Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

SIMULATION AND ANIMATION WITH SIMNET II AND ISES

Hamdy A. Taha
Department of Industrial Engineering
4207 Bell Engineering Center
University of Arkansas
Fayetteville, Arkansas 72701

R. B. Taylor
Department of Industrial Engineering
118 Covell Hall
Oregon State University
Corvallis, Oregon 97331

Nazar A. Younis
Department of Industrial Engineering
1983 East 24th St.
Cleveland State University
Cleveland, Ohio 44115

ABSTRACT

SIMNET II is a network-based discrete simulation language that utilizes only four nodes: a source, a queue, a facility, and an auxiliary. Special routing of transactions among the four nodes is affected by using seven types of branches and the so-called special assignments. SIMNET II offers flexible computational capabilities at a level equal to FORTRAN with access to all internal simulation data and files. The computational and modeling power of the language eliminates the need for the use of external FORTRAN or C inserts. The companion system ISES combines input, no-programming animation, debugging, and execution in a user-friendly interactive environment.

1. INTRODUCTION

SIMNET II is an enhanced version of the discrete simulation language SIMNET [Taha, 1988]. The new version adds important features that greatly enhance the modeling power of the language. Additionally, all the syntax restrictions regarding the names of the arithmetic variables, the structure of function arguments and arrays-subscripts have been removed. The new enhancement eliminates the need to use external FORTRAN or C subroutines which are used to complement the modeling logic of other languages.

Some significant features of SIMNET II include the following:

1. SIMNET II is totally portable among the mainframe, mini, and micro computer versions.
2. SIMNET II uses four (suggestive) nodes only: a source, a queue, a facility, and an auxiliary, which describe the main operations of the queueing orientation of (most) discrete simulation systems. All the remaining modeling functions, which are accomplished in other languages through the use of special blocks or nodes, are accounted for in SIMNET II by using special assignments. This innovative design approach allows these assignments to be used within the context of IF-THEN-ENDIF and WHEN-DO-ENDWHEN, which greatly enhances their modeling power.
3. SIMNET II is totally interactive both during debugging and execution and for all versions on the mainframe, mini and micro computer.
4. SIMNET II utilizes user-specified descriptive names for the attributes and arithmetic variables, which increases the readability of the model.
5. SIMNET II uses PROCs to model complex repetitive segments conveniently.
6. SIMNET II has a built-in experimental frame that allows the user to make runs with different initial data in a single simulation session.
7. SIMNET II offers the unique feature of allowing the estimation of the transient period through interactive graphics, following which the user can delete the transient period and then specify the subinterval or the replication method as a basis for collecting global statistics, all without leaving the interactive mode of execution.
8. The ISES animation system is a total environment that includes the input, execution, animation, and output phases of SIMNET II. ISES creates animation models without any programming effort on the part of the user.

The remainder of the paper is devoted to presenting the highlights of SIMNET II and its companion environment ISES.

2. DESIGN APPROACH OF SIMNET II

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 serviced. A fourth node, called *auxiliary*, is added to enhance the modeling capabilities of the language.

Nodes in SIMNET II are linked by different types of *branches* that specify the path to be taken by each transaction. As transactions traverse branches, they can execute *special assignments* that control the flow of transactions anywhere in the network. Equivalent modeling functions are achieved in other languages by the use of special blocks or nodes. The use of special assignments to replace the special blocks is conceptually superior because these assignments are directly amenable for use within the conditional IF-THEN-ENDIF statements, which greatly enhances their modeling power. Another advantage of the use of the special assignments is that the modeling capabilities can be extended simply by adding new assignments, which will maintain the basic four-node structure of the language unchanged.

3. SIMNET II NODES

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. This information includes two distinct types of data:

1. Those pertaining to the internal operation of the node, e.g., service time in a facility or maximum queue size.
2. Those related to the use of resources in the model and the "select" routing of transactions among nodes.

