Simulating and Modelling Queues

Part 1

Lecture 7

Communication Theory III

Eng. (Mrs.) PN Karunanayake

Why we need simulation?

A simulation experiment **aims at reproducing the dynamic behaviour** of the system (customers, servers, etc) through a computer program, most often run on specific software tools.
With simulations **hypotheses can be formulated and verified**, **alternative technological solutions may be experimented with,** eventually to give rise to an **improved solution and its model**.

On software models, **observations are performed**, which give the figures of interest, such as **mean** or **variance of delays**, **loss probabilities**,.

Simulation Approaches

- Discrete-event simulation.
- Monte Carlo Simulation

Roulette simulation

- Advantage is the **speed of execution, Simple and Efficient.**
- Example:

•The "state" stands for the number of calls in progress in the concentrator. Elementary events are:

- The arrival of a new request, which changes the state from n to n + 1 if n < R. In state n, N - n sources only are idle and able to issue a request: the birth rate is (N - n)λ .
- The end of a call, with a rate nμ, which changes the state from n to n - 1.

• Given the state n the transition probabilities to adjacent states are respectively:

$$P(n \to n-1) = \frac{n\mu}{n\mu + (N-n)\lambda}, \quad P(n \to n+1) = \frac{(N-n)\lambda}{n\mu + (N-n)\lambda}.$$

Roulette simulation

- The simulation experiment runs as follows:

  (a) Draw a random number r, uniformly distributed in the interval (0,1).

  (b) Test if $r < n\mu/[n\mu + (N - n)\lambda]$. If yes (a departure occurs), move the system to the state n - 1. If no (an arrival occurs), move to n + 1.

  (c) Repeat operations (a) and (b) until the number of events reaches the limit assigned to the experiment.

- The main drawback with roulette simulation is that it does not incorporate any notion of "time".
- The memoryless property of all the processes is centralized to the method, so that non exponential service or interarrival durations are impossible to take into account.
- Unable to measure any delay, such as waiting times and busy periods

Monte Carlo Simulation

- Simulations that use sequences of random numbers and which are independent of time, such as the use of random numbers to evaluate definite integrals, are called Monte Carlo simulations.

Discrete-event simulation

- Although the system being simulated exists in continuous time, **changes in the simulation model take place only upon the occurrence of well-defined events which are discrete**.

- This method requires estimating the date of a future event, which is obtained by generating random intervals according to a given probability distribution.

- Ex. When a service of duration exponentially distributed begins, its duration is drawn using the properties of the exponential distribution, determining the date of occurrence of the event "end of service".

Discrete-event simulation

- As events occur on a discrete-time basis (no simultaneous events) they are
- processed chronologically, in the following way:
- - The event notices describing them (date of occurrence, actions to be executed) are stored in a timetable (event list, sequencing set) in order of increasing dates.
- - The simulation logic scans the event in head of list, jumps immediately to the date of the first event and executes the corresponding actions (new request, end of session, failure of an element, etc.).
- The efficiency of such a process depends first on the **complexity of the actions to be executed**, but also on the **number of simultaneous events** to be handled in the event list.

State Descriptor

- **The representation of the system** is called the state descriptor vector or simply, the system state.
- It is a function of time. (In very simple models, it may be just a single integer.)
- For example, if a system such as an auto-repair garage has a single repair bay, all that may be needed to represent its state is the **number of cars waiting or undergoing** r**epair.**
- However, most commonly, the state descriptor vector consists of a number of items called components and each component has its own specific set of attributes. In fact, each component can be viewed as a discrete random variable which assumes the values of its attributes.

Events and Their Management

- Definition of all possible events and specify their effects on the system state.
- The occurrence of an event can cause the components of the **state descriptor to change values**. A particular event may alter the value of only one component or of many components. The occurrence of an event may also leave the system state unchanged

Simulation Time

- The time over which the model is exercised (sometimes called the internal clock time).
- It can be either a fixed period of time or the time until a pre-specified number of one or more events has occurred.
- Examples?
- **A number of performance statistics must to be collected** during a simulation run. These might include the number of times that a particular event occurred, the average duration between occurrences of the same event, or indeed any other statistic that is deemed necessary.
- In order to collect the performance statistics, the simulation program must contain various variables, usually counters.

Example

- In a simple M/M/1 queue,
- 1. What are the events?
- 2. What is the simplest way to represent the state?

