Unit IV: Discrete System Simulation

Introduction:

The system in which changes are discontinuous is called discrete system. Each change in a state of a system is called an event. For example, the arrival or departure of a customer in a queue is an event. Similarly, sales of an item from stock is an event in the inventory system. Therefore, the discrete system is often referred to as a discrete system simulation.

Representation of Time:

It refers to a number that records the passage of time. It is usually set to zero at the beginning of a simulation and subsequently indicates how many units of simulated time have pass since the beginning of the simulation.

Simulation Time:

It refers to the indicated clock time, not the time that a computer has taken to carry out the simulation. The ratio of simulated time to the real-time taken can vary to a great extent depending on the following factors:

a. Nature of the system being simulated.

b. The detail to which it is modelled.

Example: The simulation of an atomic model. Changes occur in a fraction of a microsecond in a real system. However, the simulation of these atomic models in a computer may take 1000 time more time than in the actual system.

Updating Clock Time:

There are two basic methods used to update clock time:

i. Event Oriented Method:

In these methods, the clock is advanced to the time at which next event is due to occur. Discrete system simulation is usually carried out by using the event-oriented method.

ii. Interval Oriented Method:

In these methods, the clock time is advanced by small and usually, a uniform interval of time and it is determined at each interval whether an event is due to occur at that time. Continuous system simulation normally uses the interval oriented method. But it should be noted that no firm's (visit) rule can be made about the way the time is represented in simulation for discrete and continuous system because:

a. An interval oriented program will detect discrete changes and can, therefore, simulate a discrete system.

b. An event-oriented program can be made to follow continuous changes by artificially introducing events that occur at a regular time interval and can, therefore, simulate a continuous system.

Significant Event Simulation:

It is another method to represent the passage of time and applies to a continuous system in which it is quiescent (continuous) period. A quiescent period is an interval between events in the event-oriented approach but it involves the model's representation of the system activities which create a notice of the event that terminates the interval. The significant event approach assumes that a simple analytic function can be used to project the pan of a quiescent period. The significant event is the one with the least span.

Example: an automobile travelling at constant acceleration, it movement might result in a significant event for several reasons:

a. It might reach at the end of the road.

b. Its velocity most reach some limit.

c. It might come to rest.

If the initial conditions are known, the elapsed time for each these possible events can be calculated from a simple formula.

General Arrival Pattern:

The generation of exogenous arrival is an important aspect of discrete system simulation. An exact sequence of arrivals may have been specified for the simulation. For example, in the simulation of the electronic circuit, a particular sequence of signal might be used as the simulation input to verify the circuit.

Trace-Driven Simulation:

It refers to the process of gathering a sequence of inputs based on the observation of a system. When there is no interaction between exogenous arrivals and the endogenous events of a system, it is permissible to create a sequence of arrival in preparation for the simulation. Usually, however, the simulation proceeds by creating new arrivals as they are needed.

Bootstrapping:

It is the process of making one entity creates its successor. These methods require keeping only the arrival time of the next entity, it is, therefore, the preferred method of generating arrivals for computer simulation programs.

Simulation of Telephone System:

The simulation of a discrete system can be explained by simulating a telephone system as below:

The system has several telephones (here only 8 are shown), connected to a switchboard by line. The switchboard has several lines provided the condition that only one connection at a time can be made to each line. Any call that cannot be connected at the time it arrives is immediately abandoned, and then the system is called **a lost call system**. A call may be lost due to the following reasons:

a. If the called party is engaged (busy)

b. If no link is available (a blocked call)

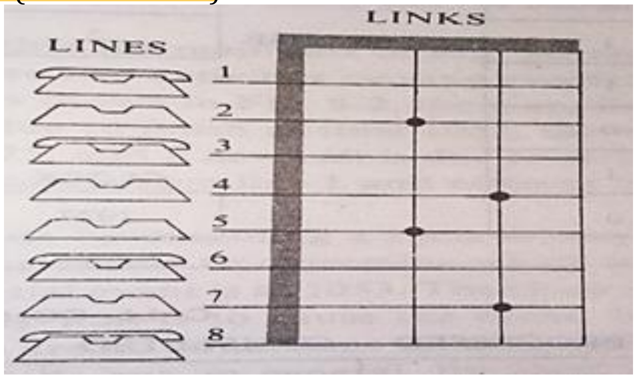


Figure 1: Simple TelephoneSystem

Object of Simulation:

To process a given number of calls and determine what, portion is successfully completed, blocked or found to be busy calls. Let's consider the current state of the system as below:

