Department of Electrical and Computer Engineering

ECE 358: Computer Networks

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

Table of Contents

1 OBJECTIVE

On the Internet, messages are queued up in routers and host computers, and queues impact the end-to-end delay of the messages. Therefore, it is important to understand the impact of various network parameters on network delay. The objective of this lab work is to design a simulator to study the behaviour of two basic types of queues (M/M/1 and M/M/1/K). After this experiment, students are expected to learn:

- the basic elements of a network simulator; and
- the behaviour of a single buffer queue with different parameters.

2 EQUIPMENT

A computer/ laptop with Python installed. Python must be used for this lab, specifically Python 3. The exact version is not important provided you don't use any advanced features, but TAs are trained to use Python 3.10 and later.

3 OVERVIEW

The performance of a communication network is a key aspect of network engineering. Delay and packet loss are two important performance metrics of computer networks. Delay and packet loss are directly related to queues in a computer network. Delay is a measure of the time it takes the packet to reach from its source to its destination. A major part of this delay is the queuing delay, which is the delay that is incurred while the packet is waiting its turn in the queue to be processed. Packet loss is the percentage of data packets that are lost in the system. Packet losses happen when a packet is received in a finite queue which is already full. In this case the queue does not have enough buffer to store the packet and the packet has to be dropped. The performance of computer networks can be evaluated by using network models. The three common types of network models are *analytical* (i.e., mathematical) models, *simulation* models, and *prototype implementation* models. It is difficult to build exact analytical models of complex systems and expensive to build prototype implementations. Therefore, we try to get an approximate analytical model which is validated with a *simulation* model.

Simulation is not only useful for validating approximate solutions but also in many other scenarios. During the design of a system, it allows us to compare potential solutions at a relatively low cost. It is also very useful to dimension a system, i.e., to decide how much resources to allocate to the system based on a-priori knowledge of the inputs. In addition, it is used to check how potential modifications of an existing system could affect the overall performance. To perform a simulation, we need a model of the system, as well as models for input(s). In this project, you are going to simulate and understand the behaviour of two basic types of queues.

4 BACKGROUND MATERIAL

4.1 Kendall Notation

Queues are typically described using Kendall notation of the form: A/S/C/K/D, where

- A corresponds to the arrival process of the packets (more precisely the distribution of the inter-arrival time between packets). A can be M (Markovian), D (Deterministic) or G (General).
- S corresponds to the server process for the packets. S can again be M, D or G.
- C corresponds to the number of servers (or network interfaces).
- K corresponds to the buffer size, i.e. the number of packets that can be accommodated in the "waiting room". If omitted, the buffer is assumed to be infinitely large.

- D corresponds to the queueing discipline, which is almost always FIFO (First-In-First-Out) see Figure 1. and is therefore often omitted from the notation.

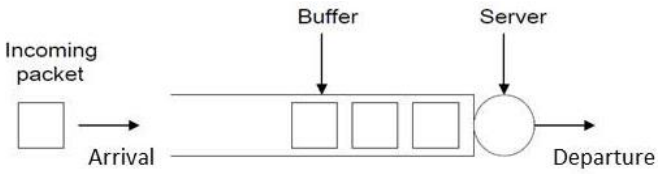

Figure 1: Model of a queue.

For example, M/M/c queue means that both the arrival and service processes are Markovian and that there are *c* servers. For the purposes of this lab, a Markovian arrival process means that the arrival process follows a Poisson distribution. In a Poisson distribution, the distribution of the time between successive arrivals (also called inter-arrival time) is modeled via an exponential distribution. A service process M means that the distribution of the service time is identical for each customer/packet, is independent from one customer to another, and is exponentially distributed

The queues you need to simulate in this experiment are M/M/1 and M/M/1/K. We will not be modeling multi-server queues here.

