S-38.148 Simulation of Data Networks

CNCL Simulation Exercise

Contents

Chapter 1

Introduction

The purpose of this exercise is to make you familiar with CNCL simulation library. You will learn how to create new simulator blocks with C++ by using the objects and functions provided by CNCL and how to collect statistics from the simulation.

In this exercise we simulate a spatial Poisson process with moving points. The purpose is to evaluate and verify certain stochastic properties of the process.

Chapter 2

Background for the exercise

2.1 The spatial Poisson process with randomly moving points

A Poisson point process on a plane is a process, where the number of points in a given area A obeys the Poisson distribution,

$$N_A \sim \text{Poisson}(\sigma A),$$

where the intensity σ denotes the expected number of points per area element. Additionally, for two disjoint subsets (with areas A_1 and A_2) the number of points in each subset are independent and obey the Poisson distribution with parameters σA_1 and σA_2. Thus, the spatial Poisson process is an extension of the one dimensional Poisson process.

Consider the entire plane $\mathbb{R}^2$ and let us assume that at time $t = 0$ points have been placed on the plane according to a Poisson process with intensity σ. Assume further that each point has been assigned a velocity v, which is constant, and a direction of movement taken from a uniform distribution $\text{U}(0, 2\pi)$. The velocity of all points remains the same for all t.

The following three properties hold for the process described above.

Property 1 The number of points crossing a differential line element ds obeys a Poisson distribution with parameter $\frac{\sigma v}{\pi} \cdot ds$. Thus, for any curve of length L the number of points crossing the curve obeys a Poisson distribution with parameter $\frac{\sigma v}{\pi} \cdot L$.

Property 2 The probability density function (pdf) of the angle θ at which the points are crossing a differential line element ds is given by

$$f(\theta) = \frac{1}{2} \cos\theta, \quad \theta \in [-\pi/2, \pi/2]\,.$$

Here the angle θ is relative to the normal vector pointing inside the area whose perimeter is defined by the curve L, see Figure 2.1.

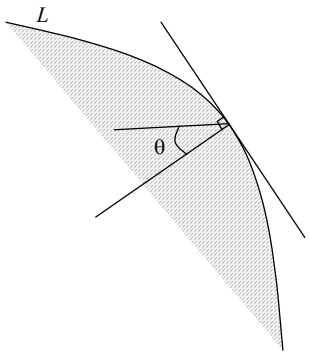

Figure 2.1: Illustration of θ.

Property 3 Conditioned on that a point crosses a given curve L, the location of the point on L is uniformly distributed according to $\mathrm{U}(0, L)$. This is a direct consequence of the fact that the intensity of points crossing a differential line element ds is a constant (Property 1).

Property 4 Assume that the velocity v is not a constant but also a random variable with pdf $f(v)$ and mean $\bar{v}$. Then the pdf of the velocities at which points cross a differential line segment ds is given by

$$f^*(v) = \frac{1}{\bar{v}} v f(v).$$

Furthermore, the number of points crossing a curve L has a Poisson distribution with intensity λ equal to

$$\lambda = \frac{L\sigma\bar{v}}{\pi}.$$

Thus, if $\bar{v} = v$, the speed distribution does not impact the distribution of the number of points inside the area.

Property 5: The angle distribution of the velocity vectors is isotropic, i.e., the angle distribution obeys $\mathrm{U}(0, 2\pi)$.

2.2 Simulation of the process

Based on the previous it can be concluded that the movement of the points on the whole plane $\mathbb{R}^2$ within a convex subset A can be analyzed, e.g., in simulations in the following manner.

- From outside the area A, new points arrive into the area according to a Poisson process with intensity
$$\lambda = \frac{Lv\sigma}{\pi},$$
where L denotes the length of the perimeter of the area A. Thus, we simulate this process by generating arrivals with inter-arrival times obeying Exp(λ).

- Each generated arrival is randomly placed on the perimeter L (location is uniformly distributed).