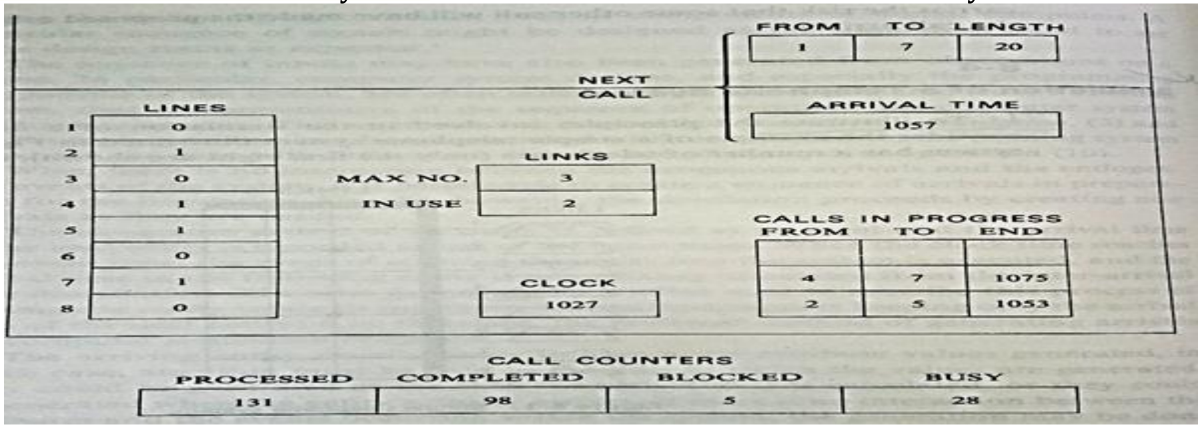


Figure 2

Here, line 2 is connected to line 5 and line 4 is connected to line 7. 0 in the line tables means is free and 1 means, it is busy. The maximum number of links, in this case, is only 3 and 2 of them are in use. A number representing clock time is included to keep track of events. In this state, the clock time is shown to be 1027. The clock will be updated to the next occurrence of the event as the simulation proceeds. Each column has its own attributes; its origin, destination, length and the time at which it calls finishes.

The "call-in progress" tables show the lines that are currently connected and finish time. Arrival time and detail of next call are also shown to generate the arrival of calls; here bootstrap method can be used. It is assumed that the call is equally likely to come from any line, i.e. not busy at the time of arrival and that is can be directed to any line other than itself. To make easy to explain, the action of the simulation the attributes of next events are shown in figure 2, although in real practice, they would not have been generated until the clock reaches 1057, the clock of next arrival.

Two possible attributes can make to change to occur in the system state; a new call can arrive or an existing call can be finished. From figure 2, it can be seen that three events can occur in the figure:

a. Call between 2 & 5 will finish at 1053
b. A new call will arrive at 1057
c. Call between 4 & 7 will finish at 1075

The simulation proceeds by executing a cycle below state to simulate each event:

1. Scans the events to determine which the next potential event is. Here, it is at 1053. The clock is updated then.

2. Select the activity that is to cause the event. Here, the activity is to disconnect a call between lines 2 & 5.

3. Test whether the potential event can be executed. Here, however, there are no conditions to disconnect a call.

4. Change the record to reflect the effect of the event. Here the call is shown to be disconnected by setting to zero in the line table for line 2 & 5, reducing the number of links is used by 1 and removing the finish call from the "call-in progress" table.

5. Gather some statistics for the simulation output. Here call counter is changed to record the number of calls that have been processed, completed, blocked or busy. The state of the system appears as shown in figure 3.