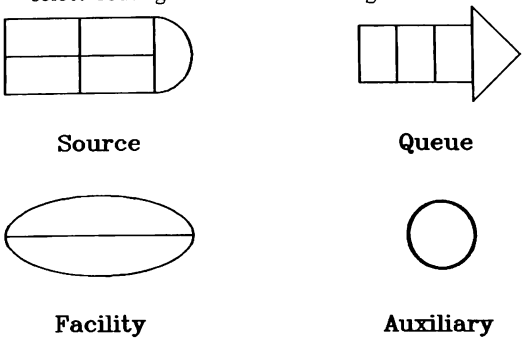


Figure 1. SIMNET II Nodes

Each of these data elements is placed in one of several positional fields associated with the node statements. Specifically, a node statement is given as follows:

node identifier; field 1, field 2,....,field m; * T:

The node identifier consists of a user-defined name followed by *S, *Q, *F, or *A to indicate the type of the node as a source, queue, facility, or auxiliary. Each of the fields is then used to represent a specific data element of the node. The last field (*T) is used to route the transaction to succeeding nodes. Figure 2 gives the SIMNET II model for the multiple-server queueing model. Transactions arrive from source ARIV every EX(5) minutes (exponential with mean 5 minutes). Arriving transactions may wait in queue LINE if necessary. Service is performed in facility SRVR where the service time is EX(4) minutes. After the service is completed, the transaction is TERMinated. Termination occurs by using the *T field of facility SRVR.

```
$BEGIN:
   ARIV        *S; EX(5)
   LINE        *Q:
   SRVR        *F;;EX(4);3;*TERM:
$END:
```

Figure 2. Single Server Model

We can illustrate the use of resources in SIMNET II by embellishing the example in Figure 2. Suppose that the model represents a manufacturing situation in which the item is inspected prior to being processed in a milling machine. In this case, we replace facility SRVR in Figure 2 with two tandem facilities named INSPECT and PROCESS. It is assumed that both inspections and processing are carried out by a single operator.

Figure 3 gives the new model. Facilities INSPECT and PROCESS use the same resource OPR which is acquired at the start of INSPECT and released after the completion of PROCESS as shown in field 5 of both facilities. The code OPR(1,0,0,0) at INSPECT indicates that one unit of resource OPR is acquired in zero transit time with no units returned. At PROCESS, the code OPR(0,0,1,0) signifies that the facility does not acquire OPR but will release it back in zero transit time. In general, resources can be defined to represent priority classes with and without preemption privileges.

```
$RESOURCES: OPR; (INSPECT, PROCESS):
$BEGIN
   ARIV        *S; EX(5):
   LINE        *Q:
   INSPECT     *F;; EX(5);/5/OPR(1,0,0,0):
   PROCESS     *F;; EX(2.5);/5/OPR(0,0,1,0);*TERM
$END:
```

Figure 3. Use of Resources

The use of the resource field in Figure 3 demonstrates how a field of a facility is used to accomplish what is normally achieved in other languages by using *special blocks* or *nodes*. In essence, a node in SIMNET II is designed to be *self contained* in the sense that each node's statement includes all the information needed to process a transaction through the node. Naturally, if certain data are not needed in a node, we simply default its associated field.

4. SIMNET II BRANCHES

SIMNET II uses seven types of branches to link nodes:

1. always (A)
2. select (S)
3. conditional (C)
4. probabilistic (P)
5. dependent (D)
6. exclusive (E)
7. 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 P-branch will link nodes according to preset probabilities. An L-branch is taken only when no other branches from the node can be taken. SIMNET II allows any number of branches (of any type) to emanate from the same node.

As transactions traverse a branch, they can execute two types of assignments:

1. arithmetic
2. special

The first type is used to effect changes in the model's variables. The second type, on the other hand, performs the important modeling function of controlling the flow of other transactions anywhere in the network.

4.1 Arithmetic Assignment

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

```
$DIMENSION Sample(20),Table (10,2)
SUM = SUM + (Sample(J)-1)*Table(K,L)
```

The variables Sample and Table are subscripted whereas the variables SUM, J, K, and L are nonsubscripted. The rules governing the construction of mathematical expressions are the same as in FORTRAN.

SIMNET II also allows the use of the conditional statements IF-THEN-ELSE-ENDIF and WHEN-DO-ENDWHEN. Additionally, loops can be implemented using FOR-NEXT constructs. The following example illustrates the use of these statements.

```
IF,SUM = K,THEN,
   FOR,I = 1,TO,N,DO,
         WHEN,I = 3,DO,
               LOOP = CONTINUE,
         ENDWHEN,
         Sample(I) = Sample(I)**2,
   NEXT,
ELSE,
   SUM = SUM + 1,
ENDIF
```

Notice that the assignment LOOP = CONTINUE is used to skip to the end of the loop. A companion assignment LOOP = BREAK can be used to cancel the remainder of the loop altogether.

