Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

SIMNET SIMULATION LANGUAGE

Hamdy A. Taha
Department of Industrial Engineering
EC4207 - Bell Engineering Center
University of Arkansas, Fayetteville, AR 72701

ABSTRACT

SIMNET is a network-based discrete simulation language that differs from available process languages in that it utilizes exactly four nodes: a source for creating transactions, a queue where waiting may take place, a facility where service is performed, and an auxiliary that is introduced to enhance the modeling flexibility of the language. Each node is provided with sufficient information that define the exact manner in which a transaction enters, resides in, and leaves the node. SIMNET does not allow the use of external (FORTRAN) subroutines. Yet, the language is capable of modeling very complex situations rather conveniently.

INTRODUCTION

Imagine the situation in which a transaction leaving a queue must dispose of a previously acquired resource before it enters one of several service facilities. The available process simulation languages model this situation by requiring the transaction to pass through two special serial blocks/nodes: the first block disposes of the resource and the second one selects a desired service facility. This approach necessarily requires that a special block/node be designed to respond to each modeling need. As a result, the indicated design philosophy may be less effective for the following reasons:

1. The user must deal with a large number of blocks/nodes (for example, GPSS has over 60 blocks)

2. The use of special blocks/nodes usually leads to a higher degree of abstractness in model representation.

3. The requirement that transactions pass serially through the special blocks/nodes to perform functions that in reality should occur in parallel reduces the flexibility of the language and hence gives rise to the need for using external subroutines written in a high-level programming language (such as FORTRAN).

4. The fact that each block/node is designed to serve a special modeling function inevitably must lead to a degree of redundancy, and hence inefficiency, in the design of the language.

SIMNET DESIGN FRAMEWORK

SIMNET is a general purpose network-based simulation language in which modeling representation reduces to nodes connected by branches. The language is designed with three fundamental goals in mind:

1. Enhancing the language "user friendliness" by eliminating altogether the use of special-purpose nodes.

2. Achieving modeling flexibility without the need to use external (FORTRAN) subroutines or functions.

3. Integrating the statistical aspects of the simulation experiment directly into the language.

The design of SIMNET recognizes that the majority of discrete simulation situations may be viewed in some form or another as queueing systems. As such, the design of SIMNET is based fundamentally on the use of four nodes: a <u>source</u> from which transactions arrive, a <u>queue</u> where transactions may wait, a <u>facility</u> where service is offered, and an <u>auxiliary</u> which in reality is conceived as an infinite capacity service facility.

To abide by the four-node restriction, which signifies the elimination of all special-purpose nodes, each node is designed to be completely self-contained; in the sense that sufficient information is provided to describe the exact behavior of a transaction as it enters, resides in, and leaves a given node. Such a definition necessitates that a transaction acquire all its "needs" (e.g., resources, next-node selection) in parallel, as compared with the serial mode used in presently-available process languages. The parallel acquisition approach represents a fresh design idea that has proven quite effective in modeling very complex simulations rather conveniently.

LAYOUT OF A SIMNET MODEL

A SIMNET model includes the following four basic segments:

1. Definitions statements:
   $PROJECT;
   $DIMENSION;
   $RESOURCES:
   $SWITCHES:
   $VARIABLES:

2. Model logic statements

3. Control statements:
   $RUNS=
   $OBS/RUN=
   $RUN-LENGTH=
   $TRANSIENT-PERIOD=
   $PLOT=
   $TRACE=

SIMNET is a trademark of SimTec, Inc.

4. Initial Data:
   $INITIAL-ENTRIES:
   $DISCRETE-PDFS:
   $TABLE-LOOKUPS:
   $V-VALUES:
   $W-VALUES:

The $PROJECT statement includes general information about the model and the analyst. $DIMENSION statement is used to allocate storage dynamically to SIMNET's files and arithmetic variables. The statement has the following general format:

$DIMENSION; ENTITY(.),A(.),V(.),W(.,.):

where ENTITY represents the maximum number of entries allowable in all SIMNET's files with A(.) defining the number of user attributes in any of the files. V(.) and W(.,.) represent SIMNET's single and double-subscripted arithmetic variables.

SIMNET defines the model's resources by using $RESOURCES statement. $SWITCHES are used to control flow out of nodes depending on whether the switch status is ON or OFF. The model's statistical variables, which may be OBS.BASED (observation-based), TIME.BASED (time-persistent), or RUN.END, are defined by the $VARIABLES statement. In all cases, resources, switches and variables are assigned user-defined names for use in the model.

The second segment includes the logic of the model that describes the flow of transactions among the model's nodes. Control statements (third segment) provide data regarding number of independent $RUNS, number of $OBS/RUN, $RUN-LENGTH, $TRANSIENT-PERIOD length, $PLOT of model's variables, resources level, and/or files length, and $TRACE of model calculations.