- Each point is assigned a velocity $v \sim f^*(v)$ (either constant or from a given distribution), and a direction θ that has the pdf $f(\theta) = \frac{1}{2}\cos\theta$, with $\theta \in [-\pi/2, \pi/2]$ (refer to the earlier figure).

- Each point moves in a straight line across the area until it exits the area and never returns (area A is assumed convex).

2.3 Modeling the process

The system can be seen as an infinite-server queue. The nodes represent jobs that enter a queue according to the Poisson process introduced in the previous section. The arriving customers stay in the system a random duration (depending on the velocity, entry point and arrival angle) and exit the area. Thus, a simple process model similar to the GI/GI/1 queue model is sufficient, see Figure 2.2.

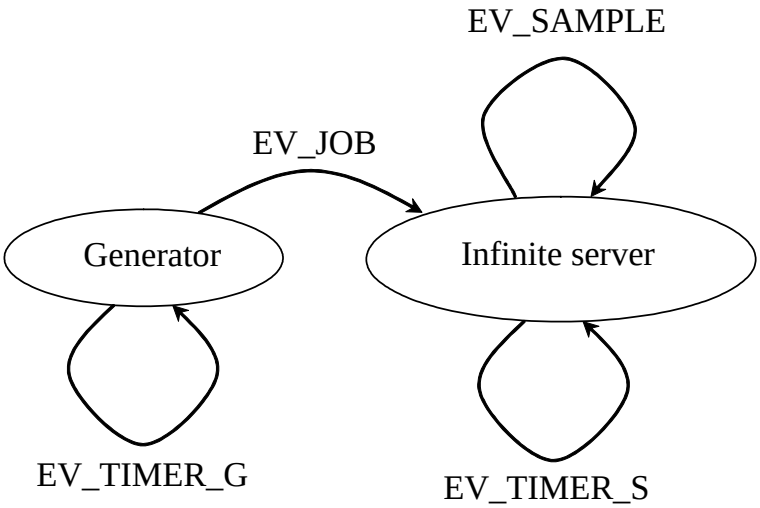

Figure 2.2: Illustration of the simulation model.

However, the queue/server object is not anymore a FIFO queue but a system with (in principle) an infinite number of servers. Thus, the server part only requires

some data structure to which CNCL objects representing the arriving nodes can be stored. For example, in CNCL a simple linked list for storing any CNCL objects is the class **CNSLList**. The class also provides iterator functions for going through the objects in the list, as well as functions to insert and delete objects.

Note that for storing the necessary state information of each node arriving in the area, the standard **CNJob** objects are not sufficient. Thus, you need to derive your own "job" class for storing the state information.

To get information about the various classes in CNCL, consult the CNCL documentation available on the course web page:

http://www.netlab.tkk.fi/opetus/s383148/doc/cncl/

Chapter 3

Exercise

3.1 Practical Instructions

CNCL is installed in all workstations in computer class B215. Remote login to the workstations is also possible, see the list of workstations from

http://www.ee.hut.fi/unix/hardware.shtml.

CNCL also works in the machine

moukari.ee.hut.fi.

In order to be able to do the simulation exercise, you have to go through the following steps:

1. Copy the necessary files (cnclexer.c and Makefile) from the course webpage to your working directory.
2. If your default shell is tcsh, write a shell script including the following lines:

   setenv LD_LIBRARY_PATH /usr/lib:/usr/local/lib:$LD_LIBRARY_PATH
   setenv PATH /opt/csw/gcc2/bin:$PATH

   If your default shell is bash, write a shell script with the following:

   LD_LIBRARY_PATH=$LD_LIBRARY_PATH: /usr/lib:/usr/local/lib
   export LD_LIBRARY_PATH
   PATH=/opt/csw/gcc2/bin:$PATH

export PATH

Execute the script with the command "source usefile", where usefile is the name of your shell script. By default your shell should be tcsh and these commands should work (you can check your shell type with the command "echo $SHELL")

3. Edit the source file cnclexer.c: Implement the required methods and the main program.
4. When you want to compile your program, create first a .depend file with the command "touch .depend".
5. Compile the program by "make".
6. Execute the program with the command "./cnclexer".