6. The above steps are repeated to continue the simulation.

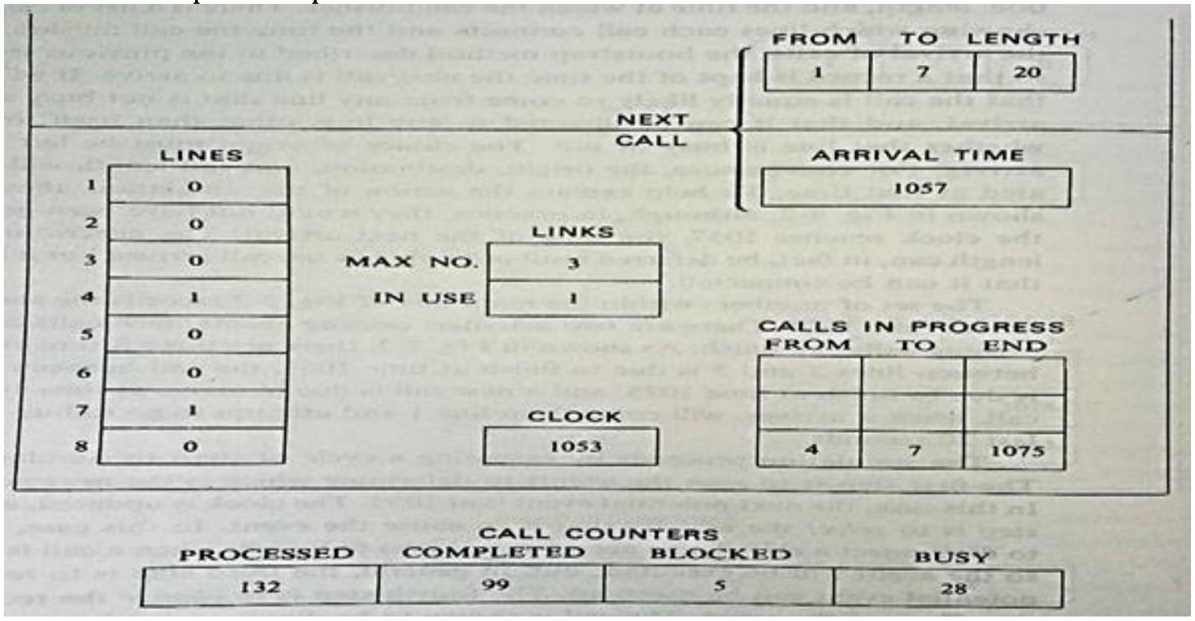
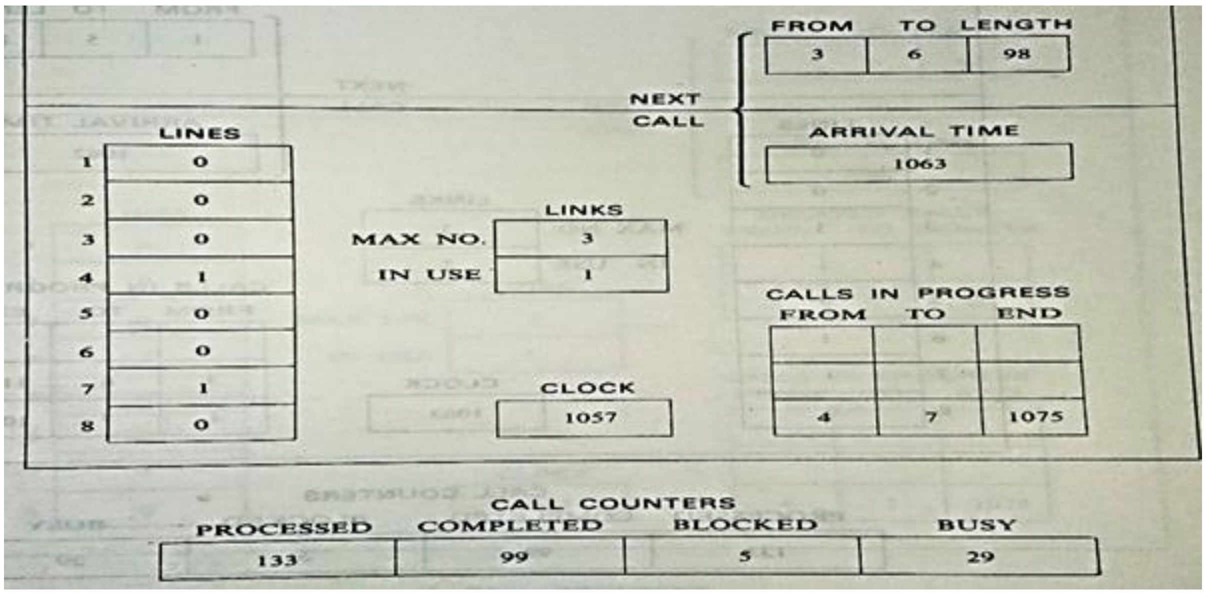


Figure 3

In the figure, it can be seen that the next potential event is the arrival of a call at time 1057 sec. the clock is updated to 1057 and the attributes of the new arrival are generated. Since the selected activity is to connect a call, it is necessary to test; to find whether a link is available and to find whether the party is busy or not. In this case, the called party, line 7 is busy so the call is lost. Both the processed call and the busy call counter are increased by one. New arrival is generated. These are shown in figure 4.



