Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

SIMULATION WITH SIMNET II

Hamdy A. Taha

SimTec, Inc.
P.O.Box 3492
Fayetteville, AR 72702

ABSTRACT

SIMNET II is a network-based discrete simulation language that utilizes only four nodes: a source, a queue, a facility, and an auxiliary. Routing of transactions among the four nodes is effected by executing special assignments along the branches linking the nodes. SIMNET II offers powerful computational capabilities at a level equal to FORTRAN, thus eliminating the need for using external FORTRAN or C inserts as in other languages. The special assignments of SIMNET II also allow the modeler to exercise complete control over the internal characteristics of the nodes during execution. The system is totally interactive and is available in totally compatible version for the mainframe, mini, and micro computers.

1 SIMNET II DESIGN APPROACH

The design of SIMNET II is based on the observation that discrete simulation deals primarily with queueing systems. Within this framework, SIMNET II utilizes three suggestive nodes: a **source** from which transactions arrive, a **queue** where transactions may wait, and a **facility** where transactions are served. A fourth node, call **auxiliary**, is added to enhance the modeling capabilities of the language.

Nodes in SIMNET II are linked by **branches** that specify the path to be taken by each transaction. As transactions traverse branches, they execute **special assignments** that control the flow anywhere in the network and also allow the modeler to change the characteristics of the different nodes during execution. The use of special assignments in place of special blocks (as in other languages) is conceptually superior because these assignments are directly amenable for use within SIMNET II conditional IF-THEN-ELSE-ENDIF and loop FOR-NEXT statements.

SIMNET II also allows the use of scarce **resources** that are shared among facilities. Logic **switches** are designed to control the flow of transactions out of queue nodes.

Repetitive segments are modeled in SIMNET II using PROCedures (or **PROC-s**). This capability allows the indexing of all the nodes, resources, and switches of the models. PROCs simplifies the modeling of repetitive segments through the use of a generic code that is manipulated internally by the SIMNET II processor to provide the correct representation of the original system.

SIMNET II allows the inclusion of run-dependent initial data in the model. This capability allows the user to execute different runs with different initial data in a single simulation session.

The SIMNET II system utilizes special READ/WRITE statements that allow the user to link the input and output of the model to formatted and unformatted external files. Such modeling facility allows the model to be totally driven by data stored in external files. This capability is enhanced by the fact that parameters such as the number of parallel servers in a facility, the maximum capacity of a queue, and the initial level of a resource can be declared as variables, which subsequently can be read from an external file.

Figure 1 depicts the symbolic representation of the SIMNET II four nodes. Each node includes a number of compartments that house the information of the nodes. Each information element is placed in one of several positional fields associated with the node statements as follows:

```
node_identifier;field_1;field_2;...;field_m:
```

The node_identifier consists of a user-defined name followed by *S, *Q, *F, *A to designate the type of the node as a source, queue, facility, and auxiliary.

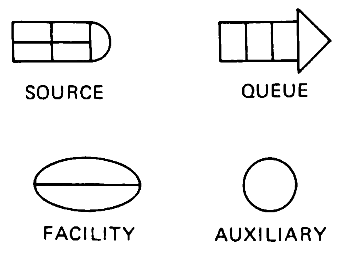

Figure 1

Figure 2 provides the SIMNET II model for a 3-server queueing model. Transactions arrive from source ARVL every EX(5) minutes (exponential with mean 5 minutes). Arriving transactions may wait in queue LINE, if necessary. Service is performed in facility CLRKS where the service time is EX(10) minutes. After the service is completed, the transaction is TERMinated. Termination occurs by using the last fields of facility CLRKS.

```
$PROJECT;node1.dat;2 April 1990;Nancy Sloan:
$DIMENSION;ENTITY(10):
$BEGIN:
   arvl   *S;EX(5):                  !arrivals
   line   *Q:                        !wait in line
   clrks  *F;;EX(10);3;GOTO-TERM:    !served by one
                                     ! of 3 clerks
$END:
$RUN-LENGTH=480:              !run model for 480 minutes
$TRACE=30-35:                 !trace from 30 to 35
$RUNS=1:                      !for one run only
$STOP:
```

Figure 2

2 SIMNET II BRANCHES

SIMNET II uses 7 types of branches to link nodes:

Always (A).
Select (S).
Conditional (C).
Probabilistic (P).
Dependent (D).
Exclusive (E).
Last choice (L).