The fourth segment provides the initial data of the model including initial file entries, discrete probability density functions, table lookup functions, and the initial values of arithmetic V and W variables. These data may be associated with different runs to allow the execution of independent runs with different initial data in the same simulation session.

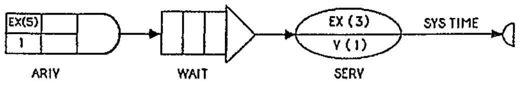
Example. Figure 1 provides a multiple-server SIMNET model in which it is desired to compare the effect of using two and four parallel servers [V(1)=2 or 4] on the period of time a customer spends in the system (SYS TIME). The variable SYS TIME is an OBS.BASED variable whose value is defined as TRANSIT(1), the difference between current simulation time and the creation time of the transaction which is stored in attribute A(1).

Transactions are created at source ARIV. The intercreation time is EXponential with mean 5 units as given in field 1. Field 2, which represents the occurrence time of the first creation, is defaulted to zero. Field 3 defines A(1) as the attributes to carry the creation time of the transaction.

When a transaction leaves ARIV, it will either join the queue named WAIT or enter facility SERV, depending on the status of the facility. The service time at SERV is EX(3). The number of parallel servers is V(1), which may assume a value of 2 or 4 as defined by Initial Data. When service is completed, the transaction passes through a branch (*B) to compute SYS TIME before being TERMinated.

The Control statements show that the $RUN-LENGTH is 1000 time units. The simulation session includes two runs ($RUNS=2) corresponding to V(1)=2 and V(1)=4, respectively. The $TRACE statement indicates that the simulation computations will be detailed between time 20 and time 60.

The Initial Data statement specify V(1)=2 for run 1 and V(1)=4 for run 2.



```
!-------- Definitions:
  $PROJECT;M/M/S Model;15/2/87;Taha
  $DIMENSION; ENTITY(30),A(1),V(1):
  $VARIABLES: SYS TIME;OBS.BASED:TRANSIT(1):

!-------- Model Logic:
  $BEGIN:
          ARIV  *S;EX(5);;1:
          WAIT  *Q:
          SERV  *F;;EX(3);V(1):
                *B;TERM;/4/SYS TIME%:
  $END:

!------ Control:
  $RUN-LENGTH=1000:
  $RUNS=2:
  $TRACE=20-60:

!------ Initial Data:
  $V-VALUES:  1-1/NS/2:     !V(1)=2 in run 1
              2-2/NS/4:     !V(1)=4 in run 2
  $STOP:
```

Figure 1: SIMNET Multiple Server Model

SIMNET NODES

A node in SIMNET is described by an identifier followed by a specific number of fields as follows:

Node identifier; field 1; field 2, ...; field m:

A semicolon is used to separate the fields with a colon signifying the termination of the statement for the node.

A node identifier consists of an arbitrary user-defined name followed by one of the symbols *S, *Q, *F, or *A that describes the node as a source, queue, facility, or auxiliary, respectively. Figure 2 summarizes the representation of SIMNET's nodes.

To illustrate how the special-purpose nodes are eliminated by applying the concept of parallel acquisition of "needs" to each of SIMNET's four nodes, consider the situation in Figure 3. Here, the facility named LDRS represents two mechanical loaders used to load two trucks named TRKS. Orders arrive from two queues Q1 and Q2. When an order arrives at LDRS, it must await the arrival of a truck before it can be filled. A truck, when available, will take 11 minutes to arrive and 45 minutes to deliver the load, after which time it becomes available again for taking a new

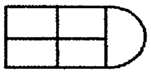
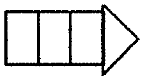
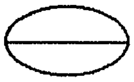
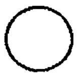
NODE	SYMBOL	IDENTIFIER		FIELDS
Source		Sname	*S	8
Queue		Qname	*Q	6
Facility		Fname	*F	6
Auxiliary		Aname	*A	4

Figure 2: SIMNET Nodes

load. Following the loading of a truck, the order is sent to one of two clerks (queues Q3 and Q4) for completion of paper work and billing.

Trucks TRKS are represented in the model as a resource to be used by the loading facility LDRS. The statements representing LDRS thus appears in SIMNET as follows:

```
LDRS *F;HBC(Q1,Q2);EX(20);2;LBC(Q3,Q4);TRKS(1,11,1,45):
          (1)      (2) (3)  (4)            (5)
```

Field 1 of the statement indicates that LDRS will choose transactions from Q1 and Q2 using the selection rule HBC (Highest Busy Capacity); that is, the longest queue. Field 2 of LDRS defines the service time as EX(20). The two parallel loaders of LDRS are defined in field 3. Field 4 indicates that the paper work of completed orders will be forward to either queue Q3 or Q4 according to the selection rule LBC (Lowest Busy Capacity). In field 5, the statement specifies how the resource TRKS is utilized by LDRS. The four elements (1,11,1,45) indicate that LDRS requests one truck which, if available, will take 11 minutes to arrive. After the service is completed, the truck is returned back to "base stock" after a delay of 45 minutes.