4.2 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, WHEN-ENDWHEN, and FOR-NEXT statements. As an illustration, consider the following statement:

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

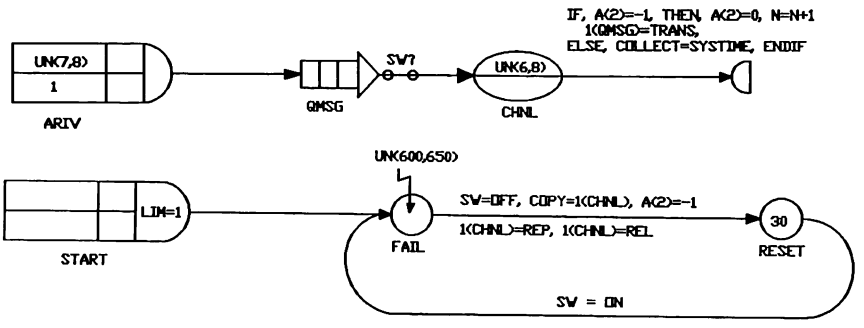
1. Source node control
2. Queue node control
3. File manipulations
4. Attributes control
5. Statistical variables collection
6. External files READ/WRITE capability

By using these special assignments, the modeler can control the operation of any of the nodes during the simulation. File manipulation assignments are particularly useful for rearranging, deleting, and swapping transactions among queues and facilities.

4.3 Example of Special Assignments

The example in this section is designed to demonstrate the use of SIMNET Special Assignments. Messages arrive every UN(7,8) seconds for transmission over a single channel. It takes UN(6,8) seconds to transmit a message. However, every UN(600,650) seconds, the channel malfunctions and any ongoing transmission must be started anew ahead of all waiting messages. It takes about 30 seconds to reset the channel.

Figure 4 summarizes the model which consists of two disjointed segments representing the transmission channel and the failure-repair cycle. When the channel fails, the transaction leav-




```
$PROJECT;Transmission Channel;8 June 90;Taha:
$DIMENSION;ENTITY(50);A(2):
$VARIABLES:  SYS TIME;;TRANSIT(1):
             PRCNT ABORTED;RUN.END;N/COUNT(CHNL)*100:
$SWITCHES: SW;;QMSG:
$BEGIN:
  ARIV      *S;UN(7,8);;1:                    !Messages arrive
  QMSG      *Q:                               !Wait in queue
            *B;CHNL/1;SW=ON?:                 !SW controls QMSG
  CHNL      *F;;UN(6,8):                      !Transmission
            *B;TERM;;
              IF,A(2)=-1,THEN,                !Aborted transmission
                A(2)=0,                       !Reset A(2)
                N=N+1,                        !Count aborted messages
                1(QMSG)=TRANS,                !Place at head of queue
              ELSE,
                COLLECT=SYS TIME,             !Message completed
              ENDIF%:
! ---------------------------------- Channel failure----------------------------------
  START     *S;5/LIM=1:
  FAIL      *A;UN(600,650):                   !Time to failure
            *B;RESET;;                        !Failure occurs
              SW=OFF;                         !Block QMSG
              COPY=1(CHNL);                   !COPY A(.) in CHNL
              A(2)=-1;                        !Set A(2)=-1
              1(CHNL)=REP;                    !Replace A(.) in CHNL
              1(CHNL)=REL%:                   !Release CHNL
  RESET     *A;30:                            !Resetting time
            *B;FAIL;;SW=ON%:                  !Unblock QMSG
$END:
$RUN-LENGTH=9000:
$STOP:
```

Figure 4. Transmission Channel Model

ing auxiliary FAIL immediately turns OFF switch SW to prevent messages from leaving QMSG whose exit end is controlled by the condition SW = ON? An ongoing message in facility CHNL is then RELeased by executing the assignment 1(CHNL) = REL. However, in order for the model to recognize that an aborted message leaving CHNL must be retransmitted, we must "tag" it *before* it is released from CHNL. We achieve this result by resetting A(2) = -1 for any aborted message [otherwise A(2) = 0]. The assignment COPY = 1(CHNL) changes the attributes of the transaction leaving FAIL to those of 1(CHNL). Next, we set A(2) = -1 and then execute 1(CHNL) = REP, in effect changing A(2) inside CHNL to -1 while leaving A(1) unchanged. (If CHNL happens to be empty, none of the special assignments will have any effect.)