Program Structure

- The event modules should include actions that,

  - update the internal clock;

  - change the components of the state descriptor vector in accordance with the effect of the event on the system;

  - increment a counter of the number of times this event has occurred;

  - generate the time of the next occurrence of this event;

  - update other variables that might be required for the computation of performance measures.

Example - Simulating the M/M/ 1 Queue and Some Extensions.

Variables

-A single integer **n** is sufficient to completely characterize **a state of the system** and thus is taken as the **state descriptor vecto**r.

- There are only two events: **the arrival of a new customer to the system** and the **departure of a customer who has just completed service.**

- Integer variables n_a and n_d count the **number of arrivals and departures**, respectively.

Example - Simulating the M/M/ 1 Queue and Some Extensions.

- Since we require the **mean time spent in the system**, we keep the arrival time of each customer.
- Element i of array T will hold the arrival time of customer i, for i ≤ N . **Subtracting T [i] from the time customer i departs gives customer i's total system time (the variable response).**
- The **average queueing time** (variable wait) is found by subtracting the sum of all service times from the sum of all response times and dividing by N .
- At simulation initialization, we shall set t = 0 and t d = ∞ where ∞ is taken to be a number so large that there is little possibility of t ever reaching this value during the simulation. The initialization will also generate the time of the first arrival and initialize t a to this value.

Example - Simulating the M/M/ 1 Queue and Some Extensions.

We shall run the simulation **until a total of N customers have been served** and then compute the **mean interarrival time, the mean service time and the mean time spent waiting in the queue.**

- The variable t will be used to denote the **internal clock time.**
- The variables t_a and t_d denote the scheduled times of the **next arrival and the next departure.**
- We use t_λ and t_μ to denote generated **interarrival times and service times,** respectively.
- During the simulation run, tot_λ and tot_μ are **running totals of arrival times and service times**.

Actions that handle events:

- An arrival:

  - Advance the internal clock to the time of arrival.

  - An additional customer has arrived so modify system state n appropriately (n = n+1).

  - Increment n_a , the number of arrivals so far and, if $n_a \leq N$, set T [n a] = t.

  - Generate t_λ , the time until the next arrival; compute the next arrival instant t a = $t+t_\lambda$ and add t_λ into tot_λ .

  - If n = 1, signifying that the server had been idle prior to this arrival, generate t_μ , the service time for this customer, compute its departure time $t_d = t + t_\mu$ and add t_μ into tot_μ .

Actions that handle events:

- A departure:

  - Advance the internal clock to time of departure.

  - A customer has left so modify system state n appropriately (n= n−1).

  - Increment n_d , the number of departures so far.

  - Compute the time this departing customer spent in the system and add into response.

  - If n = 0, set $t_d = \infty$. Otherwise generate t_μ , the service time of the next customer;

  - Compute the next departure instant $t_d = t + t_\mu$ and add t_μ into tot_μ .

Specifying the main module

- Perform all initialization functions as described above.

If the number of departures is less than N :