NEXT CALL

FROM	TO	LENGTH
3	6	98

ARRIVAL TIME
1063

LINES	
1	0
2	0
3	0
4	1
5	0
6	0
7	1
8	0

	LINKS
MAX NO.	3
IN USE	1

CLOCK
1057

CALLS IN PROGRESS

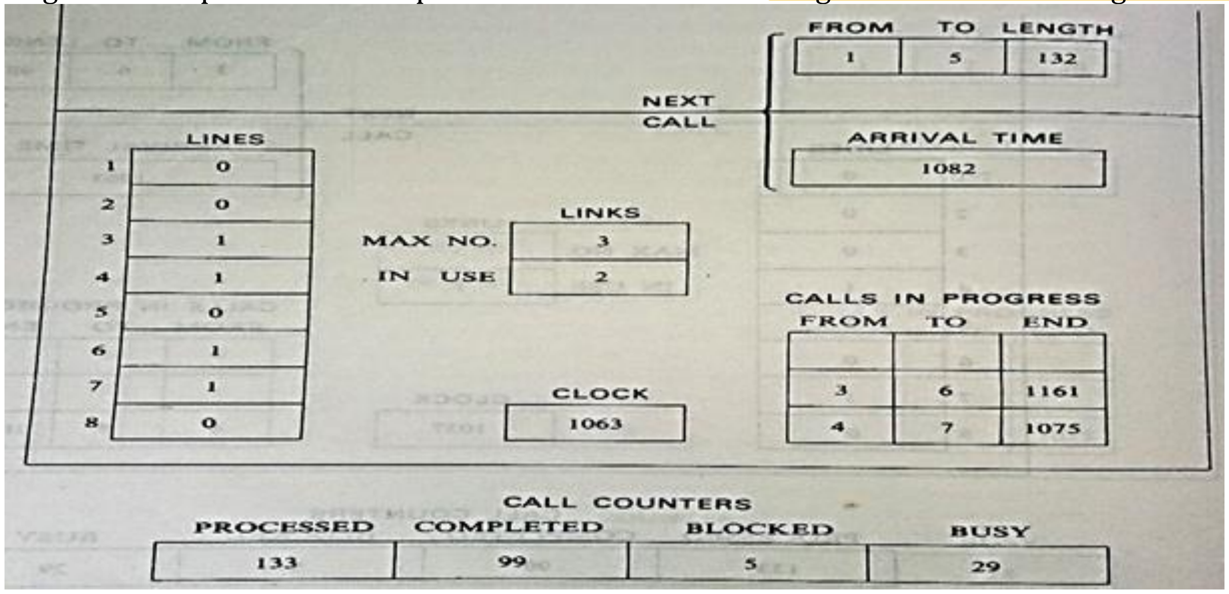
FROM	TO	END
4	7	1075

CALL COUNTERS

PROCESSED	COMPLETED	BLOCKED	BUSY
133	99	5	29

Figure 4

Suppose the next arrival time is 1063, and the call will be from line 3 to 6 and will spend 20 sec and at this time, the arriving call can be connected. The new state of the system is shown in figure 5. The procedural is repeated to a certain limit until good statistics can be gathered.



NEXT CALL

FROM	TO	LENGTH
1	5	132

ARRIVAL TIME
1082

LINES	
1	0
2	0
3	1
4	1
5	0
6	1
7	1
8	0

	LINKS
MAX NO.	3
IN USE	2

CLOCK
1063

CALLS IN PROGRESS

FROM	TO	END
3	6	1161
4	7	1075

CALL COUNTERS

PROCESSED	COMPLETED	BLOCKED	BUSY
133	99	5	29

Figure 5

Delayed Call:

When the telephone system is modified so that calls that cannot be connected are not lost, and then they will wait until they can be connected later. Such calls are referred to as a delayed call. We know that it is not possible in the case a real-time telephone system but possible to message in a switching system that has a store and forward capability.

To keep a record of the delayed call, it is necessary to build another list like the call in progress list. After arriving a call, if it cannot be connected, then it is placed in the delayed call list, waiting to next time. When a previous call is completed, it is necessary to check the delayed call list to find if a waiting call can be connected. This is illustrated in figures 6, 7, and 8.

From figure 5, it is clear that the next potential event is the arrival of the next call from line 1 to 7. Since the line 7 is not free the next call cannot be executed. It is then placed in the delay call list waiting for the next time and the new stated of the system is shown in the below figure.