4.2 M/M/1 and M/M/1/K queues

Consider an M/M/1 queue with arrival rate of 10 packets/sec and service rate of 500 packets/sec. In this queue, packets will arrive to the queue following a Poisson process where the queue is expected to receive 10 packets/sec on average. When a packet arrives to the queue, it can be serviced (i.e., transmitted by the server) right away if the queue is empty. If the queue is not empty, an arriving packet has to wait in the queue for its turn to be serviced. Note that the queue size in M/M/1 queue is infinite, and hence, any arriving packet will find a room in the queue and no packet will be dropped. The packet waiting time in the queue (queuing delay) is an important metric to measure the performance of a computer network. In this project, you will measure the queueing delay in different queue models. Once a packet arrives at the server, it will be transmitted with a service rate following an exponential distribution. The service rate of the server depends on the length of each packet, which is assumed to be different for every packet, and the transmission/processing speed of the server. Packets that are longer will take longer to process than packets that are smaller, for the same server speed.

Now let us consider M/M/1/K queue model. The only difference between M/M/1 queue and M/M/1/K is that the buffer size (K) in M/M/1 is infinite while it is limited in M/M/1/K to a size of K packets. As a result, if a packet arrives in an M/M/1/K queue and this queue has already K packets in it, the packet is dropped, i.e., the packet will be lost. Packet loss ratio is another performance metric of a computer network. It is the ratio of packets that are lost to the total number of packets that are received. In this project, you will calculate the packet loss ratio for M/M/1/K queue.

4.3 Basics of Simulation Modeling

A simulation model consists of three kinds of elements: *Input variables, Simulator, and Output variables*. The simulator is a mathematical/logical relationship between the input and the output. A simulation experiment is simply a generation of several instances of input variables, according to their distributions, and of the corresponding output variables from the simulator. We then use the output variables (usually some statistics) to analyze the performance measures of the system. The generation of the input variables and the design of the simulator will be discussed next.

4.4 Generation of input variables with different distributions

In this project, you will need to simulate random variables with exponential probability distributions. However, common libraries for generating random numbers generate only uniformly distributed variables. Because it is instructive to do so, we will write a short function to generate an exponentially distributed variable from a uniformly generated one using a method called the inverse method.

Presume that our desired distribution is given by F(x), and we can only generate a variable, U, which is drawn from a uniform distribution. In fact, we do not want F(x) (we already know this – it's the exponential distribution). We want X itself – that is, a variable drawn from the exponential distribution. If you need to generate a random variable *X* with a distribution function $F(x)$ using the inverse method, you start by setting

$U = F(x)$, where U is our uniform random variable in the range (0, 1)

Then, find the inverse of the distribution function, i.e.,

$$x = F^{-1}(U)$$

Now, if you generate uniform random variable U in the range (0, 1) and substituted in the previous equation, you will get a random variable x that is drawn from a distribution $F(x)$. You will need to use the inverse method to generate the required random variables for M/M/1 and M/M/1/K.

4.4.1 Generating exponential random variables

Assume that we want to generate an exponential random variable x with average $1/\lambda$, where λ is the rate parameter. We will use the inverse method. We start by the exponential cumulative distribution function given as:

$F(x) = 1 - e^{-\lambda x}$ (1)

Set $F(x)$ equal to U which is a uniform random variable

$F(x) = U$ (2)

Then, substitute from (1) in (2) and solve for x

$1 - e^{-\lambda x} = U$

$e^{-\lambda x} = 1 - U$

$-\lambda x = \ln(1 - U)$

$$\boldsymbol{x = -\left(\frac{1}{\lambda}\right) ln(1 - U)}$$

where U is a uniformly generated number and x is the exponential random variable.

4.5 Designing a Discrete Event Simulator (DES)

Discrete Event Simulation (DES) is commonly used to study the performance of evolving systems whose *states* change at discrete points in time in response to external and internal *events*. A DES simulator consists of a set of time-ordered events E_i, i = 1,…, N, along with their occurrences time, such that $t(E_i) < t(E_{i+1})$ for all *i* as shown in Table 1. The time of the last event, which is $t(E_{i+3})$ in this case, is the called the simulation time. Note that there is a difference between the simulation time and the execution time of your code. The execution time is how long your code will run for in order to simulate a system for an amount of time equal to the simulation time.