The important observation about the SIMNET statement of facility LDRS is that it defines all the information needed to decide how a transaction enters, leaves and resides in the facility. If special-purpose nodes were to be utilized, as in other languages, at least five such nodes will be needed to effect an equal representation.

SIMNET FILES

Both queues and facilities are treated as (user-defined) files that are automatically maintained by SIMNET. A queue may have a finite or infinite capacity. A facility capacity is necessarily finite and is taken to represent the number of parallel servers. The LENgth of a facility file thus represents the number of busy servers whereas the LENgth of a queue is taken to represent the number of waiting transactions. SIMNET also utilizes an internal event calendar file (E.FILE) that is used to advance the simulation clock. All SIMNET's files operate within the space reserved by ENTITY in the $DEMENSION statement. The number of attributes associated with each file entry is specified by the vector A(.) in the same statement.

SIMNET BRANCHES

As transactions traverse a branch joining two nodes they perform the following important functions:

1. Execute of arithmetic assignments.
2. Manipulate files (insertion, deletion, reordering, replacing, and swapping).
3. Resume and/or suspend creations from sources.
4. Cause movements, cessation and/or destruction of transactions anywhere in the network.
5. Collect statistical observations.
6. Return resources to "base stock."
7. Stop simulation run, if desired.

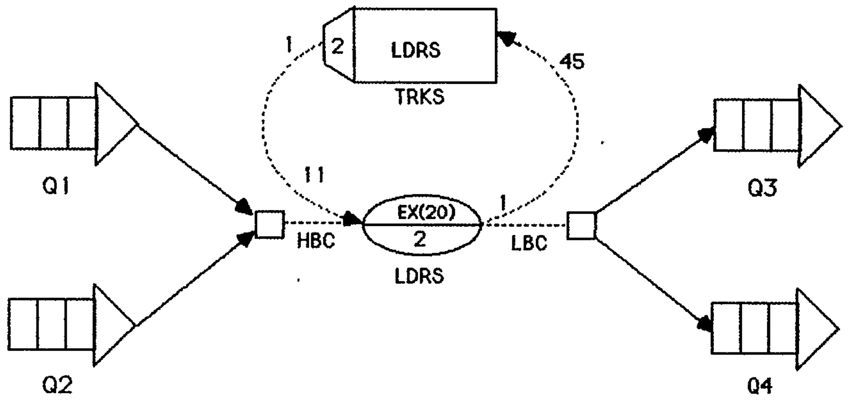


Figure 3: A Segment of a SIMNET Model

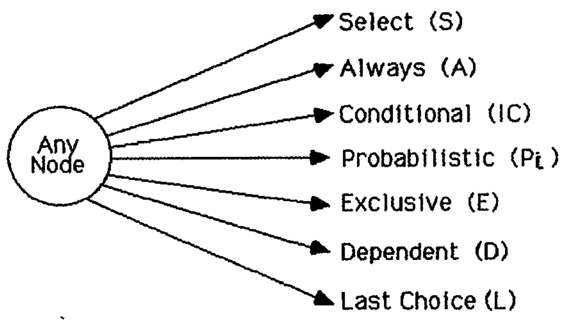
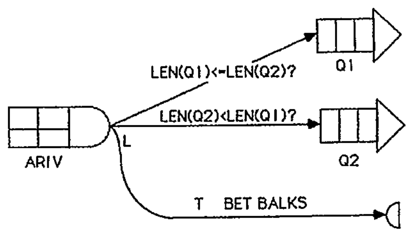


Figure 4: SIMNET Branches

SIMNET provides seven types of branches as summarized in Figure 4. Any or all of these branches may emnate from any node, with each branch acting independently depending on its type. Thus, a last-choice (L) branch is taken only if none of the other branches at the node are taken. A dependent (D) branch is taken only if at least one of the other branches (excluding the L-branch) is taken. An exclusive (E) branch is similar to the D-branch but differs in that it will block all other branches if it is impassable. Conditional (IC) branches with a given integer value assigned to IC indicate that they are controlled by conditions and that the first satisfied IC branches, if any, will be taken. Probabilistic branches are chosen according to predetermined probabilities. An always (A) branch will be permanently accessable. Finally, a select (S) branch will be associated with the "select" conditions of the node, if any.

