Modeling and Simulation of Data Communication Networks Using SARA

Rami R. Razouk

Department of Information and Computer Science
University of California, Irvine
and
Hughes Aircraft Co.
Fullerton, California

85-05

ABSTRACT

The selection of an appropriate simulation language can have a profound impact on the success of a simulation study. The available options range from domain-specific simulation languages to general-purpose programming languages. These languages are often supported by a collection of tools which form a simulation *system*. This paper examines UCLA's SARA (Systems ARchitects' Apprentice) system and explores its' usefulness in modeling and simulating a data communications network. Based on experimental use of SARA's tools, the system is evaluated with respect to required expertise, modeling power, as well as measurement and reporting capability.

Technical Report #85-05

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

January 31, 1985

Modeling and Simulation of Data Communication Networks Using SARA

Rami R. Razouk

Department of Information and Computer Science
University of California, Irvine
and
Hughes Aircraft Co.
Fullerton, California

ABSTRACT

The selection of an appropriate simulation language can have a profound impact on the success of a simulation study. The available options range from domain-specific simulation languages to general-purpose programming languages. These languages are often supported by a collection of tools which form a simulation *system*. This paper examines UCLA's SARA (Systems ARchitects' Apprentice) system and explores its' usefulness in modeling and simulating a data communications network. Based on experimental use of SARA's tools, the system is evaluated with respect to required expertise, modeling power, as well as measurement and reporting capability.

Introduction

Choosing an appropriate language for a simulation study is a complex and difficult task. The chosen language has a significant impact on the cost of constructing a model, on the cost of executing the simulation experiments, and on the effectiveness of the study in providing the desired performance measures. Simulation languages range from highly specialized languages whose scope is rather limited, to general-purpose programming languages with wider applicability. Special-purpose languages are tailored to a specific set of users and often require more problem-specific expertise than general programming expertise. General-purpose programming languages (e.g. Fortran, PL/I, Simula) are more powerful in the sense of being able to model more types of systems, but are usually more difficult to use and require a great deal of programming expertise. This work is intended to evaluate UCLA's SARA (System ARchitects' Apprentice) [EstG 78] system to determine where it fits along that spectrum. The evaluation of SARA is based on a model of a "typical" data communications network which was previously modeled in other simulation languages (e.g GPSS [GorG 78]).

Section 1 presents an overview of SARA. This paper is not intended as a tutorial on SARA, so readers are referred to [VerM 83a,b] for more detailed information. Section 2 outlines the chosen problem and identifies the desired performance measures. Section 3 presents the SARA model and discusses the results obtained during simulation. Section 4 draws conclusions about the strengths and weaknesses of SARA.

1. SARA

The SARA system was developed by a research group at UCLA. The system was implemented on MIT's Multics system with the objective of supporting the systematic design of hardware and software. SARA was intended as a design environment taking the user from requirement specification all the way to implementation. SARA's simulator is only one of a set of tools to be used in evaluating designs. Consequently, SARA places great emphasis on the sound theoretical foundation of the models (Petri-Net theory), and forces designers to model systems accurately.

SARA supports modeling of computer systems at many levels of abstraction. SARA distinguishes between modeling the *structure* of a system and modeling its *behavior*. A structural model (using the modeling language SL/1) allows a designer to decompose a system into a collection of subsystems (modules). Structural modules have no type or specific behavior. They are simply empty shells which are to house behavioral models. This de-coupling of structure from behavior offers the possibility of using multiple behavioral models to describe the behavior of a particular component.

The behavioral model currently supported by SARA is the Graph Model of Behavior (GMB) [VerM 83b]. The GMB is composed of three related models: control graph, data graph, and interpretation. The control graph portion of a GMB model describes control flow sequences and is essentially a Petri Net [PetJ 81] (with some important extensions, see [VerM 82, VerM83a,b]). Control graphs are constructed using nodes (depicted as circles) which are intended to model events, and arcs which are intended to model precedence conditions among events. *Tokens* (black circles) on arcs enable the nodes and allow them to be initiated (modeling the occurrence of events). The flow of tokens across arcs and nodes models the flow of control of the system, as well as contention for shared resources.

