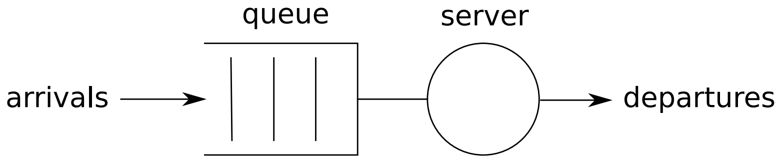
Computer Engineering 4DK4
Lab #1
Performance of Single Server Queueing Systems

Queueing systems occur when "customers" arrive to a system to obtain "service". When there is limited service capacity, the customers may be allowed to "queue up" and wait their turn before they can obtain the service that they require. Real-world queueing systems are almost everywhere. At a grocery store, for example, shoppers (customers) often must wait in line (i.e., queueing up in first-come-first-served order) until the cashier (the server) is free before obtaining service. In computer networks, customers may consist of data packets that arrive to an Internet router and must be buffered (queued up) while waiting for transmission on an outgoing data link (the server). This process may incur delays that affect the quality of the service that the network can provide. Knowledge of queueing system performance is therefore an important key to understanding the behaviour of packet switched networks such as the Internet.

This lab is an introduction to the performance of single-server queueing systems. A program written in C is used to investigate the behaviour of single server queueing systems where customers with fixed service times arrive according to a Poisson process. A single server queue is often drawn as follows.



Customers arrive to the system at the left, wait in the queue until the server is free and eventually enter the server for a time duration known as their service time. Eventually the customer departs the system on the right. The C program is very simple and does not distinguish between customers, but merely keeps track of time, the total number of customers in the system, and various other information. Based on this, it determines whether the next event to occur is a customer arrival or a customer departure. The simulation is changed in various ways to investigate different properties of the queueing system.

1 Preparation

1. It is important that you attend the lectures that introduce the C program to be used. You will be responsible for knowing how the simulator works.

2. An electronic copy of the simulation program must first be obtained from the course web site. The name of the program is `coe4dk4_lab_1_2022.c`. You must also obtain a copy

of the `simlib.c, simlib.h` and `trace.h` library files that you will compile and link together with `coe4dk4_lab_1_2022.c`. Everything needed is zipped together in a single file, `coe4dk4_lab_1_2022.zip`, available on the course web site.

3. You must first know how to compile and run the simulation. There are instructions for this in the lab section of the course web site. If you have trouble or if you are not familiar with C and a C compiler, make sure that you contact us as soon as possible.

2 Experiments

1. Familiarize yourself with running the simulation program. When the simulation finishes a run, it writes output on the screen. There are various parameters such as the customer `ARRIVAL_RATE`, `SERVICE_TIME` and `NUMBER_TO_SERVE` which you can set near the top of the simulation. Once you are familiar with compiling and running the program, continue on and perform the following experiments.

2. Make sure that `SERVICE_TIME` is set to a value of 10. Then vary `ARRIVAL_RATE` over the range

   $$0 < \mathtt{ARRIVAL\_RATE} \times \mathtt{SERVICE\_TIME} < 1 \tag{1}$$

   doing simulation runs to collect results. For each change in the simulation you must recompile and run the program[1]. For each run, record the results printed on the screen when the run finishes. For each value of `ARRIVAL_RATE`, you should do several simulation runs (at least 10) using *different* random number generator seeds. The random number seeds are set by the `RANDOM_SEED` preprocessor macro near the top of the program. *Use your McMaster ID number as the random seed for one of the runs at each value of* `ArrivalRate`. The length of each run (i.e., `NUMBER_TO_SERVE)` should be very large, e.g., millions of customers (the more the better).

   Generate a plot of mean delay vs. `ARRIVAL_RATE`. *The value for mean delay that you plot should be the average of the values you obtain for different random generator seeds at a particular value of* `ARRIVAL_RATE`.

   Explain the reason for the mean delay value at low `ARRIVAL_RATE` values, i.e., What is the mean delay axis intercept in terms of the other system parameters? Explain the behaviour of the mean delay as the `ARRIVAL_RATE` increases and approaches the right-hand inequality in Expression 1. What is the value of the `ARRIVAL_RATE` axis asymptote when this happens? Explain the shape of the mean delay curve obtained.

3. What happens when the product of `ARRIVAL_RATE` and `SERVICE_TIME`, in Expression 1 is greater than 1? To see, set `ARRIVAL_RATE` slightly greater than 1/`SERVICE_TIME`, then do a run of 10,000 customers. What happens when you keep increasing the run length? Explain why is this happening. Explain why the condition in Expression 1 is necessary.

[1] Rather than manually performing each simulation run, you may prefer to wrap the provided code in loops while changing the various parameters. The multiple runs can then be automated.

