Chapter 3

Simulation and Simulink

Chapter 3: Simulation and Simulink

This chapter discusses some basic concepts in simulation. Simulation can be divided into 3 divisions, namely model design, model execution, and execution analysis. The cell flow analysis will be taken through the three stages and a simulation study will be performed later. Section 3.4 explains the features of a tool that will be used in the simulation. The tool, Simulink is an ideal tool to perform dynamic cell modeling and it has extensive components both reusable and custom made. Its graphic facility makes the process visible.

3.1 Simulation

3.1.1 Basic Concepts

Simulation has been defined by Shannon as "the process of designing a computerized model of a system (or process) and conducting experiments with this model for the purpose either of understanding the behaviour of the system or of evaluating various strategies for the operation of the system" [POOC93]. The foundation of simulation is the concepts of modeling. Modeling involves building a simpler representation of a phenomena. Modeling is an old concepts, an example of historical models is Newton's Second Law, F=ma, relating the force exerted on a body to its acceleration. The invention of the computer has changed the approach of model-building by changing the model form. Algorithms replace equations and a computer program is written to mimic real phenomena instead of developing mathematical theory. Model building became the core process of simulation, without building a model, no simulation can be done.

Central of any simulation study is the idea of a system. Before modeling a system, one must understand what a system is. A system can be defined as "an orderly collection of logically related principles, facts or objects" or simply "a collection of objects and interactions". In the context of a simulation study, system generally referred as "a collection of object with a well-defined set of interactions among them". In view of these definitions, we can conclude that a

model, which is static in nature, is only approximate reality and accurate in a limited range. A model will not produce reliable information on every aspect of the modeled system. On the other hand, simulation, which involves digital computer programming, will give more accurate simulated result. This is due to the fact that programming is able to take into consideration various interactions of a modeled system effectively.

3.1.2 Continuous versus Discrete

The term continuous and discrete applied to a system refer to the nature or behaviour of changes with respect to time in the system state. The states of continuous systems are changing continuously over the time. Systems that state changes are occurred at precise points in time are called discrete system. Different simulation method is required for these two kinds of system. In simulating continuous system, the variables with the simulation are continuous functions. Whereas, for simulating discrete system, the variable values change just at time of some discrete event or certain time point in simulation.

Example of continuous system is the inflow and outflow of water in a reservoir. Example of discrete system is the arrival and departure of cars at a car park. There are system called hybrid system where some of the system state may vary continuously, while others may vary discretely. Discrete event simulation can serve as an approximation of continuous simulation if the time range of event occurred is set to be small.

3.1.3 Stochastic versus Deterministic

A deterministic system is a system that its new state is completely determined by previous states and by the activity. A given activity will transit a state from one to another in a completely deterministic manner. A stochastic system is a system in which the effects of the activity is vary randomly and hence, contains a certain amount of randomness in its transitions from one state to

another. Probability of state is unpredictable, it might be determined after a given state and activity, or might not be possible known even so.

3.1.4 Monte Carlo Simulation

Monte Carlo is a technique of simulated sampling of a given distribution. It is used to generate psuedorandom number tests for validating the corresponding random number generated. The greater the sampling value, the more reliable the simulated result reflects the reality.

Monte Carlo is a method by which an inherently non-probabilistic problem is solved by a stochatic process [ERNE94]. In Monte Carlo simulation, the explicit representation of time is not required. Examples of Monte Carlo simulation are waiting-line, purchasing problems, as well as the approximate solution of differential equations and integral equations.

3.1.5 Discrete Event Simulation

The following is terminology frequent used in discrete event simulation:

- State: A variable characterizing the system such as level of stock in inventory or number of jobs waiting for processing.
- Event: An occurrence at a point of time that may change the state of system such as arrival of a customer or start of working on a job.
- Entity: An object that passes through the system such as cars in an intersection or orders in a factory. Often an event (e.g., arrival) is associated with an entity (e.g., customer).
- Queue is not only a physical queue of people, it can also be a task list, a buffer of finished goods waiting for transportation or any place where entities are waiting for something to happen for any reason.
- Creating is causing an arrival of a new entity to the system in some future time.
- Scheduling is to assign a new future event to an existing entity.

- Random variable is a quantity that is uncertain such as interarrival time between two incoming flights or number of defective parts in a shipment.
- Random variate is an artificially generated random variable.
- Distribution is the law that governs the probabilistic features of a random variable.

