Computer Networks
Programming Assignment # 1
M/M/1 Queue Simulation

Problem Description->

To simulate queuing behavior, and determine whether an M/M/1 analysis is adequate and acceptable. We will do it for various sets of queue properties -

Part 1 -->

For this case, we will work with the following set of assumptions -->

(a) Messages arrive at a transmitter as per Poisson process and each message is exponentially distributed in length.
(b) The transmitter has a fixed channel capacity.
(c) There is a single queue, with unlimited number of buffers.

Part 2 -->

For this case, we will work with the following set of assumptions -->

(a) Messages arrive at a transmitter as per Poisson process and each message is exponentially distributed in length.
(b) The transmitter has a fixed channel capacity.
(c) There is a single queue, with a finite buffer capacity.

Part 3 -->

For this case, we will work with the following set of assumptions -->

(a) FOUR sources generate data at some peak rate rate (and no more), but the average is much lower than the peak. Also the inter-arrival time for each of them is different, but uniformly distributed .Messages arrive at a transmitter from all of these generators , get into a common queue and are then served by the server. Each message is of fixed length.
(b) The transmitter has a fixed channel capacity.
(c) There is a single queue, with a finite buffer capacity.

Notations used -->

AT --> Array storing the inter-arrival time of packets/customers.
CAT --> Array storing the cumulative arrival-time of packets/customers.
CDT --> Array storing the cumulative departure time of packets/customers.
QL --> Array storing the queue length of packets/customers when the kth customer arrives.
ST --> Array storing the service time of packets/customers.
IDT --> Array storing the idle time of server before the kth customer arrived.

Clock --> A variable keeping the time.
NAT --> A variable storing the next arrival time.
NDT --> A variable storing the next departure time.

Basic Algorithm -->

1. Generate and store N inter-arrival and service times as arrays AT[k], ST[k] for k = 1 to N.
2. Set AT[1]=0, and compute cumulative arrival times CAT[k] for k = 1 to N. Set IDT[k] = 0 and QL[k] = 0 for k = 1 to N.
3. Initialize I = 2, J = 1, NAT = CAT[2], NDT = CDT[1] = ST[1], Clock =0.
4. If I is less than equal to N, calculate the difference DIF between NAT and NDT else go to Step 9.
5. If DIF < 0 then
   Clock = NAT
   NAT = CAT[I+1]
   QL[i] = I - J
   Increment I
   Go back to Step 4.
6. If DIF = 0 then
   QL[i] = QL[i-1]
   Go to Step 8
7. If DIF > 0 then
   If ((I-J) >1) then
   Go to Step 9
   Else
   IDT[I] = DIF
   Go to Step 8
8. Clock = NAT
   NAT = CAT[I+1]
   CDT[J+1] = CAT[J] + ST[J+1]
   NDT = CDT[J+1]
   Increment J and I
9. Clock = NDT
   CDT[j+1] = CDT[j] + ST[J+1]
   NDT = CDT[J+1]
   Increment J
10. If J < N then go to Step 4 else go to Step 11.
11. Compute and Print the needed Statistics.

Implementation Details-->

1. Random number Generator ->

For Poison distribution, we use (((-1)/Lambda)* log(Uniformly distributed Variable varying between 0 and 1.0). Such a uniformly distributed variable is provided in Java as NextDouble in the library file for random number generation. For having a uniform distribution with peak rate much higher than average rate, we take the limits as (1/peak_rate) and ((2/average_rate)-(1/peak_rate)). This will ensure that the maximum inter-arrival time is 1/peak_time and that the average inter-arrival time will be 1/average_rate.

2. Statistics Drawn ->

The following statistics have been drawn from the algorithm-->

Simulated and actual arrival rate.
The average length of a non-empty queue
The maximum length of a queue
The maximum waiting time of a customer
The total idle time of the server
No. of Packets dropped

3. Finite Size of Buffer ->

To implement the finite size of buffer we make the service time of that element as zero and prevent that element from increasing queue size. This will happen if the QL==Buffer_size.

4. Merging the generators ->

For this we will make 4 arrays of AT and CAT, corresponding to each of the generator. After calculating the CAT, we merge the 4 arrays by keeping a pointer on each of them and incrementing that pointer as we take that particular value in our TCAT array. This will take O(n) time. This is the simulation of the actual situation as there also the four generators will generate jobs/packets/consumers independently of each other but in the queue they will all merge.