Each type branch routes transactions among nodes in a specific manner. For example, an A-branch will always link two nodes together, whereas a C-branch will link nodes only when the branch conditions are satisfied.

As transactions traverse a branch, they can execute two types of assignments: arithmetic and special. The first type effects changes in the model's arithmetic variables and the second type performs the important modeling function of controlling the flow of other transactions anywhere in the network.

2.1 Arithmetic Assignments

In SIMNET II, user-defined variables can be nonsubscripted or in the form of single- and double-subscripted arrays. The subscripted variables are dimensioned using the $DIMENSION statement. The nonsubscripted variables are simply defined directly as they are used in the assignments. The following example illustrates the use of both types of variables:

```
$DIMENSION;sample(20),table(10,2):
sum=sum+(sample(J)-1)*table(K+L,J):
```

SIMNET II allows the use of the conditional statement IF-THEN-ELSE-ENDIF and the loop construct FOR-NEXT. The following example illustrates the use of these statements:

```
IF,sum=k,THEN,
   FOR,i=1,TO,n,DO,
      IF,i=3,THEN,LOOP=CONTINUE,ENDIF,
      sample(i)=sample(i)**2,
   NEXT,
ELSE,
   sum=sum+1,
ENDIF
```

The assignment LOOP=CONTINUE is used to skip to the end of the FOR-NEXT loop. A companion assignment LOOP=-BREAK can be used to cancel the remainder of the loop altogether.

2.2 Special Assignments

SIMNET II special assignments assume the simple form A=B, where A represents the result of implementing an action B. These assignments are designed to control the flow of transactions anywhere in the network. Their modeling power is enhanced by the fact that they can be implemented within the context of IF-ENDIF and FOR-NEXT statements. As an illustration, consider the following statement:

```
IF,COUNT(NN)=100,THEN,SUSPEND=SS,ENDIF
```

The statement deals with two nodes named NN and SS, where SS is a source node. The statement stipulates that if 100 transactions have passed through node NN, then source SS must stop creations instantly.

SIMNET II provides special assignments for the following cases:

File manipulations.
Source and queue control.
Attributes control.
Statistical variables collection.
External files READ/WRITE capability.

File manipulation assignments play the crucial role of moving transactions among files (queues and facilities). For example, the assignment I(Q2)=J(Q1) will move the J th ordered transaction from queue Q1 and place it in the I th position in queue Q2. Some of the remaining assignments are described below.

3 CONTROL OF QUEUE NODE PARAMETERS

A powerful feature of SIMNET II is that it allows the modeler to change the basic parameters of the queue node during the course of execution of the SIMNET II model. The SIMNET II model initially specifies the queue's maximum capacity, accumulation condition, and discipline. The information content of these fields may be changed dynamically during the course of the simulation by using the following three special assignments:

CAP(queue name)=expression
ACCUM(queue name)=(expression)(attribute rule)
DISCIPLINE(queue name)=queue discipline

Any SIMNET II mathematical expression may be used where indicated in the assignments.

Related assignments that control the operation of a source include

SUSPEND=source name
RESUME=source name

These two assignments, when executed during the course of the simulation, will suspend and resume creations from a source instantly.

4 READ/WRITE CAPABILITY

SIMNET II allows the modeler to link to external file through the use of the following READ and WRITE special assignments:

```
READ(#,d,p)=("(f)",a1,a2,...,an)
WRITE(#,d,p)=("(f)",a1,a2,...,an)
```

where,

# = FORTRAN unit number in the range (0,5,15-99)
d = disk drive (micro version only)
p = file path (micro version only)
f = optional image FORTRAN real format (F,E, and G descriptors only) ai = i th READ/WRITE element, i = 1, 2, ..., n

An example of the use of the READ statement is

```
READ(22)=("(f6.0,f6.3)",N,SCORE)
```

In this statement the input file is connected to unit 22 and the elements N and SCORE are read according to the format (f6.0,f6.3). It is important to point out that the parameters # and ai of the READ/WRITE statement can be any legitimate mathematical expression. For example, the unit number may be expressed as I+J**2 and ai may be given as SAMPLE(I+2*J) for the READ statement or A(2)**J for the WRITE statement.