In addition to the control graph, the GMB supports the description of data and data processing through the use of a data graph and "interpretation" (data formats and program segments) models. The data graph consists of:

- Controlled processors which are depicted as hexagons and which model the activities which correspond to events. There is a many to one mapping between nodes and processors.
- Uncontrolled processors which are depicted as triangles and which models events which are occurring continuously (e.g. combinational hardware).
- Datasets which are depicted as rectangles and squares, and which models collections of data. Dataset *queues* model queueing of data values.
- Data arcs which models the paths along which data may flow.

These control and data primitives have simple, well-defined semantics. The details of data formats, data processing and timing must be described in the interpretation model using a superset of PL/I (PLIP). As a result, a thorough understanding of PL/I is required in order to construct a model of any complexity.

SARA contains tools to support:

1. SL/1 model construction. A textual form of the language is used for input.

2. GMB model construction. A textual form is used to describe GMB models and to map them onto structural modules.

3. Interactive simulation. The simulator supports breakpoints and allows a user to examine and modify simulation variables (datasets in the GMB models).

4. Control-flow analysis. Since the control graph of the GMB is equivalent to a Petri Net, it can be exhaustively analyzed for deadlocks and *critical transitions* [RazR 80a,b].

5. Storage and retrieval of models. A primitive library system has been made available, using Multics' file system. No general-purpose database management system is currently used.

2. The Problem

The problem selected for modeling is a standard store-and-forward packet switched network. The design parameters being investigated were:

1. Network Topology: three topologies ranging from 3 to 7 nodes were to be explored (see figure 1).

2. Channel capacities: seven separate runs were required to investigate different communication channel capacities.

3. Nodal Processing capacities: utilization measurements were required to determine if the nodes are sufficiently powerful.

4. Use of data compression: some of the seven runs were expected to investigate the advantages of using data compression techniques (50% reduction in message length).

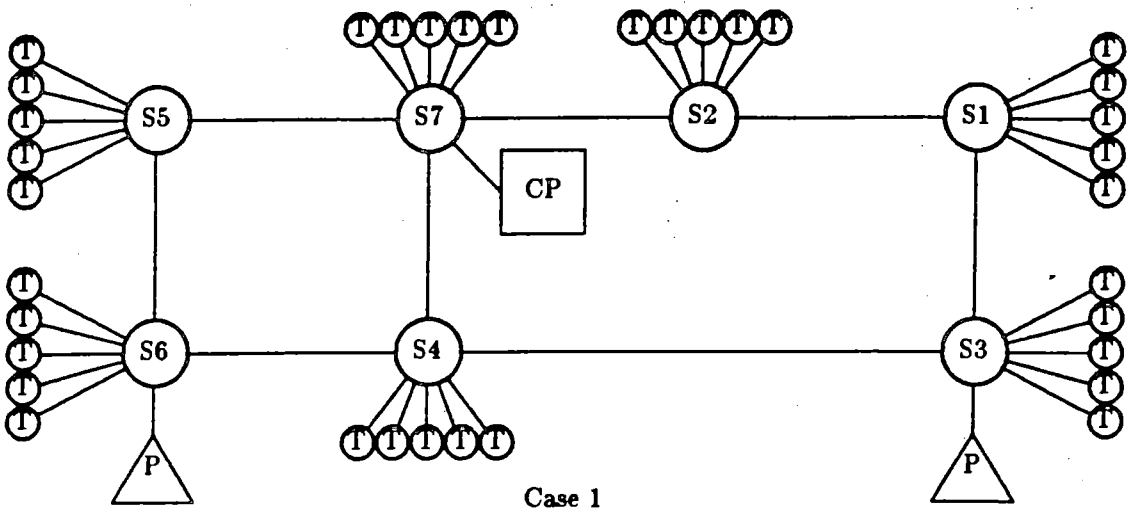
Table I shows how the design parameters were to be changed during the seven simulation runs. In order to perform these experiments, the problem statement also provided:

1. Message generation rates

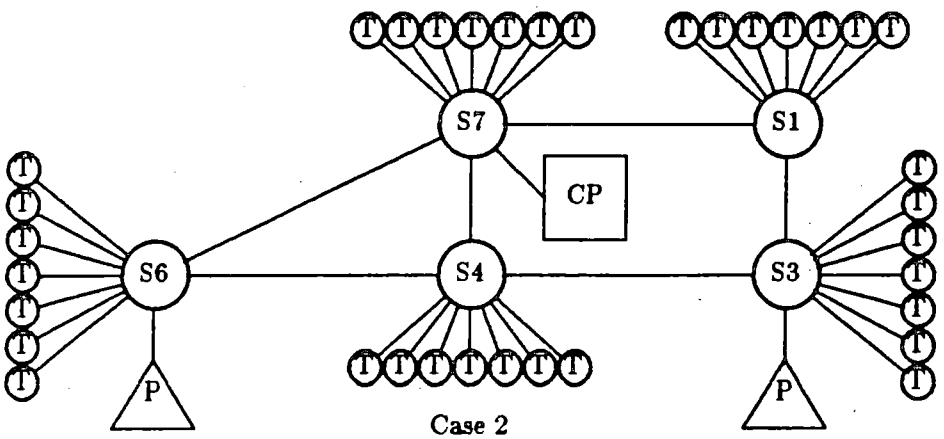
2. Message type distribution

3. Message source distribution

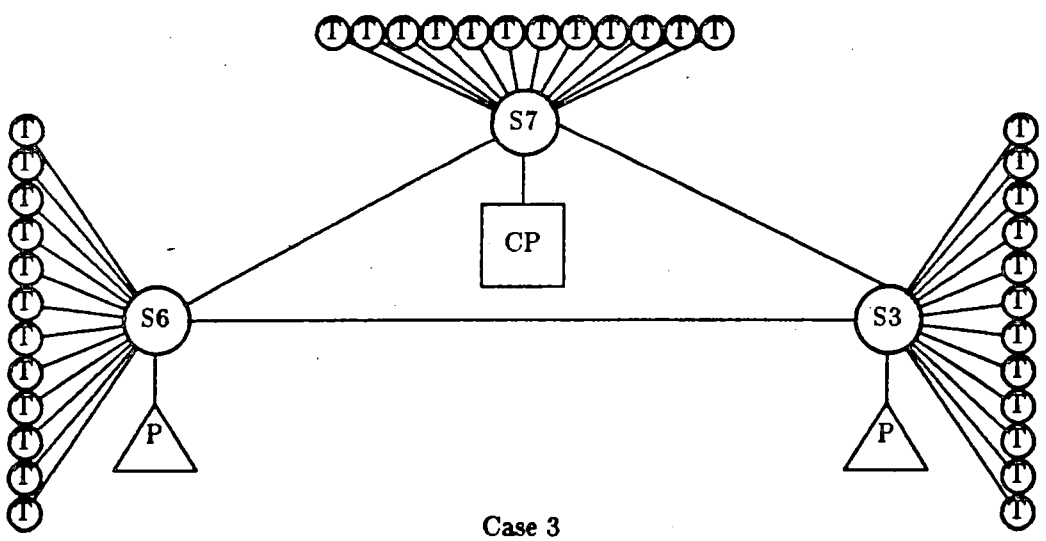
4. Message routing tables



Case 1



Case 2



Case 3

Figure 1. Network Topologies

Table I. Channel Capacities

Case	Run	Data Compr. [a]	S - S Channels (kb/s)	S - P Channels (kb/s)	S - CP Channels (kb/s)	S - S Channels (kb/s)
1	A	No	19.2[b]	40.8	230.4	4.8
	B	Yes	19.2[b]	40.8	230.4	4.8
	C	Yes	9.6[c]	9.6	230.4	4.8
	D	Yes	9.6	9.6	230.4	9.6
	E	Yes	9.6	9.6	230.4	4.8
2		Yes	9.6	9.6	230.4	9.6
3		Yes	9.6	9.6	230.4	9.6

[a] Data Compression factor is 50 percent.
[b] All S-S Channels except 4-7, which is 40.8 kb/s.
[c] All S-S Channels except 4-7, which is 19.2 kb/s.