Figure 5 illustrates the use of branches in SIMNET. A transaction leaving source ARIV will enter queue Q1 if its length does not exceed that of Q2; otherwise, it will enter Q2. If both queues are full, the transaction is terminated after computing the time between balks (T BET BALKS). The first two branches are conditional with IC=1, which signifies that only one of these branches may be taken. If neither branch can be taken, the transaction is TERMinated as a last choice.



```
ARIV  *S;EX(1):
      *B;Q1/1;LEN(Q1)<=LEN(Q2)?:
      *B;Q2/1;LEN(Q2)>LEN(Q1)?:
      *B;TERM/L;/4/T BET BALKS%:
```

Figure 5: Example of SIMNET Branching

SIMNET LOGIC SWITCHES

A logic switch is defined by using the $SWITCHES statement. Switches are used to control flow in the network by using "remote control" effect. To understand the significance of this point, consider the logic switches GATE and LOCK defined as follows:

```
$SWITCHES: GATE;ON;Q1,Q2:
           LOCK;OFF:
```

Switch GATE is initially ON and it operates on the queues named Q1 and Q2. Switch LOCK, on the other hand, is initially OFF and it controls no queues.

The status of a switch can be changed by using assignments of the type

switch name = ON or OFF

that are executed as transactions traverse branches. We can thus control flow from nodes by using conditional branches that check the status of the switch. For example, if we execute the assignment GATE=OFF and the branch out of queue Q1 carries the condition GATE=ON?, then no transactions will be allowed to leave Q1.

Switches play an important role in controlling flow out of queues. Specifically, when the definition of a switch includes reference to specific queues (e.g., GATE controls Q1 and Q2), an attempt will be made to move transactions out of these queues each time the switch status is set to ON. For example, the execution of the assignment GATE=ON anywhere in the network will automatically cause movement of transactions out of Q1 and Q2, if possible. This type of action provides the so-called remote control in SIMNET networks.

SIMNET RESOURCES

A resource may be acquired only at a facility. However, it may be disposed of any where in the network, including the exit end of a node or the terminal end of a branch. SIMNET defines each resource with preset priority classes among various facilities. Moreover, the definition allows higher priority facilities to preempt any or all of the lower priority facilities. The following example illustrates the definition of a resource:

```
$RESOURCES: R1;3(F1/F2(NPR)/F3,F4):
```

Resource R1, which has an initial level of three units, may be acquired by any of the four facilities F1, F2, F3, and F4. Facility F1 has the highest priority, whereas F3 and F4 have the lowest. By default, F1 may preempt F2, F3, and/or F4. F2, on the other hand, uses the symbol (NPR) to indicate that, though of higher priority, it cannot preempt F3 or F4. Finally, F3 and F4 occur on the same priority level.

When resource R1 becomes available, its associated facilities will be scanned in order of priority to see if any of them is awaiting the availability of the resource. If the resource is not needed, it will simply wait in "base stock." On the other hand, when a facility requests a resource, it will first attempt the base stock, and if not available will preempt lower priority facilities, if possible.

Resources are acquired and disposed of by utilizing the appropriate field of a node or a branch. To illustrate how the resource field is utilized, suppose that a resource RR is defined as follows:

RESOURCES: RR;5(F1/F2):

Figure 6 shows the use of the resource by facilities F1 and F2. The resource field of F1 defined as RR(3,1,2,0) indicates that F1 acquires three units of RR, which arrive after 1 time unit. When F1 completes service, it will instantly return two units of RR to "base stock." Facility F2 instantly acquires one unit of the resource and returns none. This is achieved by specifying the resource field for F2 as RR(1,0,0,0). The resource is eventually returned by the branch by defining the branch resource field as RR(0,0,1,0), which shows that the branch will return one unit of the resource.

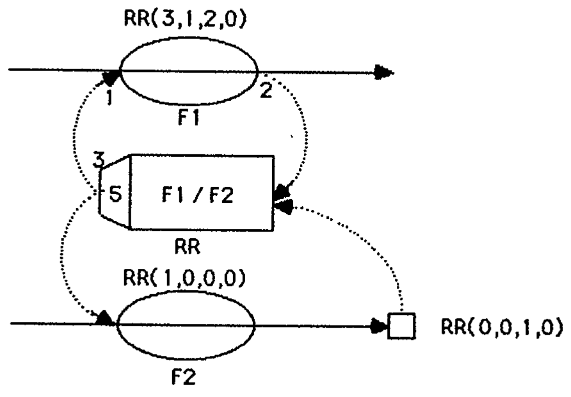


Figure 6: Example of Use of Resource

SIMNET ASSIGNMENTS

Assignments are executed in SIMNET as transactions traverse branches. SIMNET offers seven types of assignments:

1. Arithmetic
2. File manipulation
3. Switch operation
4. Source control
5. Statistics collection
6. Forced simulation stop
7. Conditional assignment

The arithmetic statement involves defined SIMNET variables. File manipulations allow swapping, deleting, reordering, and locating entries in files. These assignments are designed to allow complete control over any transaction in any file. For example, 1(Q1)=LAST(Q2) will remove the last entry in Q2 and place first in Q1. Also, COPY=I(Q) will copy the attributes of entry I of file Q into the attributes buffer A(.). On the other hand, J=LOC(Q/1<100), will locate the entry in file Q whose A(1) is less than 100, and then assign the rank of that entry to variable J. As a further illustration, 3(Q)=DEL will DELete the third entry of file Q.

