Simulation project
Task 2 – Write the basic simulation code

Remember that any exchange of code with another student is forbidden as it constitutes cheating. Keep the code to yourselves! We will run Moss at the end of this assignment to verify that there has not been any code sharing. Offenders will receive an automatic zero for the entire project.

Objectives

In this task, you will write the simulation code to reproduce the hand simulation. Also, you will introduce randomness in your simulation by assuming that the inter-arrival and retransmission times are exponentially distributed. The service time will remain constant, since it takes the same time to serve a request.

You can use any high-level language of your choice. You can develop your program on your personal computer, but it has to run and compile on eos. Programming elegance is not required! What is important is that your program runs correctly. For this, you are *strongly* advised to follow the instructions in this handout!

Use the same program you developed in task 1 but assume that the inter-arrival times and retransmission times are exponentially distributed. The mean values of these two exponential distributions are the same as in task 1, i.e., 6 and 5 respectively. The service time remains constant equal to 10 as in task 1, since the execution time of a request is more or less constant. Each time you want to generate an inter-arrival time or a retransmission time, draw a pseudo-random number r and then use it to obtain an exponential variate as described in section 3.2.2. Print a line of output each time you handle an event, i.e., you advance the master clock, with the same information as in task 1. Run your simulation until MC goes over 200.

Go over the results by hand to make sure that the simulation advances properly from event to event. Verify by hand that in both cases the simulation advances correctly from one event to the next one. This is your chance to make sure that your implementation is correct !!

Word of caution: The input values will cause the event list with the orbiting customers to continuously increase. This is done on purpose so that to create different dynamics in the simulation for debugging purposes. We will change these values in task 3.

What to submit

1. Submit your code. To simplify things, submit two different programs, one for the first deliverable and a separate one for the second deliverable.

   Make sure that the code runs on eos. Also, set up your code so that it would receive input from the terminal. Your output should be saved in a file named output.txt. The input values are

- the mean inter-arrival time,
- mean orbiting time,
- service time,
- buffer size, and
- value of the master clock at which time the simulation will be terminated.
-

2. Submit in a separate file your output from sub-tasks 1 and 2. The output will be obtained until MC=200.

Grading

- 30 points for each version of the code if it runs and compiles on eos correctly
- 20 points for each version of the code if it gives the correct output on test data provided by the TA.