5. Channel capacities

6. Processing rates

Scenario of Operation

The network consists of terminals (T-nodes), switches (S-nodes), two processors (P-nodes) and a single central processor (CP-node). Switches are interconnected by full duplex channels to form the communications subnet. The subnet provides communication facilities between terminals, processors and the central processor.

Terminals generate data messages and hardcopy display requests. Each data message is destined for another terminal, with copies sent to one processor and the central processor. Hardcopy display requests are sent only to the central processor for processing. Hardcopy display requests are answered by hardcopy output messages.

Each processor generates data messages which are sent to one terminal, the other processor and the central processor. Processors also generate display requests which are sent and processed by the central processor. Unlike terminals, processors are capable of requesting graphics as well as hardcopy output.

The central processor receives data and display request messages. Display request messages are processed and sent back to the requesting node as display output messages.

Switches are responsible for performing all the switching in the network based on fixed routing tables.

This otherwise simple scenario is complicated by the fact that messages are partitioned into packets of no more than 4000 bytes.

Modeling Issues

The performance of the network was to be evaluated based on the queue length, queueing delay, storage utilization and processor utilization of each node, and queue length, queueing delay, and utilization of each channel in the network. In addition, the user-perceived delays were to be measured in terms of transit times of messages from source to destination. Display message delays were to be measured in terms of round-trip times. Transit times for each pair of nodes in the network were to be measured.

The model made several simplifying assumptions which were consistent with those made in earlier modeling efforts by other authors. This made a comparison of the models possible, although such a comparison is beyond the scope of this paper. First, a simplifying assumption was made that all terminals connected to each node can be simulated by one terminal with a higher message generation rate connected to a channel with a higher bandwidth. This assumption was considered reasonable since no measurements were required for terminals and terminal-to-switch channels. Secondly, partitioning of messages into packets was not modeled, making the network a message-switched network. This simplifying assumption can lead to erroneous results. Section 3 of the paper discusses how this aspect of the network can be modeled more accurately. Finally, storage utilization in each processor was not modeled at all because of difficulties found in modeling and measuring storage utilization.

Below we discuss how that initial model was constructed and how it can be altered to model the system more accurately and therefore yield additional measurements.

3. The SARA model

The SARA model of the network consists of a structural model and a set of behavioral models. The structural model is used to describe the topology of the network and to assign names to terminal clusters (T-nodes), switches (S-nodes), processors (P-nodes), and the central processor (CP-node). Behavioral models are used to describe control-flow, data-flow, data processing, and timing delays.

Figure 2 shows the structural model of the simplest topology involving 3 switches. Modules T3, T6 and T7 model the three terminal clusters. Modules S3, S6 and S7 model the three switches in the network. Modules T-S3, T-S6, and T-S7 model the channels which connect terminals to switches. Modules S3-S7, S3-S6, and S6-S7 model the channels used to connect switches. Modules P8 and P9 represent the two processors which are connected to switches via modules P8-S and P9-S. Module CP represents the central processor. Module CP-S represents the channel which connects the central processor to switch S7.