Switch operation assignments allows the modeler to change the status of a switch to ON or OFF. Source control assignments assume the format:

```
SUSPEND = source name
RESUME  = source name
```

The first assignment prevents a source from creating new transactions, whereas the second assignment reverses the action by allowing the source to resume creations again.

Statistics can be collected either by listing the variable name in the appropriate branch field or by using the assignment

COLLECT = Variable name

Finally, a simulation run may be stopped from any branch in the network by using the special assignment SIM=STOP.

The power of the assignments above is further enhanced by using the conditional assignment of the form

```
IF, conditions,
  THEN, assignments,
      ELSE, assignments,
END:
```

The following SIMNET segment illustrates the use of the conditional assignment:

```
$VARIABLES: JOB LATENESS;OBS.BASED; V(1):
            JOB PROMPTNESS;OBS.BASED; V(1):
   .
   .
   .
IF, V(1)>0,
  THEN, COLLECT=JOB LATENESS,
      ELSE,COLLECT=JOB PROMPTNESS,
ENDIF
```

INDEXING AND PROCS IN SIMNET

Indexing is introduced in SIMNET to allow the representation of repetitive segments of a model in a compact and efficient manner. The following example is designed to demonstrate the use of this concept in SIMNET.

Bank Model Example. Consider the situation of a two-window drive-in bank. Each lane can accommodate a maximum of three cars excluding the car being served. An arriving customer will always choose the shorter lane with preference given to the right lane in case of a tie. Customers that cannot join balk to seek service elsewhere. If at any time, the length of one lane becomes at least two cars shorter than the other, the last car in the longer queue will jockey to the shorter lane. It is desired to estimate the time a customer spends in each lane before completing its service as well as the time interval between balks.

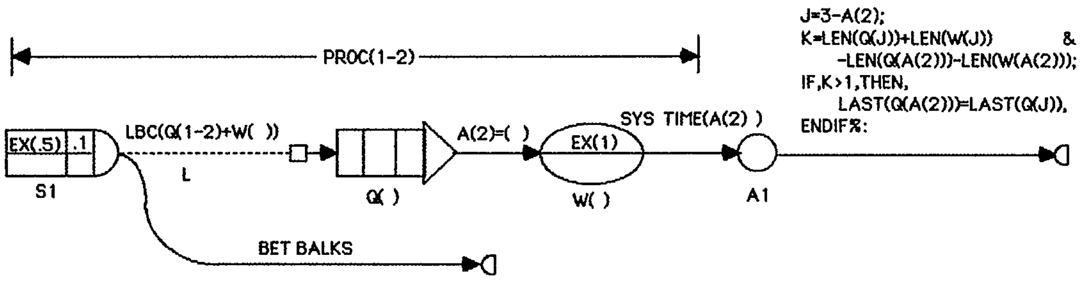


Figure 7: SIMNET Network of the Bank Model

Figures 7 and 8 provide the network and the model statements. Instead of modeling two lanes explicitly, we use a PROCedure with indexed names to represent the two lanes. First note that the variable SYS TIME is now indexed from 1 to 2, which is equivalent to defining SYS TIME(1) and SYS TIME(2) associated with lanes 1 and 2, respectively.

The main model includes a PROC that starts with the identifying statement *PROC(1-2) and terminates with *ENDPROC. The range (1-2) indicates that the PROC corresponds to two (parallel) lanes. The first node in the PROC is a source named S1 that generates arriving customers every EX(.5) minutes, with the first customer arriving at T=.1 minute. A(1) is used to mark the creation time of transaction. In Field 6 of S1 the code LBC(Q(1-2)+W()) is a select rule that specifies that either Q(1) or Q(2) will be entered depending on the combined lengths of Q(1) and W(1) relative to those of Q(2) and W(2). The rule LBC stands for Lowest Busy Capacity. We note that Q(1) or Q(2) will be taken only if the selected queue is not full. Otherwise, the transaction will be terminated after computing BET BALKS statistical variable.

The PROC represents the two parallel lanes by using the "blind" indexed names Q() and W() to represent the queue-window combination of each lane. Queues Q() each has a capacity of 3 cars. At facilities W(), the service time is given as a NOrmal distribution with mean and standard deviation of 1 and .2 minute, respectively. When Q() is exited, the "blind" assignment A(2)=() sets A(2) equal to 1 or 2 depending on whether the transaction has come from Q(1) or Q(2). Upon terminating service in W(), the tranaction enters auxiliary A1 outside the PROC boundaries. The branch from W() to node A1 computes SYS TIME(A(2)), where A(2) = 1 or 2.