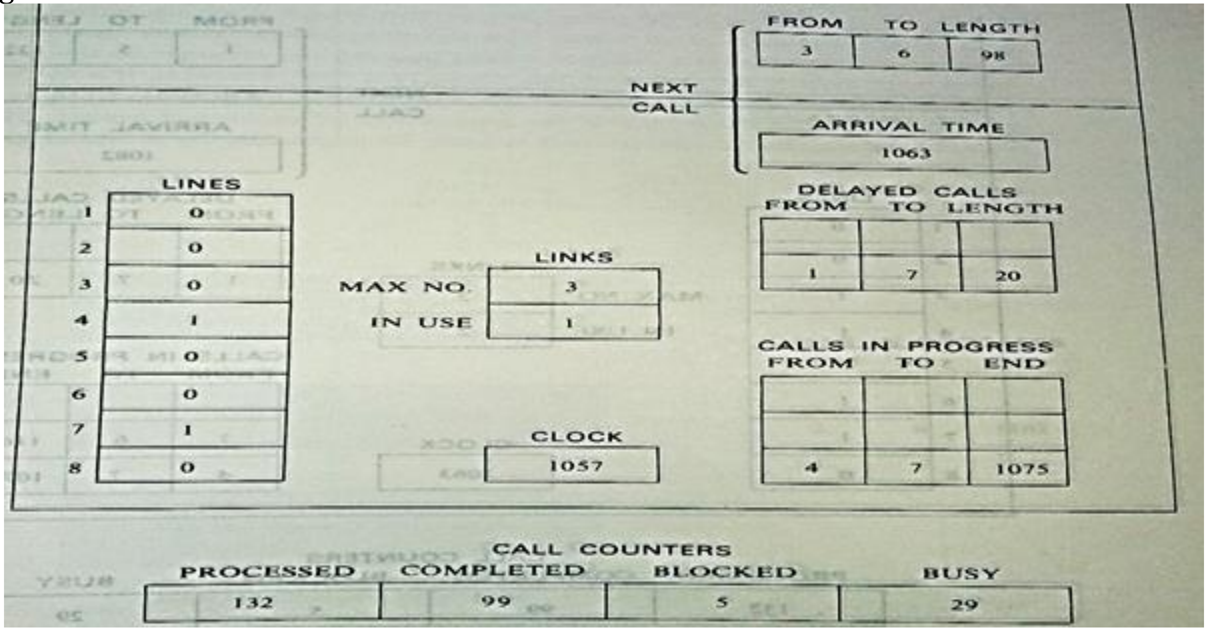


Figure 6

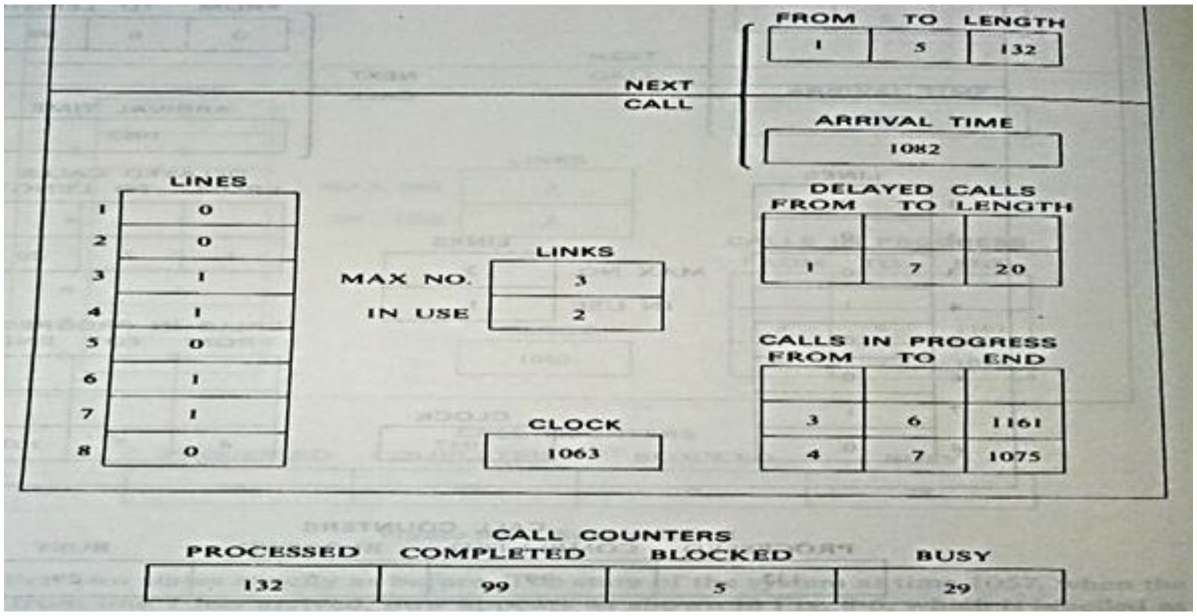


Figure 7

https://www.ckundan.com.np/2019/06/discrete-system-simulation.html
When the call from line 4 and 7 completed at time 1075, then for the next event, the delayed call list is checked first. At these time, lines 1 and 7 can be connected which is shown in the below figure.

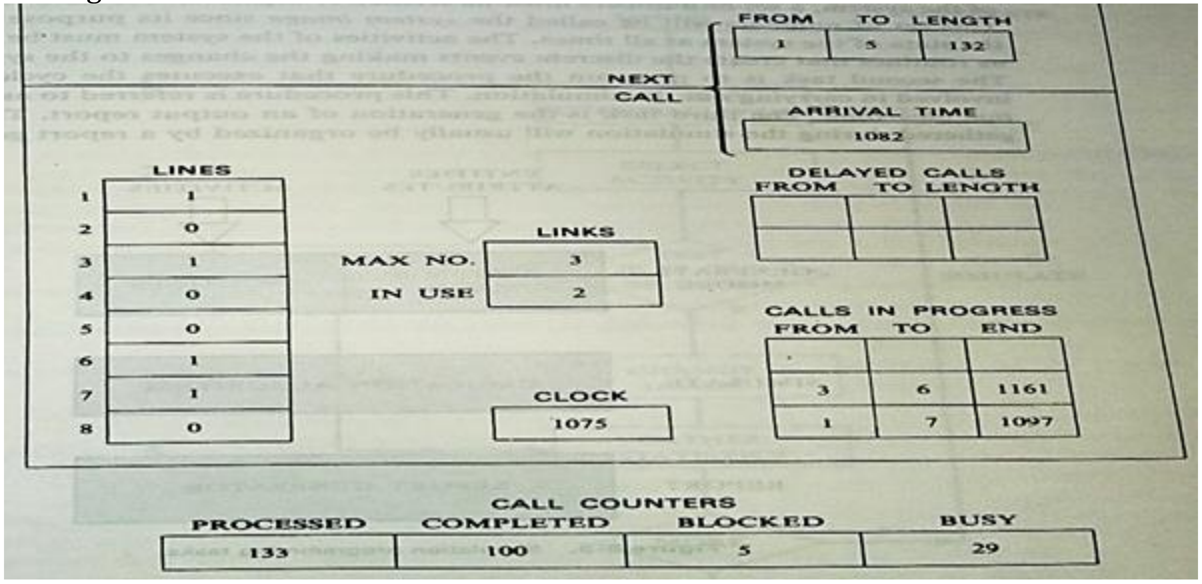


Figure 8

Simulation Programming Task:

There are mainly three tasks to be performed in simulation programming. They are:

1. Generating a Model:

The first task in simulation programming is to generate a model and initialize it. In this state, a set of several most created to represent the state of the system. Thus, this set of number is called the system image. The activities that occur in the system are in routines.

2. Simulation:

After generating a model, the next is to program the procedure that executes the cycle of actions involved in carrying out the simulation. This procedure is referred to as a simulation algorithm.

