_TICKS?
YES.
NO
END
next_generation == NONE?
YES
No
Unknown Distribution.
NO
GEN_DIST == 'M'?
YES
gen_number
generate number to seed packet generation time
gen_time
calculate the generation time in seconds,
gen_tick
convert the generation time to ticks.
nextGenTime
update next service to the next tick
tick ≥ next_generation
NO
tick
packet_queue.
YES.
transmitter
create packet
Add to Queue
packet_queue.full?
YES.
No
Add packet to Packet queue
Server idle
Remove from Queue
Service Packet
update next serve time after service packet.
tick ≥ next_service?
Packet_queue.empty?
NO

QUESTION 2:

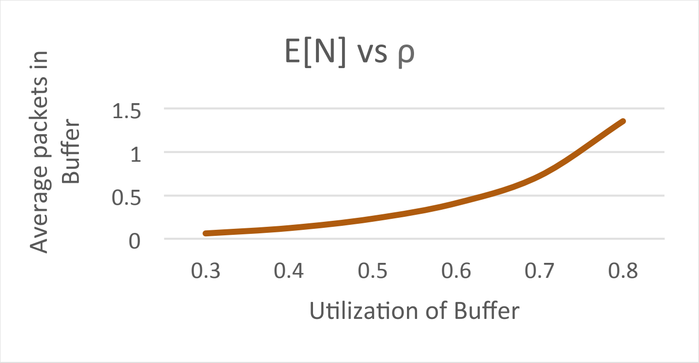
QUESTION 2								
Lambda	**Total of Ticks**	**Packet Generate**	**Packet Received**	**Packet Dropped**	**Server Idle**	**E[N]**	**E[T] (ticks)**	**E[T] (seconds)**
100	**10000000**	**9848**	**9848**	**0**	**8030400**	**0.0243324**	**224.707961**	**0.022396285**
100	**10000000**	**10013**	**10013**	**0**	**7997400**	**0.023994**	**223.9628483**	**0.002239628**
100	**10000000**	**10005**	**10005**	**0**	**7999000**	**0.0246306**	**224.6182909**	**0.002246183**
100	**10000000**	**9954**	**9954**	**0**	**8009200**	**0.0244828**	**224.5959413**	**0.002245959**
100	**10000000**	**9943**	**9943**	**0**	**8011400**	**0.0224857**	**222.6146032**	**0.002226146**

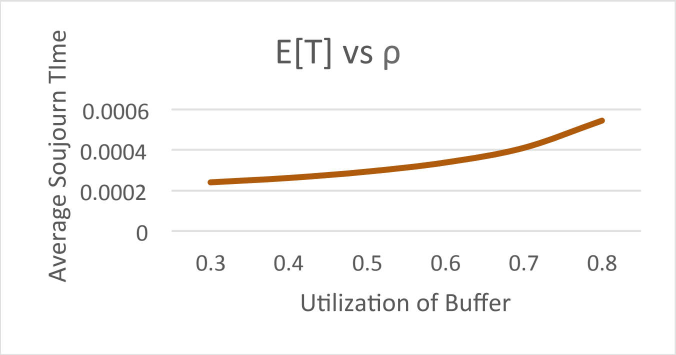
Observations:

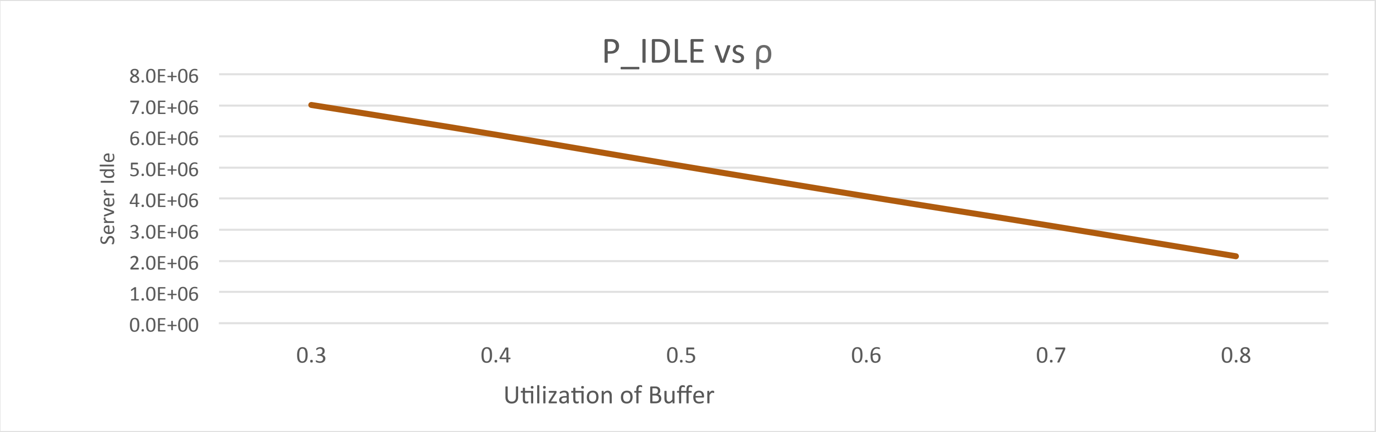
1) Average number of packets generated @ Lambda = 100: 9952.6 ~ 9953 packets.
2) Average number of packets received @ Lambda = 100: 9952.6 ~ 9953 packets.
3) Average number of Server Idle time = 8009480
4) Average soujourn time = 0.00627084 seconds.