Jockeying between lanes can occur only after a transaction leaves auxiliarly A1. The branch from A1 defines SIMNET's variable J=3-A(2) as the complement of the exited lane A(2). The variable

K=LEN(Q(J))+LEN(W(J))-LEN(Q(A(2)))-LEN(W(A(2)))

determines the difference between the lengths of the two lanes. If K>1, the last car in Q(J) is jockeyed to the last position in Q(A(2)). This result is achieved by using the IF-THEN-ENDIF conditional assignment.

```
----------------------------------------------
     S I M N E T /PC (TM) Rel 1.0
     Copyright (c) 1987 by Hamdy A. Taha
----------------------------------------------
$PROJECT;BANK MODEL;19 SPT 84;TAHA:
$DIMENSION;ENTITY(30),A(2):
!BANK MODEL USING PROC, Pritsker p. 310
$VARIABLES: BET BALKS;OBS.BASED;BET.ARVL:
            SYS TIME(1-2);;TRANSIT(1):
$BEGIN:

        *PROC(1-2):
  S1    *S;EX(.5);.1;1;/6/LBC(Q(1-2)+W()):
        *B;TERM/L;/4/BET BALKS%:
  Q()   *Q;3:
        *B;W();;A(2)=()%:
  W()   *F;;NO(1,.2):
        *B;A1;/4/SYS TIME(A(2))%:
        *ENDPROC:

  A1    *A:
        *B;TERM;;J=3-A(2);
                 K=LEN(Q(J))+LEN(W(J))     &
                 -LEN(Q(A(2)))-LEN(W(A(2)));
                 IF,K>1,THEN,
                   LAST(Q(A(2)))=LAST(Q(J)),
                 ENDIF%:
$END:
$RUN-LENGTH=480:  $OBS/RUN=5:
$STOP:
----------------------------------------------
```

Figure 8: SIMNET Statements of the Bank Model

The power of the conditional IF-THEN-ENDIF assignment stems from the fact that file manipulation can be executed within the assignment. Notice also that the full conditional assignment has the format IF-THEN-ELSE-ENDIF, which makes it even more powerful in general. This use of conditional assignment is unique to SIMNET as none of the available network and block oriented languages provide this option.

SIMNET'S DEBUGGING FACITLITIES

SIMNET provides extensive syntax and execution errors checking. All messages appear with explicit explanatory notes for the cause and location of error. In addition to these error messages, SIMNET produces a like-English trace report that accounts for all the details of simulation computations. This unique report can be obtained simply by invoking the control statement $TRACE=T1-T2, where T1 and T2 represent the time limits of the simulated period.

Figure 9 provides a segment of the $TRACE report for the Bank model. The report automatically keeps track of all the attributes of the transactions as they move about the network. It also prints the observation values of the statistical variables SYS TIME(1) and SYSTIME(2) at the moment these values are computed. The results of execution of assignments are provided as they occur during the course of the simulation. Note, in particular, how the trace report provides the details of the file manipulation assignment LAST(Q(A2))=LAST(Q(J)) at T=3.305. It shows that entry 1 is unlinked from Q(1) for linking with Q(2). However, since facility W(2) is found empty, Q(2) is skipped and service is started in W(2). Meanwhile, the report automatically updates the current length of queues and the utilization of facilities. With such detailed information, the user should be able to check the logic of the simulation model rather conveniently.

```
----------------------------------------------------------------
 0.3018E+01   EXIT 'S1    **'
                ATR(S): 0.3018E+01   0.0000E+00
              FILE NEXT 'S1    **' CREATION AT T = 0.4812E+01
                ATR(S): 0.4812E+01   0.0000E+00
              ENTER 'Q     1' -- CUR.LEN =   1
                ATR(S): 0.3018E+01   0.0000E+00

 0.3305E+01   EXIT 'W     2'
                ATR(S): 0.2166E+01   0.2000E+01
              'SYS TIME   2' VALUE =0.1139E+01
              FILE DEPARTURE FROM 'A1    **' AT T = 0.3305E+01
                ATR(S): 0.2166E+01   0.2000E+01
              MAKE SERVER IN 'W     2' IDLE -- CUR.UTILIZ =   0

 0.3305E+01   EXIT 'A1    **'
                ATR(S): 0.2166E+01   0.2000E+01
              J = 0.1000E+01
              K = 0.2000E+01
              UNLINK ENTRY   1 FROM 'Q     1' --CUR.LEN =   0
                ATR(S): 0.3018E+01   0.0000E+00
              SKIP 'Q     2' -- CUR.LEN =   0
                ATR(S): 0.3018E+01   0.0000E+00
              A(   2) = 0.2000E+01
              ENTER 'W     2' -- CUR.UTILIZ =   1
                ATO(S): 0.3018E+01   0.2000E+01
              FILE DEPARTURE FROM 'W     2' AT T = 0.4628E+01
                ATR(S): 0.3018E+01   0.2000E+01
              TERMINATE TRANSACTICN
----------------------------------------------------------------
```

Figure 9: Example of $TRACE Report of Bank Model

The utility of the trace report is further enhanced by allowing the user to dump any of the files simply by using an assignment of the form DUMP = (file name) at desired locations in the network. This assignment will be executed only if $TRACE is on.