3.1.6 Event-Driven versus Time-Driven in Discrete Event Simulation

In event-driven simulation, simulation time is advanced in fixed increment called ticks. Time values are an increasing arithmetic sequence, $\Delta s = c$, where s denotes simulation time and c I a constant greater than zero. Short ticks can result longer simulation time in order to ensure accuracy. Longer simulation is needed because nothing may happen in short tick interval. Time-driven simulation are particularly appropriate for modeling continuous time system such as digital computer suystem. To make time-driven simulation more efficient, the tick interval can be tailored to the level of activity occuring in the system. For example lengthen the tick interval when there is little activities.

In event-driven simulation, time is moved from one event to the next. The time sequence still increases monotonically but not in an arithmetic sequence as in time-driven simulation, Δs is greater than zero. The increment time of event-driven simulation is based on the occurrence of an event that represents a change in state. Hence, a great potential for acceleration exists in event-driven simulation than in time-driven simulation. Event-driven simulation is more efficient for modeling event-oriented system such as telephone system.

3.1.7 Computer Simulation

Computer simulation is a fundamental discipline for studying complex system [FISH94]. It is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output. It can be subdivided into three areas:

- Model design
- Model execution
- Execution analysis

Model design involves creating a representation of the physical object in mathematical model. The mathematical model can be in forms such as declarative, functional, constraint, spatial or multimodel. Then the model will be programmed and operate on a computer by running the executable computer program. The program will update the state and event variables of the mathematical model as it steps through time. The end result of the operation will be gathered and analysis to produce meaningful outcome.

3.2 Probability Concepts

3.2.1 Overview

Probability is concerned with the likelihood one particular event will occur. The value of probability of event A, P (A), is a real number between 0 and 1. 0 indicates that the event will never be happened and 1 means that the event will happen certainly. Development of a probability model is important in simulation, this is due to the inherent unpredictability of real-world system behaviour. Observation or measurement in a physical system is often varying depending on factors that taken into consideration, such as time of observation. These variations often appear to be random or chance variations. Generally, probability model is used for describing a system to resolve the problem. The probability models are usually developed through experimentation. Experiments are performed on the system, results of experiments are noted. A model is then built based on the results produced. Additional experiments are conducted for validating the model. More experiments performed will reflect more accurately the system. Probability model is important especially in simulating stochastic system.

3.2.2 Random Variables

A random variable assigns a value to each of many outcomes of an experiment. It associates a real number with each possible event. There are two kinds of random variables:

- Continuous

  It takes on a noncountably infinite number of distinct values. It can be described by either its distribution function $F(x)$ or density function $f(x)$:

  $$F(x) = \Pr[X \le x]; \qquad F(-\infty) = 0; \qquad F(\infty) = 1$$

  $$f(x) = \frac{d}{dx}F(x) \qquad F(x) = \int_{-\infty}^{x} f(y)dy \qquad \int_{-\infty}^{\infty} f(y)dy = 1$$

  Mean value,

  $$E[X] = \mu_X = \int_{-\infty}^{\infty} xf(x)dx$$

- Discrete

  It takes on a finite or countably infinite number of values. Its probability distribution is characterized by

  $$P_X(k) = \Pr[X = k]$$

  $$\sum_{\text{all } k} P_X(k) = 1$$

  Mean value,

  $$E[X] = \mu_X = \sum_{\text{all } k} k \Pr[x = k]$$

3.2.3 Important Distributions

Distribution plays an important role in queueing analysis, several common distributions are described below:

- **Exponential Distribution**

The exponential distribution with parameter λ ($\lambda > 0$) is given by

$F(x) = 1 - e^{-\lambda x}$ Distribution

$f(x) = \lambda e^{-\lambda x}$ Density $x \geq 0$

$E[X] = \sigma_X = 1/\lambda$ Mean value

This distribution is referred as a random distribution when referring to a time interval such as a service time. This is because, for a particular time interval, time for its interval to be finished is likely to be equal. It is important in queueing theory because the service time of a server in a queueing system can always be assumed as exponential. For instance, service time for telephone traffic is the time for which a subscriber engages the equipment of interest. There is not sound theoretical reason why services time should be exponential, but the fact is that in most cases they are very nearly exponential. This can simplify the queueing analysis.