The READ (and WRITE) statements may be used any time during the execution of the SIMNET II model. However, the statement offers a special capability that allows the modeler to change the initial parameters of nodes and resources at the start of each run. This means that the model can be driven totally from an external file. Such node parameters include number of parallel servers in a facility, queue capacity, source limits, and resource initial levels.

5 PROCS

PROCedures (or PROCs) in SIMNET II deal with the modeling of repetitive segments in a compact and efficient way. A PROC provides a generic code that is manipulated internally by the processor to provide the correct representation of the simulated system. Actually the use of PROCs is more than just a convenience in producing a code. Situations arise in simulation where it will be next to impossible to represent a system properly without the capability of the PROC. Typical among these situations is job-shop scheduling.

A PROC starts with the statement *PROC(L-U), where L and U are positive integers that define the PROC's range of applicability. Repetitive nodes inside the PROC are represented by a user-selected base-name followed by a blind index(). For example, consider the following statements:

```
        *PROC(1-3):
   SS()  *S;EX(2):
```

The PROC range (1-3) automatically specifies the indices of all its blind indexed nodes. This means that SS() inside the PROC stands as a generic representation of SS(1), SS(2), and SS(3).

A PROC may include nonindexed nodes as well. This means that nodes inside a PROC can be regular or blind-indexed only. Explicitly indexed nodes, such as QQ(1) *Q:, are not allowed in the definition of nodes inside a PROC.

Figure 3 provides an illustration of the use of PROCs in SIMNET II. Transactions leaving a single source SS are assigned to three production lines according to the probabilities .3, .5, and .2. Figure 3 shows that the probabilistic direct transfer is used to route the transaction. Specifically the source statement

```
   SS  *S;EX(1);*QQ(1)/.3,QQ(2)/.5,QQ(3)/.2:
```

will orient created transactions in the desired manner.

```
$PROJECT;proc3.dat;4/8/91;Taha:
$DIMENSION;ENTITY(60):
$BEGIN:
            *PROC(1-3):
     ss     *S;EX(1);*qq(1)/.3,qq(2)/.5,qq(3)/.2:
     qq()   *Q:
     ff()   *F;;EX(2);*TERM:
            *ENDPROC:
$END:
$RUN-LENGTH=100:
$STOP:
```

Figure 3

6 GATHERING STATISTICAL OBSERVATIONS

The nature of the simulation experiment poses statistical difficulties because, by the nature of simulation, successive observations are not independent. The effect of observation dependence is more prominent during the transient or warm-up period of simulation that occurs at the early stages of the simulation.

The adverse effect of observation dependence and transient conditions can be dampened in simulation experiments by following two general rules:

(a) Truncating the transient period.

(b) Defining individual observations as batch averages.

Batch averages can be determined in the simulation experiment using two methods:

(a) The subinterval method.

(b) The replication method.

SIMNET II allows for the interactive estimation of the transient period using interactive graphics. Following the estimation of the transient period, the modeler can instruct SIMNET II to disregard the statistics collected during that period and to collect the final global summary based on either the subinterval or the replication method.

7 SUMMARY

This paper has presented an outline of the SIMNET II simulation language. The four-node structure of the language makes it particularly easy to use. The use of the special assignments together with the conditional and loop constructs gives SIMNET II a superior modeling capability. Also, the fact that all the parameters of the SIMNET II model can be defined from an external file through the use of the READ statement allows greater flexibility in the execution of the simulation model.

REFERENCES

Taha,H.A. (1988), *Simulation Modeling and SIMNET*, Prentice-Hall, Englewood Cliffs, NJ.

Taha,H.A. (1990), *Simulation with SIMNET II*, SimTec, Inc., Fayetteville, Arkansas.

AUTHOR BIOGRAPHY

HAMDY A. TAHA is President of SimTec, Inc. and Professor of Industrial Engineering at the University of Arkansas. He holds a BS degree in Electrical Engineering (Alexandria University, 1958), MS degree in Industrial Engineering (Stanford University, 1961), and Ph.D. degree in Industrial Engineering (Arizona State University, 1964). He is the developer of the SIMNET simulation language and the author of four books in Operations Research and Simulation. His most recent book is *Simulation Modeling and SIMNET*, Prentice-Hall, 1988. His consulting experience is focused on the application of Operations Research and Simulation to the oil industry. He has worked on consulting projects in the U.S., Mexico, and the Middle East.
(501) 575-756-6146 FAX (501) 575-7446
e-mail HT27009@UAFSYSB.uark.edu