Upon leaving CHNL, a transaction having A(2) = -1 is identified as an aborted message. Thus, the conditional branch from CHNL executes the assignments A(2) = 1,1(QMSG) = TRANS which places the aborted message [with its A(2) reset to zero] back at the head of QMSG. Following the repair of the CHNL (exit from RESET), we execute SW = ON to move the transaction into CHNL for retransmission.

5. DATA INITIALIZATION IN SIMNET II

An important feature of SIMNET II is the ability to provide run-specific data initialization for the model's variables. This means that a single simulation session may include several runs, each with its own initial data. The following types of initialization are available in SIMNET II.

1. Initial file (queues and facilities) entries
2. Discrete probability density functions
3. Table lookup functions
4. Array (subscripted) variables values
5. Non-subscripted variables values
6. Functions or mathematical expressions

The general format for these initial data is as follows:

```
$Data type: i-j/initial data
.
.
.
.
repeats
```

The code i-j defines the range of simulation runs for which the simulation is applicable. This type of format provides greater flexibility in initializing the variable of the model.

The example below shows the use of $DISCRETE-PDFS, $TABLE-LOOKUPS, and $FUNCTIONS initial data within the context of a general job-shop scheduling problem. This problem normally represents a difficult situation. However, with the use of data initialization, a general model can be developed for the problem.

5.1 Job-shop Scheduling Model

Consider the situation where two types of jobs are processed on three different machines. Each job has its particular route among the three machines as summarized below. It is assumed that the ratio of the number of Type 1 to Type 2 job is 2 to 3.

Job Type	Machine Route	Processing Time
1	3-2-1	EX(1),GA(2,1),WE(2,1)
2	2-3	UN(1,2),NO(1,.1)

The model is shown in Figure 5 (see next page). The initial data of the model gives the ratio of the two types of jobs by using a $DISCRETE-PDFS initial data statement. The routes of the jobs and their processing times are summarized by $TABLE-LOOKUPS and $FUNCTIONS, respectively. Each set of initial data elements is preceded by the range 1-1, indicating that these data apply to run number 1. In general, any number of ranges, each with a different set of data may be used in the model, thus allowing the execution of multiple runs with different initial data in one simulation session.

The model starts with the arrival of jobs from source JOBS according to an EXponential interarrival time with mean 3 time units. The machines in the jobshop are modeled as a PROC(1-3), a SIMNET II facility that allows the modeling of repetitive segments conveniently. The PROC is entered through auxiliary DISPATCH which selects the correct queue of the job depending on its route as summarized in $TABLE-LOOKUPS. Initially, when the job arrives from source JOBS, it will carry three attributes. A(1) is the job type, A(2) is the sequence of the operation, and A(3) is the machine number. These attributes are given the descriptive names job#, oper#, and mach#. Within the PROC, these attributes are modified properly to represent the next operation before being sent back to DISPATCH. The model in Figure 5 actually applies to jobshop with any number of machines, job types, and routes. All we have to do is to change the initial data accordingly. In fact, we can simulate several situations in one session simply by changing the initial data.

6. ISES SIMULATION ENVIRONMENT

ISES is a SIMNET II interactive environment that comprises all the phases of a simulation study. Figure 5 summarizes the main components of the ISES System, including

1. Input editor
2. SIMNET II interpretation/execution program
3. Animation program
4. Output program

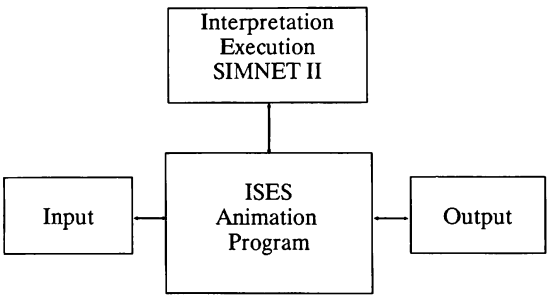
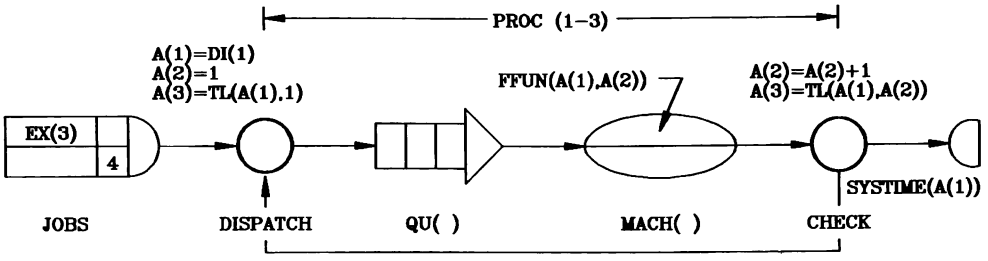