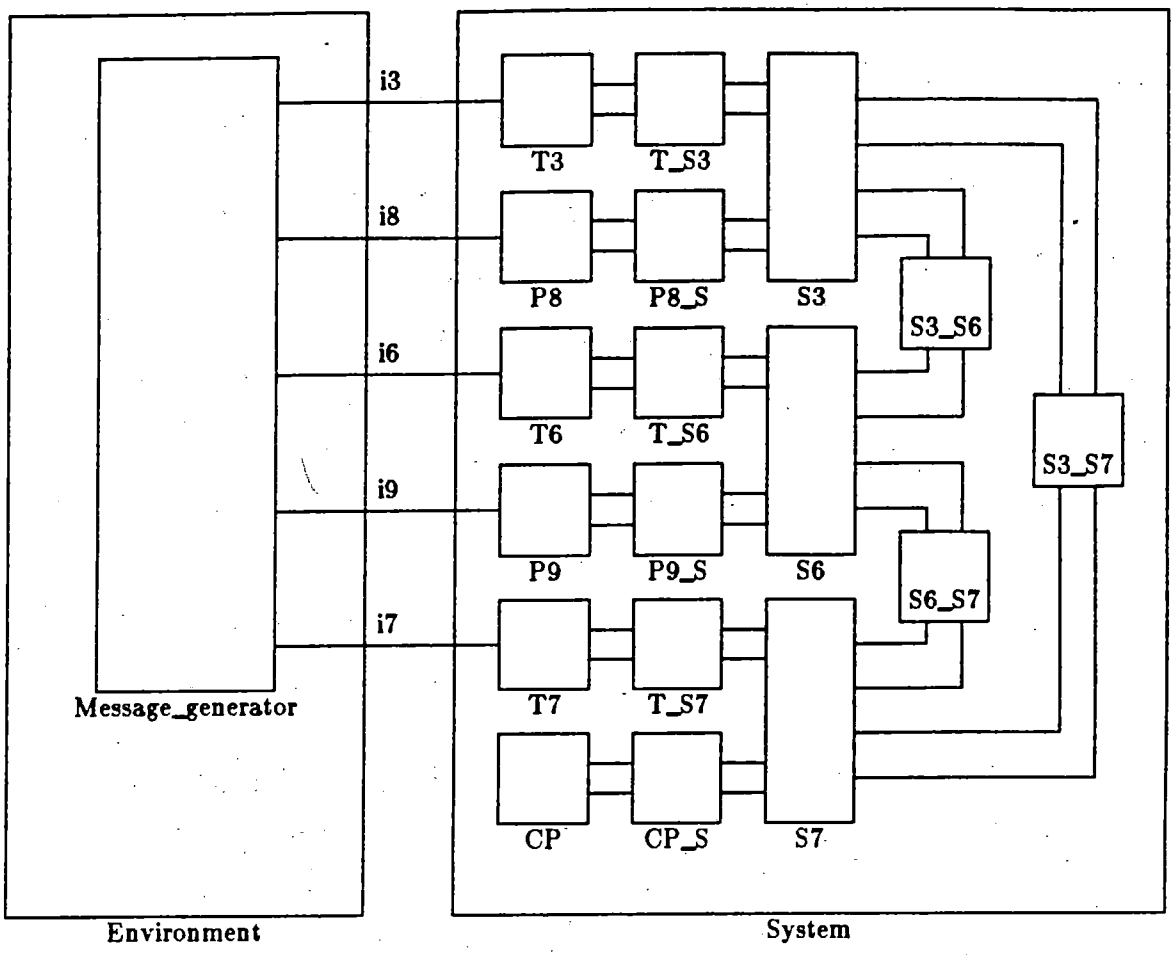


Figure 2. Structural Model of 3-Node Network

All the models described above comprise the SYSTEM being modeled. In the ENVIRONMENT we find a module called MESSAGE_GENERATOR which is responsible for generating data and display-request messages at the specified rates.

The behavior of each module described above is modeled using the Graph Model of Behavior. While there are 19 modules in the structural model, only six behavioral models are required:

1. A model of the message generator.

2. A model of the CP node.

3. A model of an P node.

4. A model of a T node.

5. A model of an S node.

6. A model of a channel.

These six behavioral models are described below and have been parameterized in a way which allows them to be modified trivially to map onto the different structural modules. The basic approach taken in all these models is that messages traveling in the network can be modeled as tokens flowing through the control graph, and data values flowing through dataset queues. The placement of a token on an arc signals the arrival of a message whose parameters are defined in a dataset queue. Queue length measurements are derived from the numbers of tokens accumulated on various arcs. Service times are determined by data processors, based on the message parameters found in the dataset queues.