QUESTION 3:

	QUESTION 3								
rho	**Lambda**	**Total of Ticks**	**Packet Generate**	**Packet Received**	**Packet Dropped**	**Server Idle**	**E[N]**	**E[T]**	**E[T] (seconds)**
0.3	**150**	**10000000**	**149001**	**149001**	**0**	**7019980**	**0.0600684**	**24.03140919**	**0.000240314**
0.4	**200**	**10000000**	**197291**	**197291**	**0**	**6054180**	**0.1222863**	**26.19827057**	**0.000261983**
0.5	**250**	**10000000**	**247343**	**247343**	**0**	**5053146**	**0.2305827**	**29.32238632**	**0.000293224**
0.6	**300**	**10000000**	**296164**	**296164**	**0**	**4076729**	**0.4092324**	**33.81776313**	**0.000338178**
0.7	**350**	**10000000**	**343635**	**343635**	**0**	**3127329**	**0.7228962**	**41.03676586**	**0.000410368**
0.8	**400**	**10000000**	**392466**	**392466**	**0**	**2150680**	**1.3533265**	**54.48264308**	**0.000544826**







Observations:

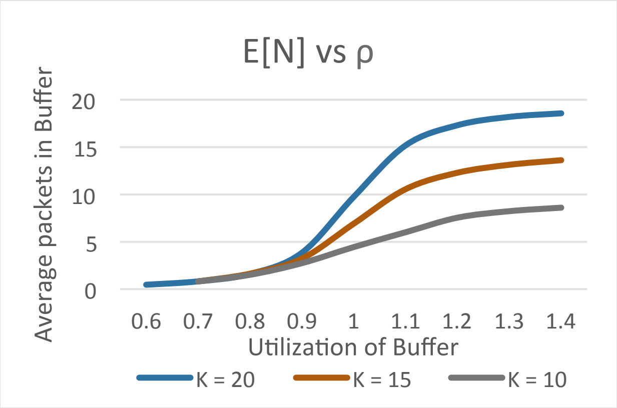
1) As the value of rho increases, the value of lambda also increases proportionally.
2) As Lambda increases the number of packets generated also increases.
3) There is no loss of packets as the queue is infinite.
4) The Server is less idle as the number of packets increases.
5) E[N] represents the average number of packets in queue, which increases with increased lambda.
6) E[T] or the average soujourn time also increases as it takes longer with more packets in queue to queue them and service them.