- **Poisson Distribution**

The Poisson distribution with parameter λ ($\lambda > 0$), which takes on values at the points 0,1…:

$$\Pr[X = k] = \frac{\lambda^k}{k!} e^{-\lambda} \qquad k = 0,1,2.....$$

$E[X] = \mathrm{Var}[X] = \lambda$ Mean value

Poisson distribution is also important in queueing analysis because a Poisson arrival must be assumed so that the queueing equation can be developed. Fortunately, the assumption of Poisson arrival is usually valid. If items arrive at a queue according to a Poisson process, this may be expressed as

$$\Pr[k \text{ items arrive in time interval } T] = \frac{(\lambda T)^k}{k!} e^{-\lambda T}$$

Expected number of items to arrive in time interval $T = \lambda T$

Mean arrival rate, in items per second = λ

Arrivals occurring according to a Poisson process are often referred as random arrivals. This is because the probability of arrival of an item in a small interval is proportional to the length of the interval and is independent of the amount of elapsed time since the arrival of the last item. That is, when items are arriving according to a Poisson process, an item is as likely to arrive at one instant as any other, regardless of the instants at which the other customer arrive.

- **Normal Distribution**

The normal distribution with parameter λ and σ ($\mu > 0$) has the following density function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

with

$E[X] = \mu$	Mean value
$Var[X] = \sigma^2$	Variance

3.3 Queueing Theory

3.3.1 Queueing System

Queueing is one of the most common problems encountered in modeling or simulating the operation of a system. A queueing system is a system in which request of service arrive, wait in a queue if the service is not immediately available, being served after some time of waiting in the queue, and then depart from the service facility.

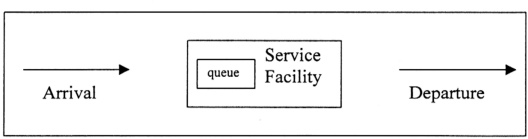


Figure 3.1 Queueing System

3.3.2 Queueing Theory

Queuing theory plays an important role in explaining the average delay of an application with certain equivalent bandwidth that causes by queuing for service of a switch. At each node in the network, queues are developed as items arrive and wait for service. The main factor of time delay in the network is the queuing delay due to the waiting for service.

A queueing system need to consider:

- The time taken by the server to process the item.
- The rules for the formation of the queue of items requiring service.
- The rules for selecting an item for service by the server from the queue.

3.3.3 Queueing Models

Items arrive at the facility at the average rate λ. After a certain time, some items will be waiting in the queue; the average number waiting is ω and the mean time the item has to wait is T_{ω}. The servers handles incoming items with an average service time T_S. Utilization ρ is the fraction of time that the server is busy. The average number of items in the system is q and the average time that an item spends in the system is T_q. Figure 3.2 and Figure 3.3 below will illustrate more clearly some of the important parameter associated with a queuing model.

Assume that the arrival rate follows the Poisson distribution, where the interarrival times are exponential, which the arrivals occur randomly and independent to each other. For an arrival rate of λ and the average time between arrivals is $1/\lambda = T$, then

$$P(T > t) = P(\text{zero arrival in time } t) = e^{-\lambda t}$$

From which the exponential distribution function,

$$F(t) = P(T \leq t) = 1 - P(T > t) = 1 - e^{-\lambda t}$$

Assume that the queue is infinite. As the arrival rate passing through the system increases, ρ increases and going to get congestion. At the same time increase the queue length and T_ω. When $\rho=1$, the server becomes saturated. Therefore the maximum input rate becomes $\lambda_{max}=1/T$.

There are two types of queueing system:

- Single-server queue

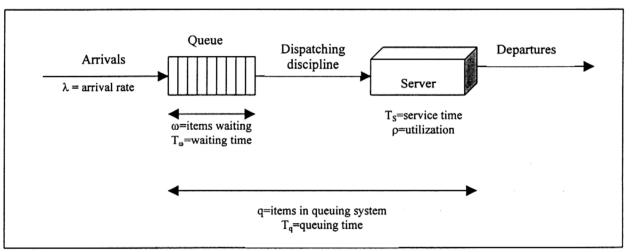


Figure 3.2 Queueing system structure and parameters for single-server queue

If the server is idle, an item is served immediately or an arrival item will be at the waiting line. When the server has completed serving an item, the item departs.

