INSTITUTO SUPERIOR TÉCNICO

Traffic Engineering

1st lab project

Simulation of Poisson distributions and M/M/1 queues

Fernando Mira da Silva

APRIL 2024+5

1 Goal

Simulate, analyze and discuss the properties of Poisson arrival processes and its application to queue processing.

The full first lab project will cover three lab sessions. Please note that before the end of the second session the work must be assessed on site by the teaching staff, otherwise the final report will not be considered for grading. The written report including lab results (see section 4 below) must be delivered until May 12th, Monday, on the Fenix System.

2 Poisson process simulation

2.1 Introduction

The probability of observing a given number of events on a unitary period in a Poisson arrival process is given by

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

The Poisson process is closely related to the exponential distribution. As it is well known, the time *t* between two successive events is a continuous random variable with distribution given by the probability density function

$$f(t) = \lambda e^{-\lambda t}$$

A pseudo random variable δt with exponential distribution can be generated following the according to the following procedure:

1. Generate a random number *u* with a uniform distribution on [0,1).
2. Apply the transformation

$$\delta t = -\frac{\ln(1-u)}{\lambda}$$

where ln is the natural logarithm.

2.2 Arrival process generation

The first goal of this lab is to write a computer program in python or matlab in order to simulate a Poisson process:

1. Generate a sequence of N=500 events with time interval δt between successive events. Generate δt according to a exponential distribution of parameter λ. Make λ=5 for initial tests

2. Use the above sequence to make a histogram (bar chart) of the number of events occurring in a unitary time interval. **The use of any built-in *hist()* function is forbidden, develop your own version! This is mandatory!**

   Why is the *hist()* function forbidden*?* We noticed that many students use the *hist()* function without fully understand its behavior, therefore achieving wrong results. Since the implementation of the *hist()* function is quite simple and just requires a few lines of code, the mandatory implementation will help you to understand what is happening…

3. Use any plotting software (*matlab*, *gnuplot, excel*, built-in matlab) to display the above histogram against the theoretical Poisson distribution. Use a bar plot for the histogram and a X/Y line plot for the theoretical distribution. Both plots must be displayed in the same window and scale. Check if the experimental data fits the theoretical distribution.

4. Repeat for different values of λ. Use as reference at least the following values for λ: {0.5, 1.0, 5.0,10.0, 50.0}. Try your program with values of *N* in the range 10 to 50,000. Check the results. Discuss.

2.3 Superposition of Poisson processes

1. Modify the program above to generate 4 independent sequences with different λ values: (3,7,13,15);
2. Generate a single sequence that combines the events of the 4 event generation processes developed in point 1;
3. Use the same sequence above to generate a histogram of the combined process.
4. Plot a graphic representation of the histogram jointly with the theoretical distribution that results from the combination of the 4 processes ago. As in point 3 of 2.2, use a single window and scale to display both plots.

Repeat for different values of the number of generated samples. Discuss the result.

3 M/M/1 queue simulation

Simulate a M/M/1 queue using a discrete event simulation written in either python or matlab.

Assume that the arrival process at the queue is Poisson with parameter λ. Assume that events on the queue are processed on a FIFO basis, and that each event at the top of the queue takes *t* seconds to be processed. *t* is a random variable with an exponential distribution with parameter μ.

The simulator in this particular case is quite simple, since only two types of events need to be supported: a new packet arriving at the system and the end of processing of one packet by the server/processor. The event list is a simple list of tuples *(t,event_type),* sorted on *t*, where *t* is time and *event_type* one of the two event types described above. Furthermore, you need to simulate the M/M/1 packet queue, a FIFO of packets waiting to be processed by the M/M/1 server. Do not confuse the event list (which contains the next events to be simulated) with the packet queue (the set of packets waiting to be processed by the M/M/1 server).

For each event in the event list, the simulator does the following:

1. Read the next tuple (*t,event_type)* from the event list;
2. If the event type (*event_type*) is a packet arriving to the system, add it to queue and generate a new packet arrival event at time $t_1=t_0+\delta t$, where t_0 is the current time and δt is a random variable with exponential distribution and parameter λ, and add it to the pending event list at time t_1. Go to step 4.
3. If the event type (*event_type*) is the end of processing of a packet by the server, mark the server as free.
4. If the server is free, check if the queue is empty. If empty, go to 1.
5. Take the oldest packet from the queue and put in the server. Mark the server as busy and generate a new end of processing event at time $t_1=t_0+\delta t$, where t_0 is the current time and δt is a random variable with exponential distribution and parameter μ, and add it to the pending event list at time t_1.
6. Go to step 1.

To bootstrap the discrete event simulator, just create a packet arrival event at *t=0* and add it to pending event list.

After writing the simulator code

1. Check the average queue size for different values of λ and μ and compare it with the theoretical value. Simulate the queue in the cases $\mu < \lambda$, $\mu = \lambda$ and $\mu > \lambda$. Discuss the results. Test combinations of λ and μ such that the average queue size is 1, 10, 100 and 1000.
2. For the case where is queue size is 100, plot the evolution of the queue against time for 1000, 10 000 and 100,000 generated events.
3. Does the queue size converge to the theoretical value? Find a theoretical explanation for the observed results.

4. Adapt your code such that you are able to compute (1) the average time a packet is in the system (2) the average time a packet is in the queue (3) the percentage of time the server is busy.

4 Assignment report

The written report of this assignment must include:

1. Discussion of used algorithms and developed software;
2. Graphics, tables and other results of the simulations which may be relevant for this experiment;
3. Discussion of achieved results;
4. Software source (supplied as an attached archive in any standard format: zip, tar, tgz, rar…).