COLLECTING STATISTICS IN SIMNET

SIMNET provides a standard global statistical summary based on either the subinterval or the replication method. These results are automatically obtained through proper specifications of the control statements $RUNS and $OBS/RUN. If $RUNS equals 1 and $OBS/RUN is greater than one, SIMNET collects the data based on the subinterval method. On the other hand, if $RUNS is greater than one and $OBS/RUN equals 1, the replication method is automatically assumed. These are the only two combinations allowed.

Notice, however, that in the case of the replication method, initial data for all the runs are required to be identical. When the SIMNET detects the use of different initial data for the different runs, the situation will be treated as independent runs with no global statistics computed.

SIMNET allows for the truncation of the initial warmup period of a run through the use of the control statement $TRANSIENT-PERIOD = T1. If T1 is greater than zero, no statistics will be collected during the first T1 time units of the simulation run.

```
SCALES:
S=SYS TIME 1  0.0000E+00                 0.1950E+01                0.3900E+01
Q=Q        1  0.0000E+00                 0.7500E+00                0.1500E+01
*=STD-Q    1  0.0000E+00                 0.1500E+01                0.3000E+01

              0%        20%       40%       60%       80%       100%
    TIME      +....+....+....+....+....+....+....+....+....+....+
    0.2000E+02 .      *Q         S      .                        .
    0.4000E+02 .               *        S Q                      .
    0.6000E+02 .              *       S .Q                       .
    0.8000E+02 .                *       .  S        Q            .
    0.1000E+03 .                *       .S     Q                 .
    0.1200E+03 .               *        S    Q                   .
    0.1400E+03 .                *       . S      Q               .
    0.1600E+03 .                *       . S        Q             .
    0.1800E+03 .                *       . S        Q             .
    0.2000E+03 .                *       . S         Q            .
    0.2200E+03 .                *       . S          Q           .
    0.2400E+03 .                *       .  S            Q        .
    0.2600E+03 .                *       .  S            Q        .
    0.2800E+03 .                *       .  S            Q        .
    0.3000E+03 .                *       .   S             Q      .
    0.3200E+03 .                *       .  S            Q        .
    0.3400E+03 .                *       .  S           Q         .
    0.3600E+03 .                *       . S          Q           .
    0.3800E+03 .                 *      .  S           Q         .
    0.4000E+03 .                 *      .  S            Q        .
    0.4200E+03 .                 *      .  S            Q        .
    0.4400E+03 .                 *      .   S             Q      .
    0.4600E+03 .                 *      .   S             Q      .
    0.4800E+03 .                 *      .    S              Q    .
    0.5000E+03 .                 *      .    S               Q   .
    TIME      +....+....+....+....+....+....+....+....+....+....+
              0%        20%       40%       60%       80%       100%
```

Figure 10: Use of $PLOT with the Bank Model

To assist the modeler in estimating the length of the $TRANSIENT-PERIOD, a $PLOT control statement is used to graph the variation of any of the model's files, variables and resources together with their standard deviations as a function of simulation time. As an illustration, Figure 10 shows the result of using the statement with the Bank Model.