For equation $\rho = \lambda T_S$, the arrival rate of λ where the average time between arrivals is $1/\lambda = T$. So if T is less than T_S (more items arrive than for it to service, then during a time interval), the server is busy for a time T_S for a utilization of $T_S/T = \lambda T_S$.

For equation $q = \omega + \rho$, at any time, the number of items in the system is the sum of the number of items waiting, ω plus the number of items being served, ρ.

- Multiserver queue

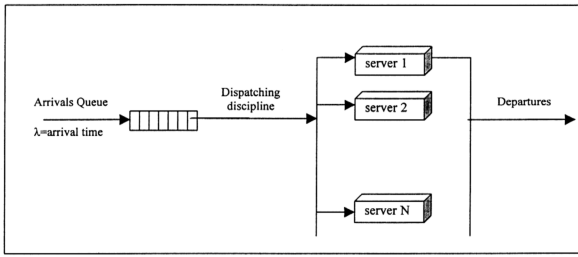


Figure 3.3 Queueing system structure for multiserver queue

If the items arrive and a least one server is available, the item is immediately dispatched to that server. Assume that all the servers are the same. If all the servers are busy, a queue is going to be formed. As soon, one of the servers become free, an item is dispatched from the queue.

If there are N servers, then $\rho = \lambda T_S/N$ is the utilization of N server and therefore $N\rho$ will be the utilization of the entire system or the traffic intensity μ. For equation $q = \omega + N\rho$, at any time, the number of items in the system is the sum of the number of items waiting, ω plus the number of items being served for N servers, $N\rho$.

3.3.4 Queueing Discipline

The queue discipline considers the techniques that are selected from the queue for service. The most common queue discipline is FIFO (first-in, first-out). Packets are scheduled in order when they entered the queue where FIFO serves packets in the order of arrival.

In the FIFO buffer, the average output rate a connection is determined by the relative proportion of packets from the connection in the buffer. Let μ_i and χ_i be the output rate and the buffer occupancy respectively of VC_i. Let μ and χ be the total output rate and the buffer occupancy respectively. Because the buffer is FIFO,

$$\frac{\chi_i / \chi}{\mu_i / \mu} = 1$$

If the buffer occupancy of every active VC is maintained at a desired threshold, then the output rate of each VC can be controlled. That is, if a VC has χ_i cells in the buffer with a total occupancy of χ cells; its average output rate will be at least $\mu\chi_i / \chi$.

However, there are several drawbacks to the FIFO queuing discipline:

- No special treatment is given to packets of higher priority or more delay sensitive
- Flows of larger packets get better service
- FIFO queuing cannot isolate packets from various VCs at the egress of the queue
- FIFO provides no protection against uncooperative users because all users experience the same delay on average even the overload is caused by a small subset of the users

Lead to unfairness even when all the end users behave properly.

3.4 Computer Simulation using Simulink

3.4.1 What is Simulink?

Simulink is a software package for modeling, simulating, and analyzing dynamical systems. It supports linear and non-linear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate, which means it can have different parts that are sampled or updated at different rates.

Simulink provides a graphical user interface (GUI) for building models as block diagrams by using click-and-drag mouse operations for modeling. This is the main advantage of Simulink over other simulation packages that require the formulation of differential equations and different equation in a language or program.

Simulink includes a comprehensive block library of sinks, sources, linear, and non-linear components and connectors. One can also customize and create their own blocks by writing S-functions (system functions) that will be discussed more detail later.

Models are hierarchical built by using both top-down and bottom-up approaches. The system can be viewed at a high level, then double-click on blocks to go down through the levels to see increasing level of model details. This approach provides insight into how a model is organized and how its parts interact.

Simulation can be started after defining a model, either from the Simulink menus or by entering commands in Matlab's command window. Using display blocks provided, the simulation result can be seen while simulation is running. In addition, the parameters can be changed to see the effects immediately, for "what if" exploration. The simulation results can be put in the Matlab workspace for post-processing and visualization.

Model analysis tools include linearization and trimming tools, which can be accessed from the Matlab command line, plus the many tools in Matlab and its application toolboxes. Because Matlab and Simulink are integrated, the model can be simulated, analyzed and revised in either environment at any point.