Figure 6. Organization of ISES

All four components can be reached at any time during the execution of a SIMNET II model. Thus, the input editor may be called upon by ISES to initiate a new model. If during the interpretation phase an error is discovered, ISES will automatically invoke the editor, stopping the model at the location where the error is detected. Following a successful interpretation of the model, ISES will create the animated model directly from the interpreted file without input from the user. During the animation, the user can invoke the plot and histogram capabilities which can be run concurrently with animation. The plot capability is particularly useful for estimating the length of the transient period, following which the warm up period is truncated and the steady state results can be collected.

Figure 7 shows how ISES accesses the input editor. By selecting the *Error* option under *Edit*, ISES will discover errors and automatically place the cursor at the proper location for prompt correction as illustrated in Figure 8. This process will be repeated, one error at a time, until a successful interpretation is accomplished.

To set up the animation option for a successfully interpreted model, the user will select the *Node* option under *Edit* as shown in




```
$PROJECT;Job Shop;2/21/89;Taha:
$DIMENSION;ENTITY(80),A(4):
$attributes: job#,oper#,mach#:
$VARIABLES: SYS TIME(1-2);;TRANSIT(4):
$BEGIN:
  JOBS        *S;EX(3);;4:                          !Jobs arrival
              *B;DISPATCH;;job# = DI(1),            !Job number
                          oper# = 1,                !Operation 1
                          mach# = TL(A(1),1)%:      !Machine #
!-----------------------------------------START JOBSHOP
              *PROC(1-3):                           !3 machines
  DISPATCH    *A;NIL;*Q(mach#):                     !Goto Q(A(3))
  Q()         *Q:                                   !Wait in queue
  MACH()      *F;;FFUN(job#,oper#):                 !Processing time
              *B;CHECK;;oper# = oper# + 1;          !Next operation
                        mach# = TL(job#,oper#)%:    !Next machine
  CHECK       *A;NIL:
              *B;TERM/1;mach# = 0?;                 !End of route?
                        /4/SYS TIME(job#)%:         !Compute SYS TIME
              *B;DISPATCH/L:                        !Else, next machine
              *ENDPROC:
!-------------------------------------------END JOBSHOP
$END:
$RUN-LENGTH = 600:
!-----------------------------------------JOB TYPE
$DISCRETE-PDFS: 1-1/2/1,.4; 2,.6:                   !2 job types
!-----------------------------------------JOB ROUTES
$TABLE-LOOKUPS: 1-1/4/1,3; 2,3; 3,1; 4,0:           ! 3-2-1-0
                3/1,2; 2,3, 3,0:                    ! 2-3-0
!-----------------------------------------JOB TIMES
$FUNCTIONS: 1-1/EX(1),GA(2,1),WE(2,1.1),;           !Job 1
            UN(1,2),NO(1,.1):                       !Job 2
$STOP:
```

Figure 5. Jobshop Scheduling Model

Figure 9. A generic display of all the model's nodes at the top of the screen will then be produced as Figure 10 shows. The user can then position these nodes where desired on the screen as demonstrated in Figure 11. Figure 12 shows the final network positioned in the middle of the screen. All this work is done simply by clicking the mouse at the appropriate locations. ISES will automatically link the nodes using the data from the interpreted file of SIMNET II.

A plot of the model's variables can be secured concurrently during animation as shown in Figure 13. At the point where it appears that steady state has been reached, the user can *set* the *transient* option which will truncate the transient period and start collecting the steady state data. ISES also provides an on-line histogramming capability as shown in Figure 14.

Once the animation screen is set up, the user can run the model. During execution, each node will carry a number showing the flow of transactions in the network. One of the important execution options is the ability to plot selected statistical variables of the model concurrently with the animation as shown in Figure 13. By observing these plots, the user can then estimate the length of the transient period. At that point, the execution can be interrupted and a command issued to delete the transient period and start collecting the steady state results.

It must be stated that the user can interrupt the animation process at any time and simply invoke the execution capabilities of SIMNET II. The process can alternate between SIMNET II and ISES as desired.