Figure 11 provides a sample output of the bank model based on the subinterval method. This is acheived by specifying $RUNS=1 (by default) and $OBS/RUN =5 to represent the nubmer of subintervals (observations). The output automatically presents a 95% confidence interval for each variable in the output report. It also gives a complete transactions count for the entire simulation run. This count should prove useful in checking the logic of the model as well as in calculating rates of flow at any point in the network.

```
                                    **************************************
                                    *                                    *
                                    *    S I M N E T  OUTPUT REPORT      *
                                    *                                    *
                                    **************************************

PROJECT:     BANK MODEL       RUN LENGTH =  .4800E+03       NBR RUNS =   1
DATE:     19 SPT 84           TRANSIENT PERIOD =  .0000E+00 OBS/RUN =   5
ANALYST:     TAHA             TIME BASE/OBS =  .9600E+02

                         ***  G L O B A L   S T A T I S T I C A L  S U M M A R Y   ***
                                   (SUBINTERVAL METHOD - NBR OF OBS =  5)

                                          --------------------
                                             Q U E U E S
                                          --------------------
            CAPA-   IN/OUT  MEAN/S.D.     MIN/MAX/LAST  MEAN/S.D.   MEAN/S.D.   PERCENT NO-WAIT
            CITY    RATIO   LENGTH        LENGTH        DELAY       POS DELAY   TRANSACTIONS
Q   1         3      1/ 1    .1634E+01     0/  3/  3     .1300E+01   .1365E+01   .5000E+01
                             .3324E+00                   .2755E+00   .6652E-01
             95% LOWER CL =  .1221E+01                   .9578E+00   .1282E+01
             95% UPPER CL =  .2047E+01                   .1642E+01   .1447E+01

Q   2         3      1/ 1    .1436E+01     0/  3/  2     .1463E+01   .1614E+01    .1200E+02
                             .2891E+00                   .2425E+00   .3462E-01
             95% LOWER CL =  .1077E+01                   .1162E+01   .1571E+01
             95% UPPER CL =  .1795E+01                   .1764E+01   .1657E+01
                                          --------------------
                                           F A C I L I T I E S
                                          --------------------
            NBR     MIN/MAX/LAST  MEAN/S.D.     MEAN/S.D.    MEAN/S.D.    MEAN/S.D.    MEAN/S.D.
            SRVRS   UTILIZATION   UTILIZATION   BLOCKAGE     BLKGE TIME   IDLE TIME    BUSY TIME
W   1         1       0/  1/  1    .9635E+00     .0000E+00    .0000E+00    .5809E+00    .1991E+02
                                   .2726E-01     .0000E+00    .0000E+00    .1597E+00    .1079E+02
                    95% LOWER CL = .9296E+00     .0000E+00    .0000E+00    .3826E+00    .6518E+01
                    95% UPPER CL = .9973E+00     .0000E+00    .0000E+00    .7791E+00    .3330E+02

W   2         1       0/  1/  1    .9518E+00     .0000E+00    .0000E+00    .5295E+00    .1275E+02
                                   .2951E-01     .0000E+00    .0000E+00    .1653E+00    .6971E+01
                    95% LOWER CL = .9152E+00     .0000E+00    .0000E+00    .3243E+00    .4096E+01
                    95% UPPER CL = .9885E+00     .0000E+00    .0000E+00    .7347E+00    .2141E+02
                                          --------------------
                                            V A R I A B L E S
                                          --------------------
            GLOBAL MEAN    GLOBAL S.D.    GLOBAL MIN    GLOBAL MAX    95% LOWER CL  95% UPPER CL
BET BALKS    .1070E+02      .8924E+01      .4517E-02     .7221E+02    -.3785E+00     .2178E+02
SYS TIME 1   .2580E+01      .3136E+00      .5490E+00     .4906E+01     .2191E+01     .2969E+01
SYS TIME 2   .2571E+01      .2037E+00      .6192E+00     .4504E+01     .2318E+01     .2824E+01


*** TRANSACTIONS COUNT AT T = .4800E+03 OF RUN  1:
NODE        IN        OUT       RESIDING    SKIPPING    UNLINKED/LINKED    TERMINATED
                                            (BLOCKED)   (DESTROYED)
*S:
 S1                   1007                              (     0)               75
*Q:
 Q    1     566        439         3           30         128/    4             0
 Q    2     309        413         2           45           4/  110             0
*F:
 W    1     469        468         1        (     0)    (     0)                0
 W    2     458        457         1        (     0)    (     0)                0
*A:
 A1         925        925         0                    (     0)              925


E N D   O F   S I M U L A T I O N   S E S S I O N
-------------------------------------------------
```

Figure 11: SIMNET Output report for the Bank Model

CONCLUSION

Experience over the past four years shows that SIMNET is capable of developing general models for very difficult simulation projects. Because SIMNET provides powerful file manipulation assignments, these projects have been modeled rather conveniently without the need for using FORTRAN inserts as in other languages. The fact that SIMNET deals with four nodes only together with its unique trace report makes the language extemely user-friendly.

REFERENCES

1. Taha, H., Simulation Modeling and SIMNET, Prentice-Hall, Englewood Cliffs, N.J., 1988.

Hamdy A. Taha is a professor of Industrial Engineering at the University of Arkansas. His BSEE degree was obtained at Alexandria University, Egypt, his MSIE degree at Stanford University, and his Ph.D. at Arizona State University. He is the author of Simulation Modeling and SIMNET (Prentice-Hall, 1988), Operations Research: An Introduction, 4th ed (Macmillan, 1987), and Integer Programming: Theory, Application, and Computations, (Academic Press, 1975). Taha has extensive consulting experience in industry and government in the United States, Latin America, and the Middle East. His current research interests include the design and implementation of Simulation languages.