Model of message generator

The message generator models the environment with which the network interacts, and is responsible for generating both data and display request messages at specified rates. The message generator is also responsible for uniformly distributing the messages among the nodes. These two processes can be easily modeled using the GMB. Generation of data and display request messages is modeled as two concurrent nodes (NDATA0 and NDISPLAY0) as shown in the figure 3a. The nodes are initially enabled by tokens on arcs CALOOPDATA and CALOOPDISPLAY. The nodes are mapped to processors (PDATA0 and PDISPLAY0 in figure 3b) whose interpretations specify delays which model message interarrival times. For example, when node NDATA0 is activated, processor PDATA0 is called upon to create a message and to randomly select a delay. At the end of the delay, node NDATA0 terminates and generates tokens on arcs CADATA (modeling a message), and CALOOPDATA (re-enabling itself). Processor PDATA0 stores the parameters of the message (e.g. length) in dataset DATA_MESSAGE. Once messages are generated they are passed on to a separate part of the model (nodes NDATA1 and NDISPLAY1) which is responsible for targeting the messages according to a specified distribution. The OR (+) logic on the output of the nodes models the routing. The probability distributions are specified in the interpretation of processors PDATA1 and PDISPLAY1.

The only parameter in the model of the message generator is the dataset DSCOMPFACT which models the data compression factor. The dataset can be altered during simulation.

Model of CP

The central processor receives data messages and display requests and enqueues them for processing. Data messages are consumed with zero delay because no processing rate was specified for data messages. Display requests are transformed into display output messages and sent back to the source. The total processing time for the display request is based on the sum of the lengths of the request and the output. The length of the output display message is exponentially distributed with a mean of 10,000 or 6,300 characters (based on the type of the request).

The model of the CP module (see figure 4) is based on a queue (control arc QUEUE and dataset DSQUEUE) and a server (control node NSERVER and controlled processor PSERVER). The server is responsible for:

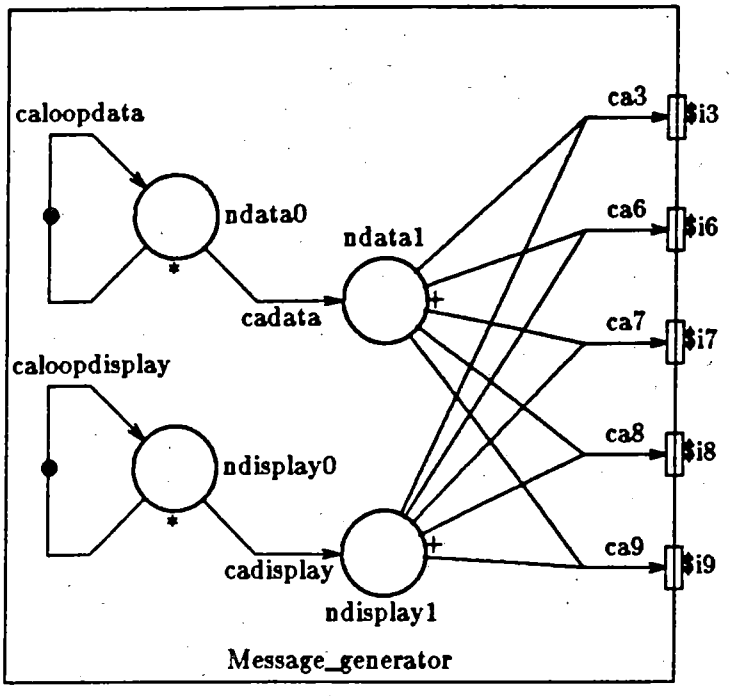


Figure 3a. Control Graph of Message Generator

1. Determining the service time, given that the service rate is based on the type of the message.

2. Consuming data messages while sending display messages back to the source (after some appropriate processing).

Since the CP produces messages it must also be able to model data compression. This is achieved through a dataset (DSCOMPFACT) which can be altered at simulation time. The other parameters of this model are datasets DSRATE0 and DSRATE1 which model the processing rates (in microseconds/char) of hardcopy display requests and graphics display requests respectively.