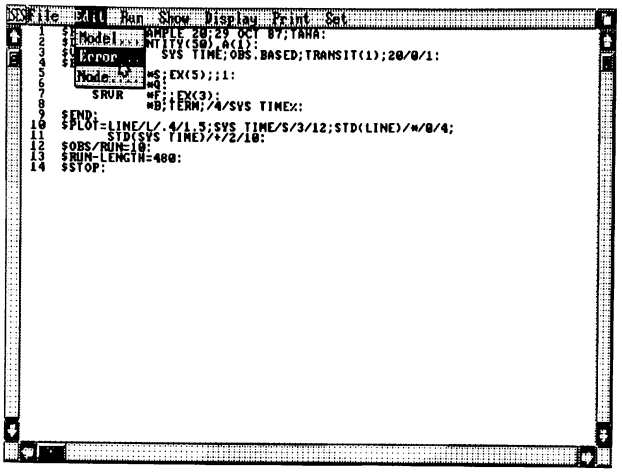


Figure 7. Selection of Error Option

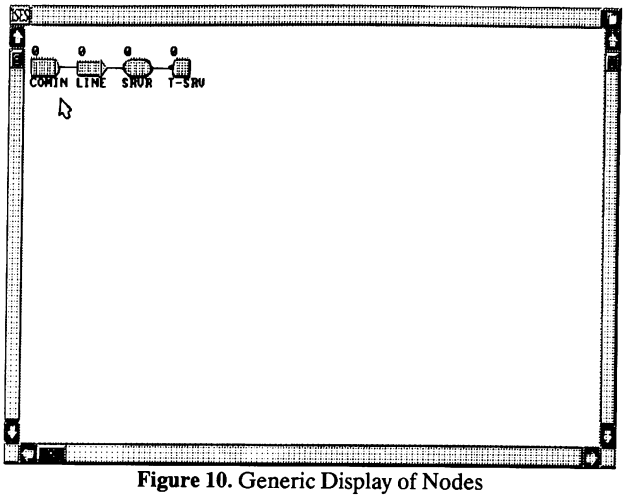


Figure 10. Generic Display of Nodes

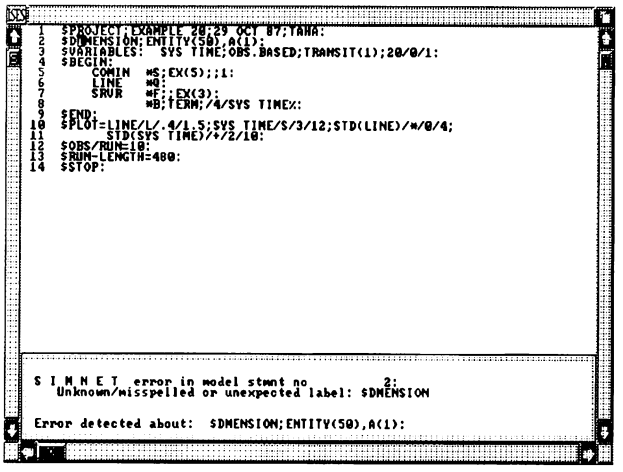


Figure 8. Correction of an Error

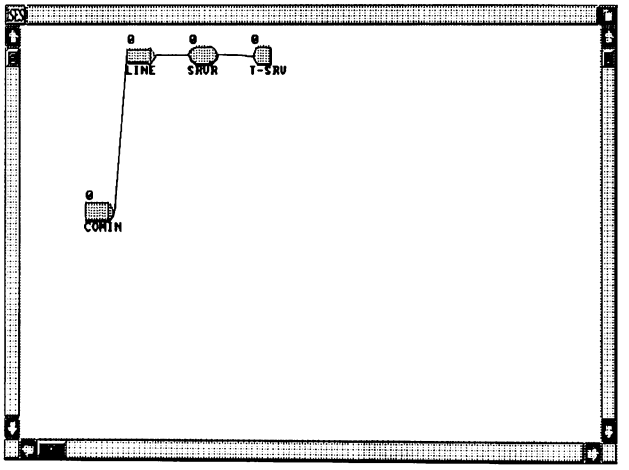


Figure 11. Positioning the Nodes on the Screen

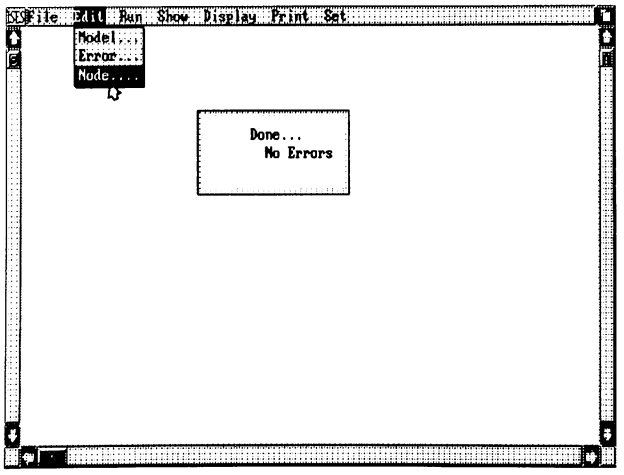


Figure 9. Setting Up the Animation Option

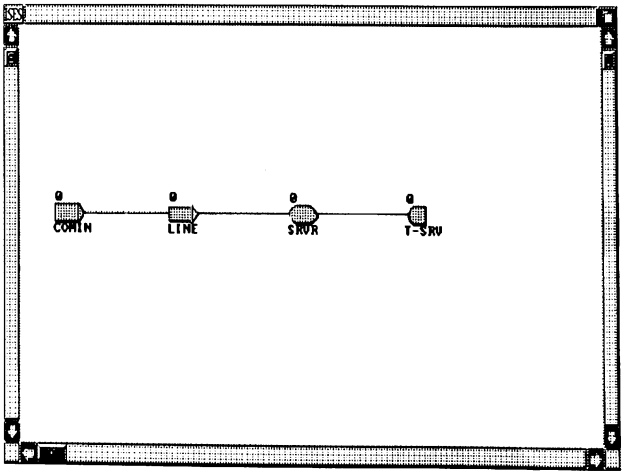


Figure 12. Network on Screen

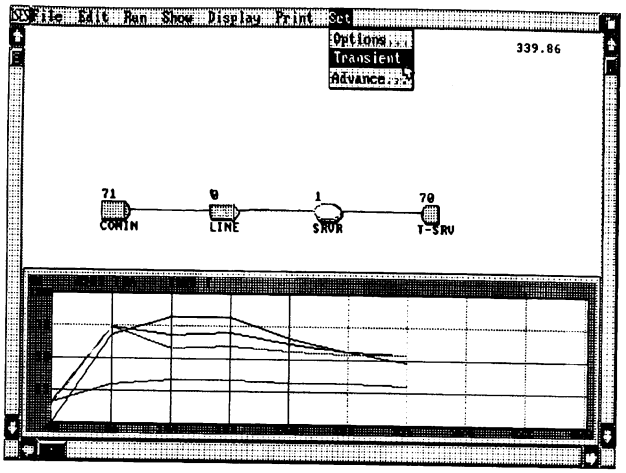


Figure 13. Plot of Variables

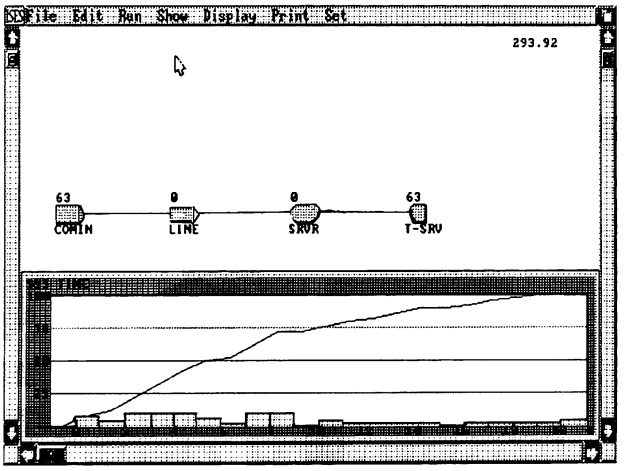


Figure 14. On-line Histogramming

7. CONCLUSION

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 give SIMNET II a superior modeling capability.

The ISES simulation system puts SIMNET II in a user-friendly environment for editing, executing, and animating simulating models. The fact that ISES generates the animation model without any special programming effort on the part of the user is an important feature of the system.

REFERENCES

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

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

Taylor, R. B. (1990), "An Integrated Visualization Environment For SIMNET II," Ph.D. Dissertation, Department of Industrial Engineering, University of Arkansas, Fayetteville, Arkansas.