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

This lab is an introduction to discrete-event simulation applied to computer networks. A program written in C is used to investigate the behaviour of a single server queueing system where customers with fixed service times arrive according to a Poisson process. An arriving customer starts service immediately if there is no other customer in the system, otherwise it is placed in a queue to await service. The program is very simple and does not distinguish between customers, but merely keeps track of the total number of customers in the system and various other information. Based on this, it determines whether the next event is a customer arrival or 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 which 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_2016.c`. You must also obtain a copy of the `simlib.c, simlib.h` and `trace.h` library files which you will compile and link together with `coe4dk4_lab_1_2016.c`. Everything needed is zipped together in a single file, `coe4dk4_lab_1_2016.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

$$0 < \texttt{ARRIVAL\_RATE} \times \texttt{SERVICE\_TIME} < 1 \qquad (1)$$

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., 5,000,000 customers or more (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., the mean delay axis intercept). Explain the behaviour of the mean delay as the `ARRIVAL_RATE` increases. What is the `ARRIVAL_RATE` axis asymptote when the mean delay becomes very large? Explain the shape of the mean delay curve obtained.

3. What happens when the product of `ARRIVAL_RATE` and `SERVICE_TIME` (In Equation 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 Equation 1 necessary.

4. Repeat Part 2 but this time set the `SERVICE_TIME` to 20 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 this type of queueing system (M/D/1 queue) is given by[2]

$$\frac{\bar{X}(2-\rho)}{2(1-\rho)}$$

   where $\bar{X}$ is the `SERVICE_TIME` and ρ is the product of the mean arrival rate and the service time, i.e., $\rho = \lambda\bar{X}$, i.e., λ = `ARRIVAL_RATE`. 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[3].

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

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

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

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, except that now the mean delay (for an M/M/1 system) is given by

$$\frac{\bar{X}}{1-\rho}$$

where $\bar{X}$ is the *mean service time* (i.e., `SERVICE_TIME`) and $\rho = \lambda \bar{X}$ as before.

3 Writeup

Submit a writeup for the lab. Each group (of 2 maximum) may submit a single writeup. 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. Include with your writeup the plots and a listing of any of the code that you changed.