3. Generating Report:

After programming the simulation algorithm, the next and final task is to run the simulation to generate an output report. The statistic gather (data collected) during the simulation will be organized by a report generator.

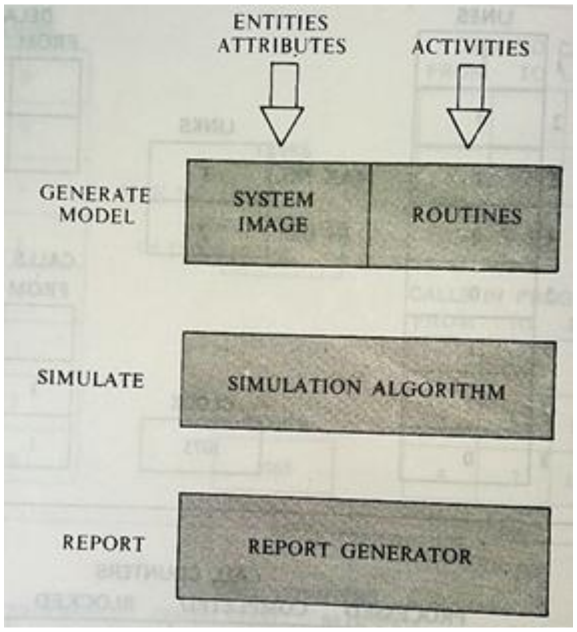


Fig: Simulation Programming Tasks

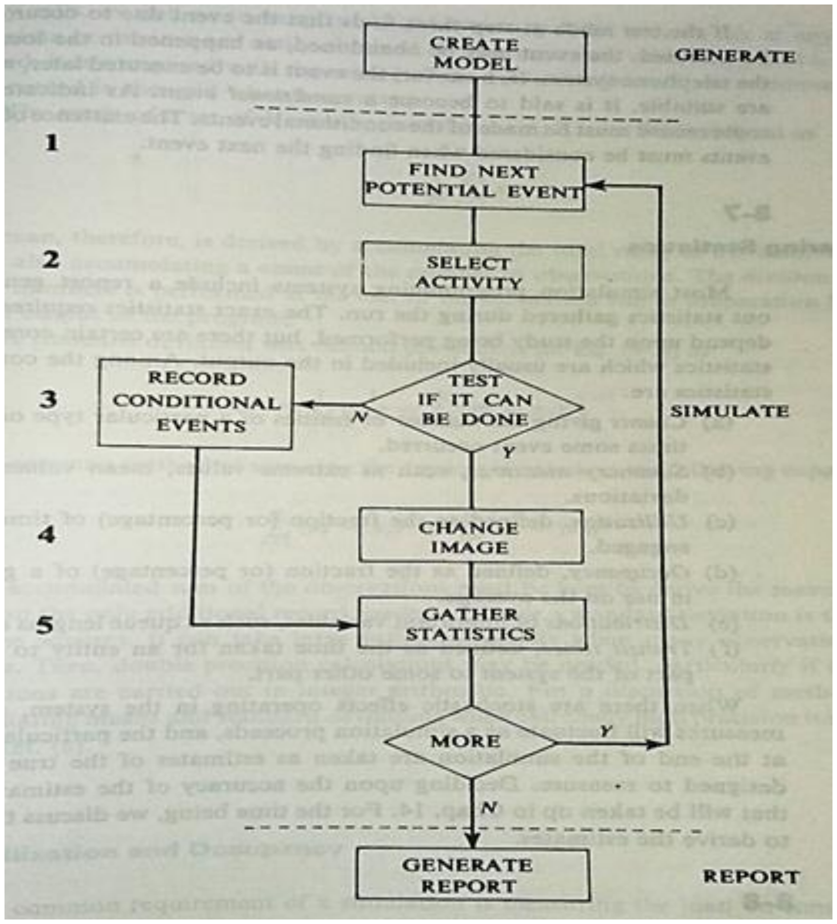


Fig: Execution of a Simulation Algorithm

The general flow of control during the execution of the simulation program is shown above. At the top of the figure is the task of generating a model which is executed once. At the bottom is the report generation task, which is usually executed once at the end of the simulation. Carrying out the simulation algorithm involves repeated execution of five steps. These steps are:

1. Find the next potential event.
2. Select an activity.
3. Test if the event can be executed.
4. Change the system image.
5. Gather statistics.

Gathering Statistics:

To print out the statistics gathered during the run, the simulation programming system includes a report generator. The exact statistics required from a model depend upon the study being performed. Some of the commonly required statistics which is usually included in the output during the simulation are as follows:

a. Counts: This gives the number of entities at a particular type or the number of times some events occurs.

b. Summary measures: This includes measuring some quantities such as extreme values, mean values, and standard deviation.

c. Utilization: This measure the time infraction or percentage, that some entity is engaged.

d. Occupancy: These give the percentage of a group of entities in use on the average.

e. Distribution: These record the distribution of important variables such as queue length or waiting time.

f. Transit time: These records the time taken from an entity to move from one part of the system to some other part.

When stochastic effects are operating in the system, all these measures will fluctuate as a simulation proceeds and the particular values reached the simulation are taken as estimates of the true values they are designed to measure.