If you need to look at the *.h or *.c files of CNCL classes, the CNCL library is installed at:

/p/gen/courses/S38/S38.148/cncl/src/cncl-2.1/

The *.h files are under include-directory and *.c files under lib-directory.

3.2 Simulation task

The purpose of the exercise is to examine certain properties of the spatial Poisson process with moving points including the influence of velocity distributions. In summary your task is to:

- Define a suitable model for your simulations, i.e., define the objects and their roles that will act as the processes sending and receving events (event handlers).
- Implement the objects in CNCL, i.e., the **event_handler()** methods in the classes that send event between each other.
- Implement routines for statistics collection for the various metrics to be defined later. The statistics collection requires that you are able to sample the state of the system at fixed time intervals. Hint: It may be helpful to use **CNMoments** objects for statistics collection.
- Implement the main method for controlling the simulation.

We simulate the process in a unit square, i.e., in an area with $(x, y) \in [0, 1] \times [0, 1]$. On the average there are 100 nodes in the area, i.e., the intensity per area element $\sigma = 100$. Also, two velocity distributions are used:

- Constant: $v = 1$
- Random: $v \sim \mathrm{U}(0.25, 1.75)$, i.e., $\bar{v} = 1$. Thus,

$$f^*(v) = vf(v) = \frac{2}{3}v, \quad v \in [0.25, 175].$$

The detailed simulation tasks are as follows:

Task 1 Divide the area $[0, 1] \times [0, 1]$ in 4 smaller squares (i.e., each with dimensions 0.5×0.5). Simulate the system and show that the number of nodes in each sub-area obeys a Poisson distribution with parameter σA_i, where A_i is area of the sub-area, $A_i = 0.25$. In this task it is sufficient to just simulate using a constant velocity.
Hint: To show that number of points has a Poisson distribution sample the system at fixed intervals $\Delta = 0.5$ and compute the number of nodes. From the statistics compute the mean and the variance (for Poisson distribution they should be the same). Perform a few independent repeated simulations to obtain confidence intervals for your estimates.

Task 2 Now consider the whole area A, i.e., the unit square. Sample the number of nodes in the area at fixed intervals $\Delta = 0.1$ (or at even smaller intervals). Study the correlation between the samples for the two velocity distributions. In the simulation, remember that the speed of the node crossing into A is drawn from the density $f^*(v)$. What is the time when samples become independent in both cases? How does the velocity distribution affect this? Can you explain the observed behavior?

Task 3 In Task 2 you have obtained an estimate for the sampling interval to obtain independent samples. The final task concerns the the stationary velocity distribution of the moving points inside A when the velocities are random. Draw the velocities from $f^*(v)$, estimate the stationary velocity distribution and verify that the distribution of the velocities obeys $\mathrm{U}(0.25, 1.75)$. To do this sample the system at the estimated interval between independence, and collect statistics about the velocity of the nodes. Based on this compute a histogram (with at least 5 bins or so). Also, indicate confidence intervals for the histogram values.

Note! All tasks above relate to stationary simulations. Therefore, remember to implement a warmup period in the beginning before you start the data collection.

3.3 Handout requirements

The following items should be included in your final report:

- All source code with comments
- Graphs/tables of the simulation results
- Brief analysis of the results

3.4 Getting help

Two sessions will be organized, where students will be able to get personal help from the exercise assistant. Help is also available to English speaking students. The time and place of the sessions are:

- Session 1: Fri, Oct 20 (week 42), at 10-12 o'clock in B215
- Session 2: Tue, Oct 31 (week 44), at 14-16 o'clock in B215

You can additionally send e-mail to the assistant responsible for the project work Tuomas Tirronen (email: tuomas@netlab.hut.fi).

3.5 Returning

The deadline of the exercise is

- Mon, November 6, at 12 o'clock.

Return your report and codes via e-mail to tuomas@netlab.hut.fi