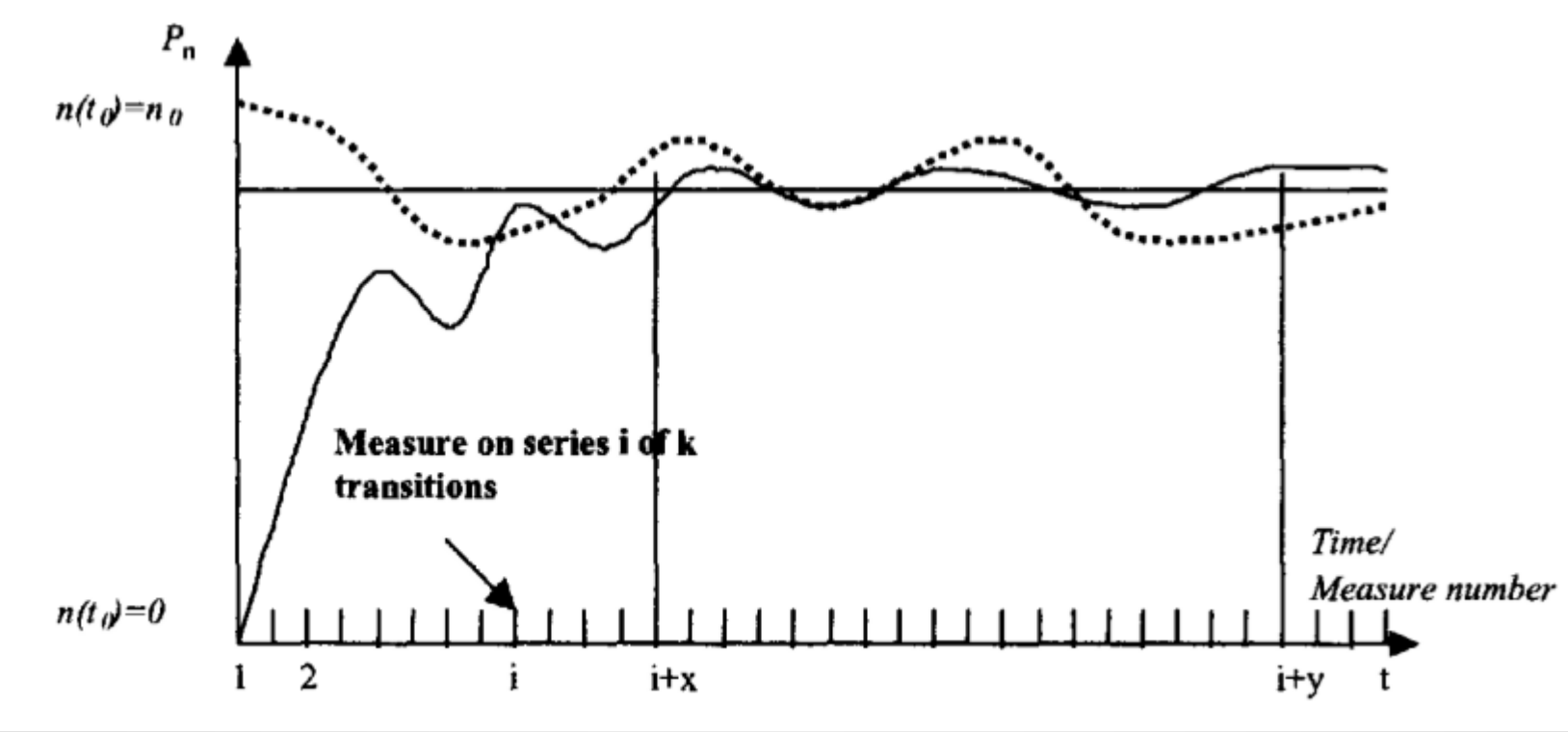
- initiate an arrival event if t a < t d : otherwise initiate a departure event (t a ≥ t d).

- Generate terminal statistics and print report

Measurements & Accuracy of Simulation

- The ideal **measure** should supply a random sampling of the states the system goes through during the simulation.

- Example for accuracy, assume one wants to estimate some state probability (for instance the probability of having n busy servers, or the probability of more than n customers in a buffer).



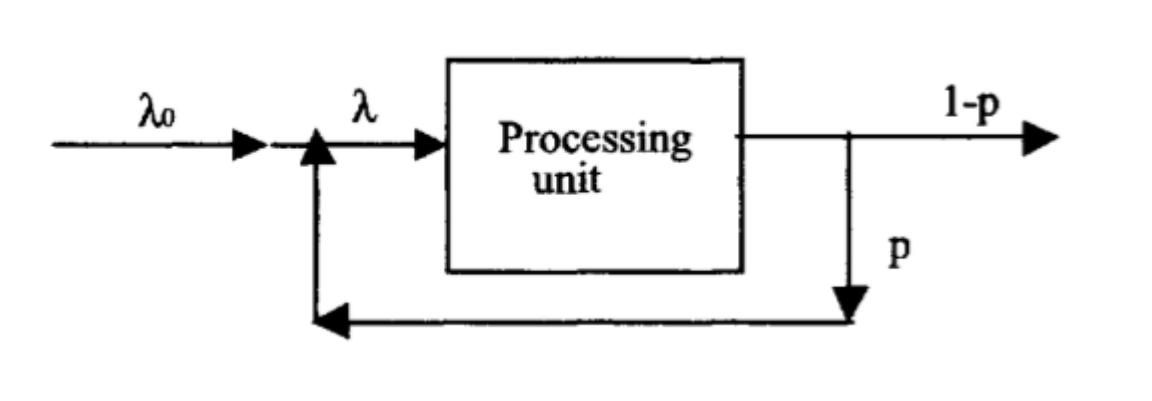
Why we need models?