Counter and Summary Statistics:

The counter which is the basis for most statistics are used to accumulate totals and to record current values of some level in the system. For example, in the simulation of the telephone system, the counter is used to record the total number of the lost call, busy call, and to keep track of how many links were in use at any time. Whenever a new value of a count is established, it is compared with the record of the current maximum or minimum, and the record is changed when necessary. The accumulator sum of observation must be kept to derive the mean value and standard deviation as below:

The mean of set of N observation $X_i (i = 1, 2, 3, \dots \dots \dots n)$ is given by

$$\text{Mean, } M = \frac{1}{N}\sum_{i=1}^{N} Xi$$

A mean is derived by accumulating the total value of the observation, and also accumulating a count of the number of observations.

$$\text{And, Standard Deviation, } \sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(M - X_i)^2}$$

The accumulated sum of the observations must be kept to derive the mean value, and so the additional record needed to derive a standard the deviation is the sum of the squares.

Measuring Utilization & Occupancy:

To measure the load on some entity such as an item of equipment, the simplest way is to determine what fraction/percentage of the time, the item as engaged during the simulation. Measuring those statistics is referred to as the utilization of that equipment. The time history of an equipment usage might appear as shown in the figure below:

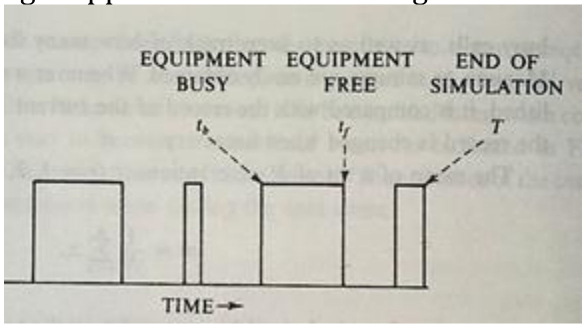


Fig: Utilization of Equipment

Let, tb= the time at which item last become busy.
tf= the time at which the item last become free.
T =total time in the simulation run.
Then, the utilization u is given by:

$$u = \frac{1}{T}\sum_{i=1}^{N}(t_f - t_b)_i$$

In dealing with a group of entities, rather than individual items, the calculation is similar, requiring that information about the number of entities involved also kept. The below figure represents as a function of time and the number of links in the telephone system that are busy. To find an average number of link in use, a record must be kept of the number of links currently in use and the time at which the last change occurred. If the number changes at time ti to the value ni, then, at the time of the next change ti+1, the quantity ni(ti+1-ti) must be calculated and added to an accumulated total. The average number in use during the simulation run, A, is then calculated at the end of the run by dividing the total by the total simulation time T, so that:

$$A = \frac{1}{T} n_i (t_{i+1} - t_i)$$

The below figure might also represent several entities waiting on queues.

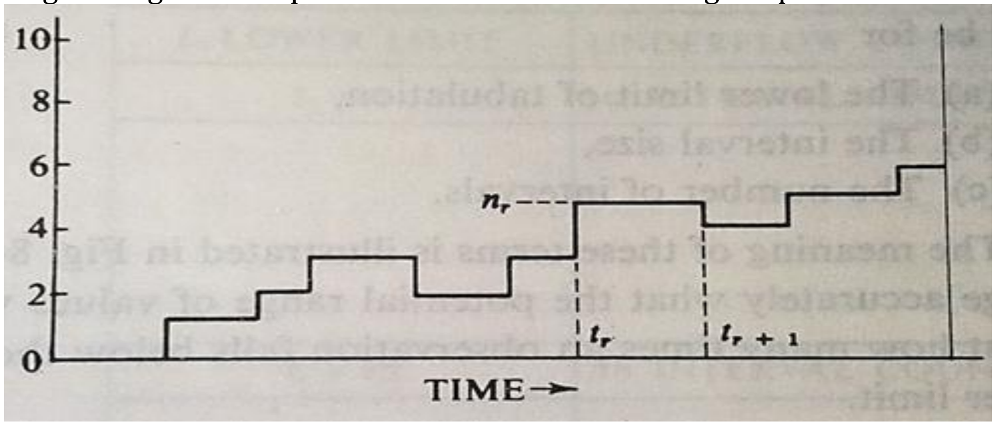


Fig: Time History of Busy Telephone Links

https://www.ckundan.com.np/2019/06/discrete-system-simulation.htm

If there is an upper limit on the number of entities, as there was a limit on several links in the telephone system then the occupancy is defined as the ratio of the average number in use to the maximum number.

$$Occupancy = \frac{Average\ no\ is\ use}{Maximum\ number}$$

If M= links in a telephone

ni = number of busy in the interval ti and ti+1 then the average occupancy, assuming the number ni changes n times is given by:

$$Occupancy\ (B) = \frac{1}{NM} \sum_{i=1}^{N} n_i (t_{i+1} - t_i)$$

Recording Distribution & Transit Time:

To determine the distribution of a variable it requires counting the number of times the value of the variable fall within specific interval. For this purpose, a table with location to define the internal and to accumulate the count has to be maintained. When an observation is made, the value is compared with the defined intervals and the appropriate must be incremented by one.