3.4.2 Why use Simulink?

- By using Simulink, models can be built from scratch easily, or take an existing model and add to it.
- Simulation in Simulink are interactive, parameter can be changed "on the fly" and immediately see what happens.
- All of the analysis tools in Matlab can be accessed instantly through Simulink.
- With Simulink, one can move beyond idealized linear models to explore more realistic non-linear models such as factoring in fraction, air resistance and other things that describe real-world phenomena.
- New blocks can be created and using masks to customize their appearance and use in Simulink.
- Subsystems where execution depends on triggering signal aids in simplifying the model design.
- Simulink debuggers are provided to debug Simulink models in an effective way.

3.4.3 S-function (System function)

3.4.3.1 What is an S-function?

An S-function is a computer language description of a dynamic system. S-function can be written using Matlab or C language. C language S-functions are compiled as MEX-files using the MEX utility. As with other MEX-files, they are dynamically linked into Matlab when needed.

S-functions use a special calling syntax that enables one to interact with Simulink's equation solvers. This interaction is very similar to the interaction that takes place between the solver and built-in Simulink blocks. The form of an S-function is very general and can accommodate continuous, discrete, and hybrid systems. As a result, nearly all Simulink models can be described as S-functions.

S-functions are incorporated into Simulink models by using the S-functions block in the Non-linear Block sublibrary. Simulink's masking facility can be used to create custom dialog boxes and icons for the S-function blocks. Masked dialog boxes can make it easier to specify additional parameters for S-functions.

3.4.3.2 Why use S-function?

- S-function provides a powerful mechanism for extending the capabilities of Simulink.
- S-function allows one to add their own algorithms in Matlab or C language following a set of simple rules.
- The user interface of an S-function can be customized by using masking

3.4.3.3 When to use an S-function?

The most common use of S-function is to create custom Simulink block. S-function can be used or a variety of application, including:

i. Adding new general purpose block to Simulink
ii. Incorporating existing C code into a simulation
iii. Describing a system as a mathematical set of equations
iv. Using graphical animations

An advantage of using S-function is that a general purpose block built can be used many times in a model, varying parameters with each instance of the block.

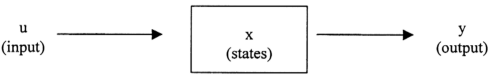
3.4.3.4 How S-function works?

Each block within a Simulink model has the following general characteristics:

i. a vector of inputs, u

ii. a vector of outputs, y

iii. a vector of states, x



The state vector may consist of

a) continuous states

b) discrete states or

c) combination of both

The mathematical relationships between the inputs, outputs and the states are expressed by the following equations:

$y = f_o(t, x, u)$ - output
$x_c = f_d(t, x, u)$ - derivation
$x_{d\,k+1} = f_u(t, x, u)$ - update

where $x = x_c + x_d$

Simulink makes repeated call during specified stages of simulation to each block in the model, directing it to perform tasks such as

- computing its outputs
- updating its discrete states
- computing its derivations

Additional calls are made at the beginning and end of a simulation to perform initialization and termination tasks.

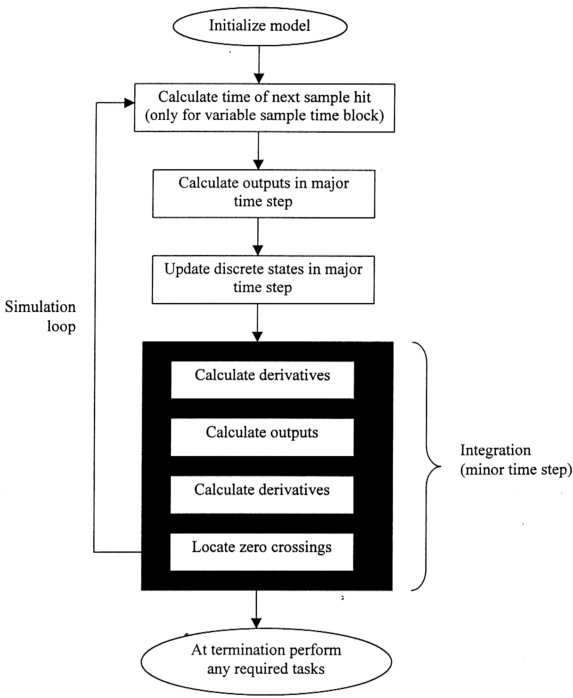


Figure 3.4 How Simulink Performs Simulation

Simulink makes repeated calls to S-functions in the model. During these calls, Simulink calls S-function routines (also called methods), that perform tasks required at each stage. These tasks include:

- Initialization – prior to the first simulation loop, Simulink initializes the S-function. During this stage, Simulink:
  - Initializes the *SimStruct*, a simulation structure that contains information about the S-function.
  - Sets the number and size of input and output ports.
  - Sets the blocks sample time(s).
  - Allocates storage areas and the *sizes* array.
- Calculation of the next sample hit – If a variable step integration routine is selected, this stage calculates the time of the next variable hit, that is, it calculates the next stepsize.
- Calculation of outputs in the major time step – After this call is completed, all the output ports of the blocks are valid for the current time step.
- Update discrete states in the major time step – In this call, all blocks should perform once-per-time-step activities such as updating discrete states for next time around the simulation loop.
- Integration – This applies to models with continuous states and/or non-sampled zero crossings. If the S-function has continuous states, Simulink calls the output and derivatives portions of the S-function at minor time steps. This is so Simulink can compute the state(s) for the S-function. If the S-function (C MEX only) has non-sampled zero crossings, then Simulink will call the output and zero crossings portion of your S-function at minor time steps, so that it can locate the zero crossings.

Table 3.1 Simulation Stages

Simulation stage	S-function routine
Initialization	*mdlInitializeSizes*
Calculation of next sample hit (optional)	*mdlGetTimeOfNextVarHit*
Calculation of outputs	*mdlOutput*
Update discrete states	*mdlUpdate*
Calculation of derivatives	*mdlDerivatives*
End if simulation tasks	*mdlTerminate*

3.4.3.5 S-function Concepts

- Direct feedthrough

  Direct feedthrough means that the output or the variable sample time is controlled directly by the value of an input port. An S-function input port has direct feedthrough if:

  - The output function (*mdlOutputs*) is a function of the input *u*. That is, there is direct feedthrough if the input *u* is used in the equations defined in *mdlOutputs*. Outputs may also include graphical outputs, as in the case of an XY Graph scope.
  - It is a variable sample time S-function (calls *mdlGetTimeOfNextvarHit*) and the computation of the next sample hit requires the input *u*.

  An example of a system that requires its inputs is the operation $y = k \times u$, where u is the input, k is the gain, and y is the output.

- Dynamically sized inputs

  S-function can be written to support arbitrary input widths. In this case, the actual input width is determined dynamically when a simulation is started by evaluating the width of the input vector driving the S-function. The input width can be used to determine the number of the continuous states, the number of discrete states, and the number of outputs.

  To indicate that the input width which id dynamically sized, specify a value of –1 for the appropriate fields in the *sizes* structure, which is returned during the *mdlInitializeSizes* call.

The actual input width can be determined when the S-function is called by using *length(u)*. If a width of 0 is specified, then the input port will be removed from the S-function block.

- Setting sample times and offsets

  Setting sample times and offsets facility allow a high degree of flexibility in specifying when an S-function executes. Simulink provides the following options for sample times:

  - Continuous sample time – For S-function that have continuous states and/or non-sampled zero crossings. For this type of S-function, the output changes in the minor time steps.
  - Continuous but fixed in minor time step sample time – For S-functions that need to be executed at every major simulation step, but do not change value during minor time steps.
  - Discrete sample time – If the S-function block's behaviour is a function of discrete time intervals, a sample time to control can be defined when Simulink calls the block. Offset can also be defined to delay each sample time hit. The value of the offset cannot exceed the corresponding sample time. If a discrete sample time is defined, Simulink calls the S-function *mdlOutput* and *mdlUpdate* routines at each sample time hit.
  - Variable sample time – A discrete sample time where the intervals between sample hits can vary. At the start of each simulation step, S-functions with variable sample times are integrated for the time of next hit.
  - Inherited sample time – Sometimes an S-function block has no inherent sample time characteristics (that is, it is either continuous or discrete, depending on the sample time of some other block in the system). The block's sample time is specified as *inherited*. A block can inherit its sample time from:
    - The driving block
    - The destination block
    - The fastest sample time in the system

    To set a block's sample time as inherited, use –1 as the sample time.

This chapter briefly explained the fundamental concepts in simulation with particular emphasis in the use of the tool Simulink. Due to a significant absence of components relevant to dynamic modeling of cell flow for an ATM network, custom made components using S-function have to be designed. This chapter ends with an explanation of the concept of S-function.