4. Repeat Part 2 but this time set the `SERVICE_TIME` to 30 and vary `ARRIVAL_RATE` over a suitable range. Compare the mean delay vs. `ARRIVAL_RATE` curve to what you obtained before. Plot this curve on a graph with the curve from Part 2. Explain the differences in the curves.

5. Using Queueing Theory it can be shown that the mean delay for a work conserving single server queueing system with Poisson process arrivals and a general service time distribution (i.e., an M/G/1 queue) is given by[2]

$$\overline{d}_{M/G/1} = \overline{X} + \frac{\lambda \overline{X^2}}{2(1-\rho)} \tag{2}$$

where $\overline{X}$ is the mean service time (`SERVICE_TIME`), ρ is the product of the mean arrival rate and the mean service time, i.e., $\rho = \lambda \overline{X}$, λ = `ARRIVAL_RATE`, and $\overline{X^2} = \sigma_X^2 + \bar{X}^2$ is the second moment of the service time[3]. Note that the total delay of a customer is the sum of its service time and its queueing delay. Therefore, the two terms in Equation 2 are the mean service time and the mean queueing delay incurred by the customers, respectively.

When the service times are fixed (i.e., an M/D/1 queue, as in the supplied simulation), $\sigma_X = 0$, and therefore, the mean delay can be written as [4]

$$\overline{d}_{M/D/1} = \frac{\bar{X}(2-\rho)}{2(1-\rho)}$$

Compare the results that you plotted in Parts 2 and 4 to this analytic result.

6. Modify the program so that instead of M/D/1 you simulate an M/M/1 queuing system, i.e., instead of Poisson process arrivals and fixed service times, we now have Poisson process arrivals and exponentially distributed service times with the same mean of `SERVICE_TIME` i.e., When the program needs a service time,

```
exponential_generator((double)SERVICE_TIME);
```

will return an exponentially distributed service time with the proper mean[5].

Repeat Part 2 above using the same value of `SERVICE_TIME`. Plot the mean delay vs. `ARRIVAL_RATE` for the M/D/1 and M/M/1 cases on the same graph. Explain the differences and similarities.

Repeat Part 5, using Equation 2 for an M/M/1 system. Give a simple expression for $\overline{d}_{M/M/1}$.

7. Modify the code so that it simulates a single server queueing system with a maximum queue size, e.g., define `MAX_QUEUE_SIZE` at the top of the simulation. When a customer arrives and finds that the number in the queue is at its maximum, the arriving customer is

[2] Read a description of the M/G/1 queue at `https://en.wikipedia.org/wiki/M/G/1_queue`.

[3] Equation 2 is known as the Pollaczek-Khinchine formula. See `https://en.wikipedia.org/wiki/Pollaczek-Khinchine_formula`.

[4] Read a description of the M/D/1 queue at `https://en.wikipedia.org/wiki/M/D/1_queue`.

[5] Read a description of the M/M/1 queue at `http://en.wikipedia.org/wiki/M/M/1_queue`.

rejected from the system without being served. At the end of each simulation run, output the customer rejection probability, i.e., the ratio of the number of rejected customers to the total number of customer arrivals. Note that only served customers are included in the mean delay calculation.

In this type of system, there is no limit on the arrival rate that can be used, since customers that find a full queue will simply be rejected, i.e., the queue can never grow larger than `MAX_QUEUE_SIZE`. For this reason, the upper limit in Expression 1 need not be satisfied.

For various values of `MAX_QUEUE_SIZE`, plot the mean customer delay versus arrival rate. Make sure that you use values of arrival rate that are sufficiently large to see the asymptotic value of mean delay. Also do plots of the rejection fraction versus the same arrival rate values. Discuss the differences and similarities between the infinite and finite queue cases. Find a relationship between the mean delay asymptote and the other system parameters as the arrival rate becomes large.

8. Start with the original version of the code with `SERVICE_TIME` set to 10. Modify the code so that instead of Poisson process arrivals, the time between arrivals is fixed with a value equal to 1/`ARRIVAL_RATE`, i.e., the time between successive arrivals is always the same. Repeat Parts 2 and 3 above. Explain why you get such a different mean delay vs. `ARRIVAL_RATE` curve. Explain the shape of the curve by drawing an example of what is happening in time.

3 Writeup

Submit a writeup for the lab.

1. Each group (of 2 maximum) may submit a single writeup.
2. Include in your writeup a description of everything that you did including all data and the random number generator seeds that were used to obtain the graphs.
3. Include with your writeup the plots and a listing of any of the code that you changed.
4. Unless otherwise indicated, your performance data and curves should cover the system operations from low traffic load to high traffic load.
5. No piecewise linear plots or overfitted curves are allowed.