The definition of a destination table required specification for the lower limit of the tabulation, the interval size and the number of intervals. Normally, the tabulation interval sizes are uniform.

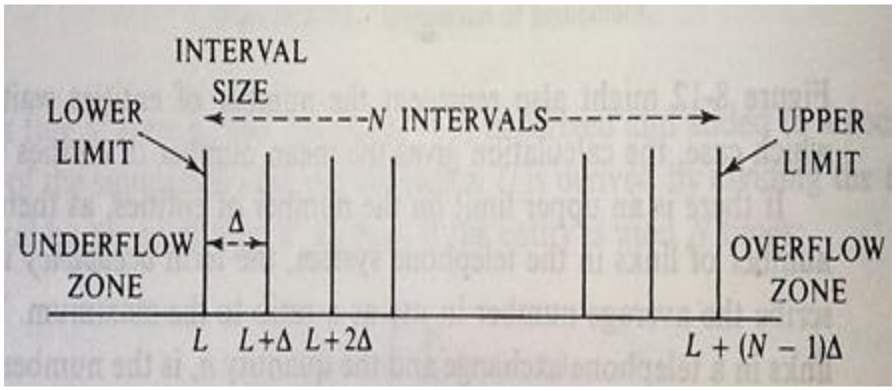


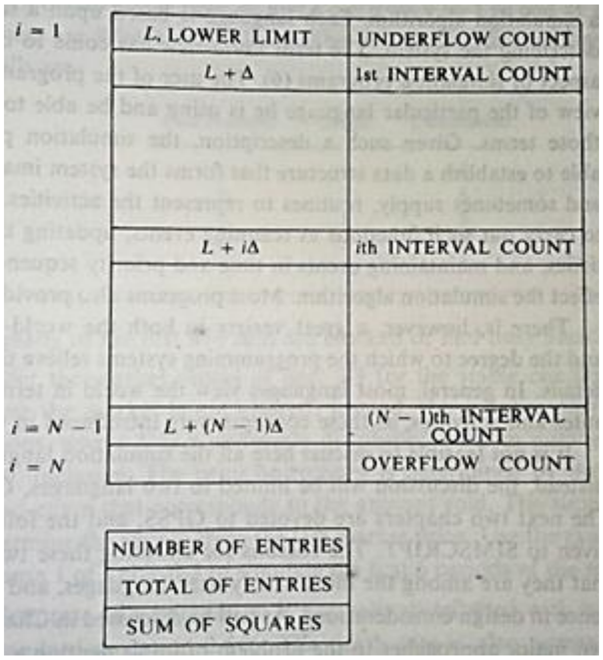
Fig: Definition of a Distribution Table

https://www.ckundan.com.np/2019/06/discrete-system-simulation.html
Generally, it is required to keep count the number of time and observation falls below the lower limit and beyond the upper limit rather than to obtain accurately the potential range of valuation.

To determine the mean and standard deviation, it will be required to accumulate number of observation (X_i) and the sum of squares $\sum(X_i)^2$. For each observation X_i,

- One is added to appropriate counter
- X_i is added to the sum, $\sum(X_i)$
- $(X_i)^2$ is added to the sum, $\sum(X_i)^2$

The space required for the tabulation shown in the figure:



Since values of observation are matched to an interval the derived destination is an approximation. However, note that the mean and standard deviation will be accurate within the accuracy limit of the computer even if some observations fall outside the table limits.

The instances when the observations are made are determined by the native of the random variable being measured. To understand this, consider the following two examples:

1. To measure the mean waiting time for a service, an observation must be taken as each entity starts to receive service so that the times at which the observations are tabulated are randomly spaced.

2. To measure the distribution of the number of entities waiting, observation should be taken at uniform interval time. The final output weight also expresses the data in another commercial form as required. For example, the cumulative distribution may be given or the distribution may be resolved to express the counts as a percentage of total observation. These results are calculated at the end of the simulation.

A clock is used in the manner of the time stamp to measure transit time. When an entity reaches a point from which a measurement of transit time is to start. A note of the time of arrival is made.

Later when the entity reaches the point, at which, the measured ends, a note of the clock time upon arrival is made and these two clock time noted are used to compute the elapsed time.

Discrete Simulation Languages:

To simplify a task of writing a discrete simulation program, several programming languages are sued. The two most commonly used are:
1. GPSS
2. SIMSCRIPT

These languages are used to describe a system and establish a system image and execute a simulation algorithm. Most programming also provides report generators. These languages also offer many convenient facilities such as,
a. Automatic generators of streams of pseudo-random numbers for any desired statistical destination.
b. Automatic data collection
c. Their statistical analysis and report generations
d. Automatic handling of queues, etc.

Also, a good simulation provides a nodded builder with the view of the world that next nodded building easier. Every discrete system simulation language must provide the concept and statement for:
a. Representing the state of a system at a single point in time (static modelling)
b. Moving a system from state to state (Dynamic modelling), &
c. Performing relevant task such as the random number of generation, data analysis and report generation.

Discrete system simulation languages can be classified into three main categories:
a. Event oriented languages
b. Activity-oriented languages
c. Process-oriented languages.