Table 1. *Discrete Event Simulation (DES)*

Event	Time
E_i	$t(E_i)$
E_{i+1}	$t(E_{i+1})$
E_{i+2}	$t(E_{i+2})$
E_{i+3}	$t(E_{i+3})$

To perform DES, you need to create events and their corresponding occurrence time. But when should the DES terminate? By performing DES, we are performing a statistical experiment. In order to get good results in a statistical experiment, your simulation time should be large enough to truly represent the system under consideration, which is a queue in this case.

An easy way to get stable simulation results is to run the experiment for T seconds, take the result, run the experiment again for 2T seconds and see if the expected values of the output variables are similar to the output from the previous run. For example, the difference should be within 5% of the previous run. If the results agree, you can claim the result is stable. If not, you try again for 3T, 4T, … If you cannot seem to find a proper T, it may mean that your system is unstable. For this project, the value of T should be starting from than 1000 seconds, but you should still use the procedure described above to check the stability (and hence the validity) of your results.

4.5.1 Simulating A Queue

Systems involving queues are generally simulated with a DES simulator. A queue comprises two components: a buffer and one or many servers.

1. In order to simulate a queue using DES, you need to select the simulation time T and create queue events and their corresponding time. In a queue, you have three types of events:
    a. Packet arrival: Based on the type of the arrival process specified in the Kendall notation, generate a random variable that represents the arrival times of packets to a queue. For example, in M/M/1 queue, the packet arrival will follow an exponential distribution. Use the inverse method previously explained to generate packet arrivals for a period of time equal to the simulation time T.

    b. Packet departure: Calculate the departure time of a packet based on the state of the queue, i.e., whether the queue has packets or it is idle. If the queue has packets, then the departure time of a packet pkt_i will be the departure time of the previous packet pkt_{i-1} plus the service time of pkt_i. Note that the service time depends on the length of the packet, L, and the transmission rate, C, of the server. If the queue is idle, then the departure time of packet pkt_i will be its arrival time plus its service time. See Table 2.

Table 2. Example of generating arrivals and departures

Event	Arrival Time	Length (bits)	Service Time	Departure
Arrival	0.01172	3694.4	0.0037	0.0154136
Arrival	0.01942	13438	0.0134	0.032857
Arrival	0.04083	7341.3	0.0073	0.0481729
Arrival	0.06028	24220	0.0242	0.0845039
Arrival	0.0734	9477.1	0.0095	0.0939809
Arrival	0.08096	6789.5	0.0068	0.1007704
Arrival	0.09245	6895.9	0.0069	0.1076663
Arrival	0.11049	4626.9	0.0046	0.1151162

    c. Observer: In order to monitor the queue, you need to generate observer events at which you are going to record the state of the queue. Generate a set of random observation times according to the packet arrival distribution with rate at least 5 times the rate of the packet arrival. At an observer event, you will record the state of the queue which can be done by recording the following: the time-average of the number of packets in the queue, E[N], the proportion of time the server is idle (i.e., the system is empty) P_{IDLE} and in the case of a finite queue, the probability P_{LOSS} that a packet will be dropped (due to the buffer being full when it arrives). The measurement and recording of the state should be performed such that we have statistically representative information on the performance that we are interested in. In order to record the state of the queue, you need to have three counters: N_a = number of packet arrivals, N_d = number of packet departures, and N_o = number of observations. The onus is on you to find out how to use the previous three counters to calculate the state of the queue, i.e., E[N], P_{IDLE}, and P_{LOSS} (in case of finite queue). We typically assume that the observation is instantaneous (that is, there is no service time for observation).

2. After generating all the three events, put all the events in a list and sort them according to time as shown in the Fig. 2.