To evaluate the performances of systems and networks, based on the theoretical results and tools.

The objective is **an accurate reflection** as possible of the highly varied problems associated with modelling in an industrial environment.

Model for System Control

- Corresponds to a transmission system with a failure transmission probability p per message, which results in its retransmission.



$$\lambda = \lambda_0 + p\lambda \text{, that is } \lambda = \frac{\lambda_0}{1-p}$$

Model for M/M/1 Queue considering packet loss

- M/M/1 queue with poissonian arrivals with rate λ, and with a service time 1/μ,
- Mean time spent in the system

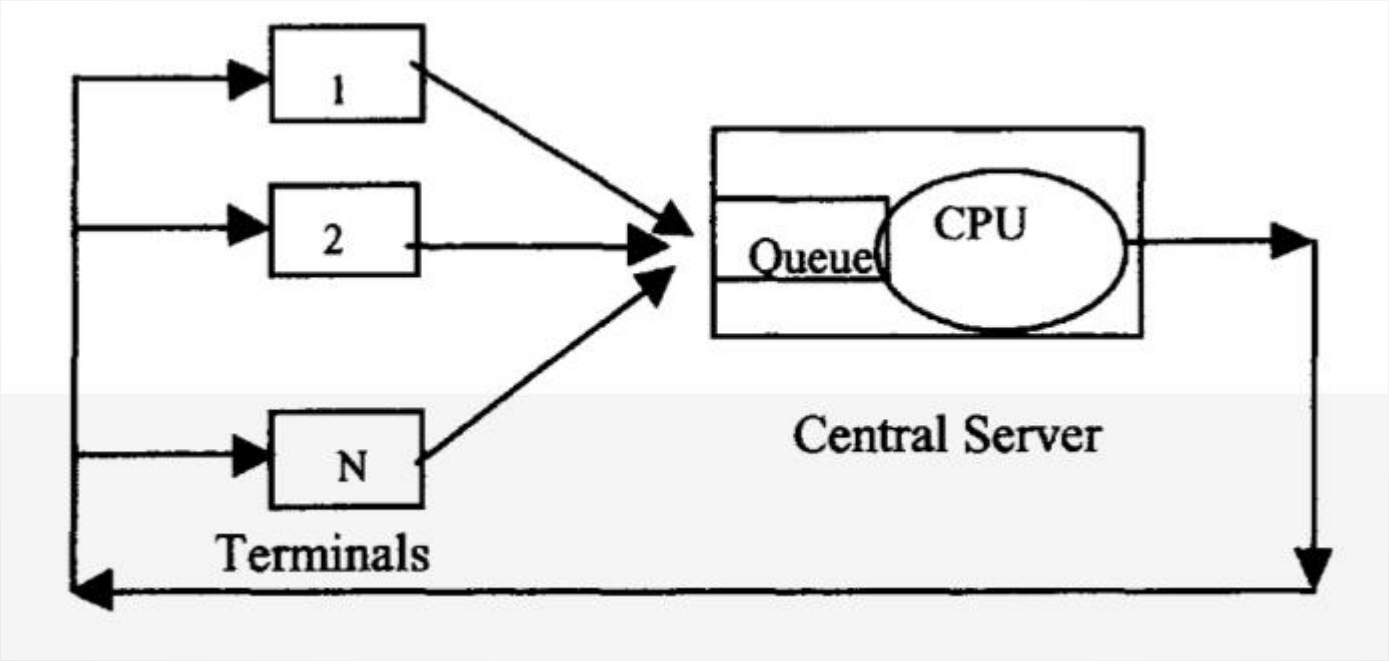
$$\bar{t}_s = \frac{1/\mu}{1-\rho} = \frac{1-p}{(1-p)\mu - \lambda_0}, \quad (\rho = \lambda/\mu)$$

- Mean number in the system

$$\bar{n} = \frac{\rho}{1-\rho} = \frac{\lambda_0}{(1-p)\mu - \lambda_0}$$

Server Model

- Ex. Each client submits a new request every 1/λ time units on average, The processor of the central server processes a request in a mean time of 1/μ, referred to as the service time




Thank You!