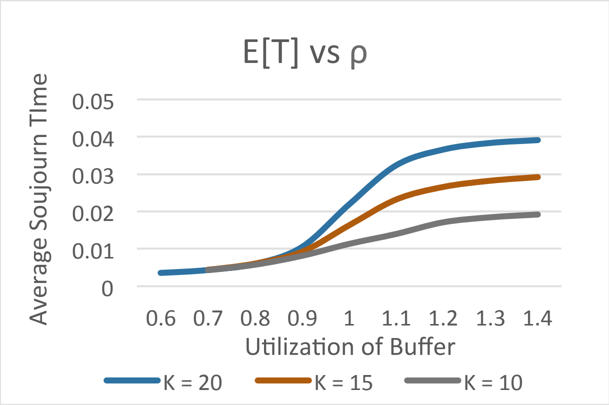
II. M/D/1/K Queue

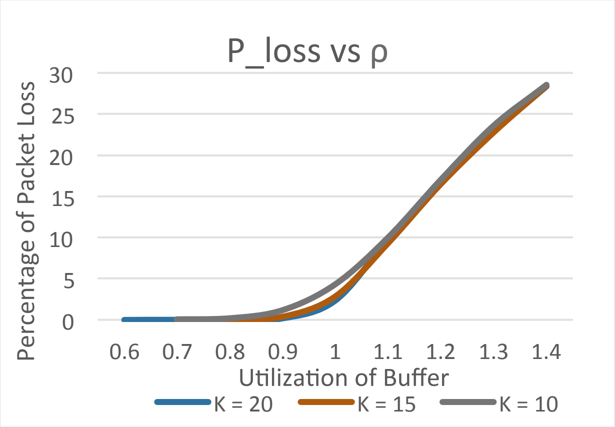
Question 4:

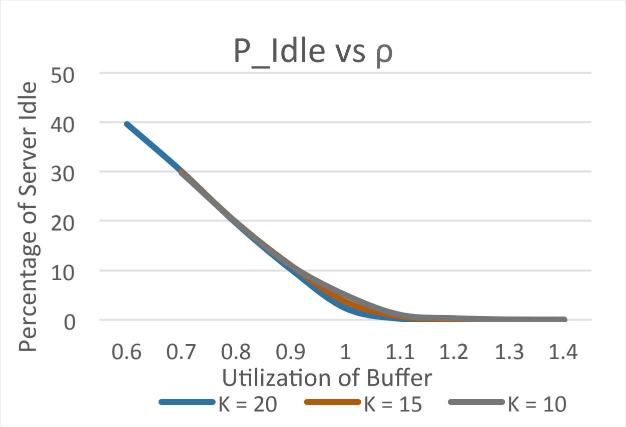
The system for the M/D/1/K queue is really similar to M/D/1. The only change we made for this function, was checking the queue size. If the queue size is not infinity, then we will define a queue with the size defined. If the queue size is infinity, we will just define a queue with no fixed size

Question 5:









In the average number of packets in the buffer versus utilization of the queue plot, we observed the average packages in buffer increases when the utilization of buffer increases. The number of packets in the queue increases rapidly when the buffer utilization is between 0.9 and 1.2. When the buffer utilization is greater than 1.2, the average number of packets in the buffer reach a saturation. The saturation value is dependent of the buffer size. When the buffer size is 10, the curve saturates around 9 packets in the buffer on average. Similarly, the curve saturates around 14 packets in the buffer on average when buffer size is

15. Lastly, the curve saturates around 19 packets in the buffer when buffer size is 20. The average soujourn time of a packet is porportional to the average number of packets in the buffer. In the average sojourn time versus utilization of the queue plot, we also observed a rapid increase of sojourn time when the utilization of buffer increases. The average sojourn time increases rapidly when the buffer utilization is between 0.9 and 1.2. When the buffer utilization is greater than 1.2, the soujourn time of a packet reaches a saturation. The saturation value is dependent of the buffer size. When the buffer size is 10, the packet takes around a total 0.02 seconds. When the buffer size is 15, the packets take a total of 0.03 seconds to queue and served. Lastly, the sojourn time of a packet 0.04 seconds when the buffer size is 20. We observed a saturated value of sojourn time due to limited queue size. When the buffer is full, a packet will need to wait for all packets in the queue to be served before it can be served. Therefore, the maximum sojourn time will take the total service time for the maximum packets in the buffer. When the utilization of the buffer is low (around 0.3 to 0.5) the number of drop packets is almost 0 and the server idle time is really high. This is because the lack of packets which the receiver needs to process, the server have enough time to process a packet before a second packet arrives. This result in an empty buffer. When the utilization increase, the average numbers of packets increase. When queue is full, packets start dropping. We can see the number of dropped packets increases in an exponential matter when the buffer utilization is greater than 0.9. This is because the buffer is, usually, full. The packet which can be transmitted to the server is determined by the rate of the receiver. On the other hand, when the buffer utilization is low, the receiver is idle more often. The server is idle 40% of the time when the buffer utilization is around 0.3. As the number of packets transmitted increases, the percentage of server idle decreases rapidly. When buffer utilization is between 0.4 and 1.1, the server is rarely idle. When the buffer utilization reaches 1, the server idle percentage is almost 0. This is because, the buffer is full most of the times and the server is constantly serving packets from the buffer.