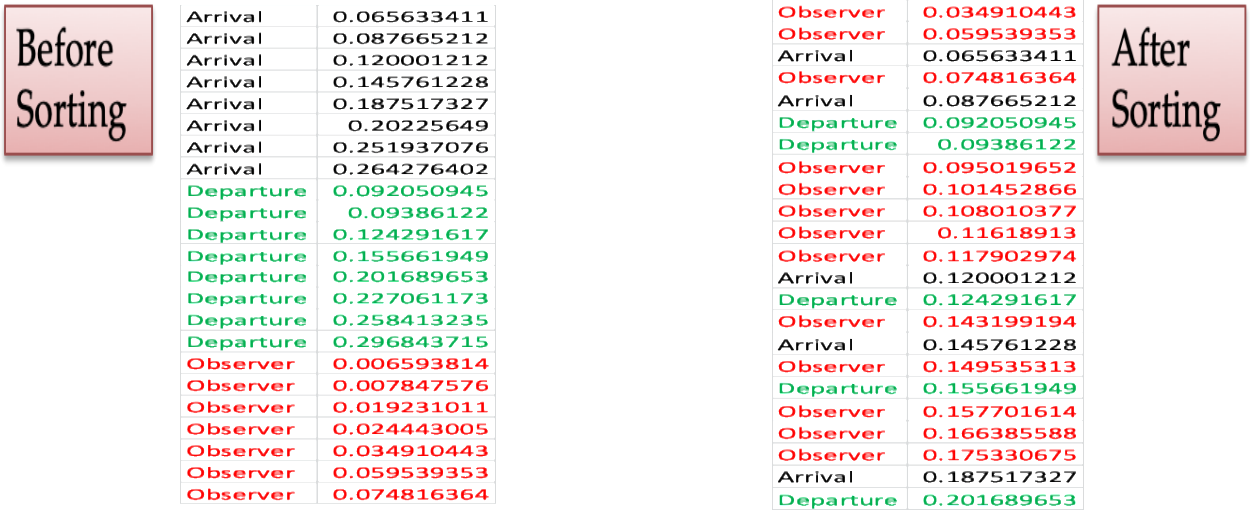

Figure 2. DES example

3. Start processing the events in order from DES. Based on the event type, you should update the three counters N_a, N_d and, N_o. If the event is observer, calculate the performance metrics of interest, e.g., P_{IDLE}, P_{LOSS}. After an event has been processed, it should be deleted from DES.

4.6 Notations

- λ = Rate parameter
- L = Average length of a packet in bits
- α = Average number of observer events per second
- C = The transmission rate of the output link in bits per second
- ρ = Utilization of the queue (= input rate/service rate = L λ/C)
- E[N] = Average number of packets in the buffer/queue
- P_{IDLE} = The proportion of time the server is idle, i.e., no packets in the queue nor a packet is being transmitted
- P_{LOSS} = The packet loss probability (for M/M/1/K queue). It is the ratio of the total number of packets lost due to buffer full condition to the total number of generated packets

5 Report Requirements

Read all the questions in the report template document before you start designing your simulator.

Note that the questions build on each other. It is imperative that you ensure the code for each question is working before moving on to the next, or you will compound your errors.

6 FINAL REPORT

Submit the following in a *.zip file to the dropbox on LEARN. Your zip file may not include subdirectories. It must be named using your (maximum) 8-character UW ID plus the suffix _lab1.zip, such as “mstachow_lab1.zip”. Incorrectly named files will lose marks.

1. Source code with proper documentation/comments. Insufficient documentation will result in losing marks. This source code must be submitted as *.py files. Do not submit source code in your report!
2. Your filled in report template, submitted as a pdf. Submitting in any format other than pdf will result in lost marks.

You may be asked to give a demo of your simulator.

7 REFERENCES

A. Law, W. D. Kelton, *Simulation Modeling and Analysis*, 2nd edition, McGraw-Hill, 1991.