T nodes and P nodes

T nodes and P-nodes behave similarly (see figures 5 and 6). Each receives from the environment messages at a certain rate and places each message in an output channel. In both types of nodes data messages are triplicated and sent to three destinations. Display request messages are only sent to CP. The only difference between T nodes and P nodes is that T nodes need not process incoming messages (processor utilization measures are not required) while P nodes must enqueue and process incoming messages. As a result, P nodes include an extra queue (arc CARECEIVE) and a server (node NSERVER) for processing messages.

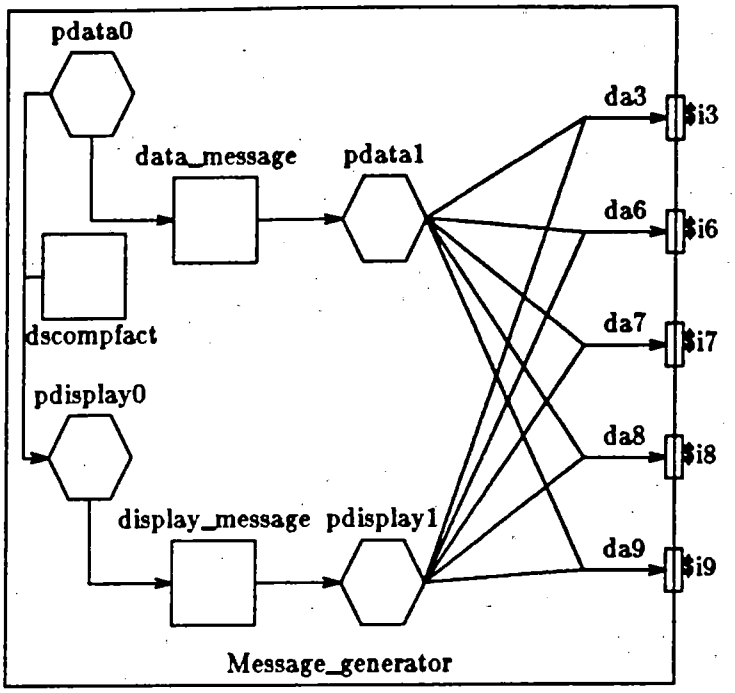


Figure 3b. data Graph of Message Generator

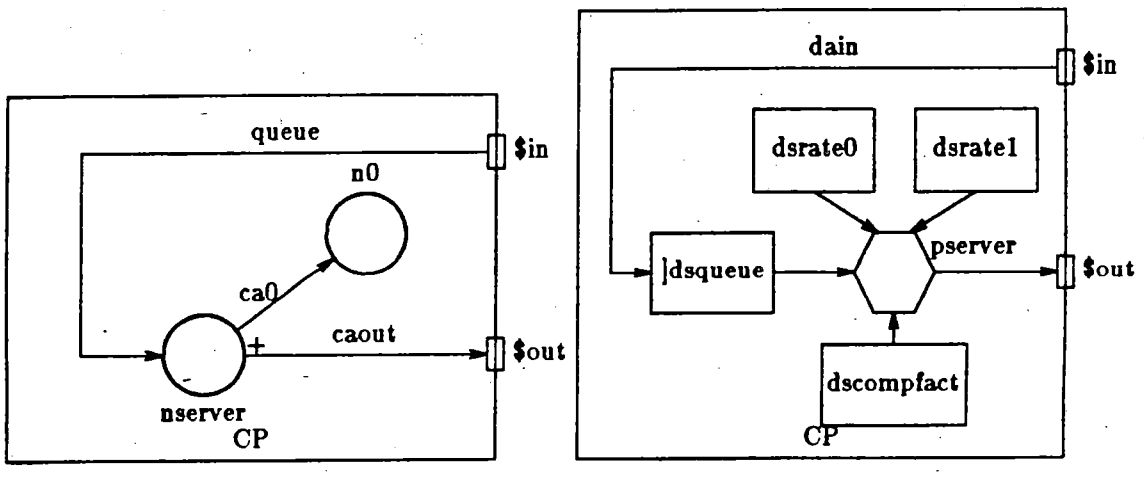


Figure 4. GMB Model of Central Processor

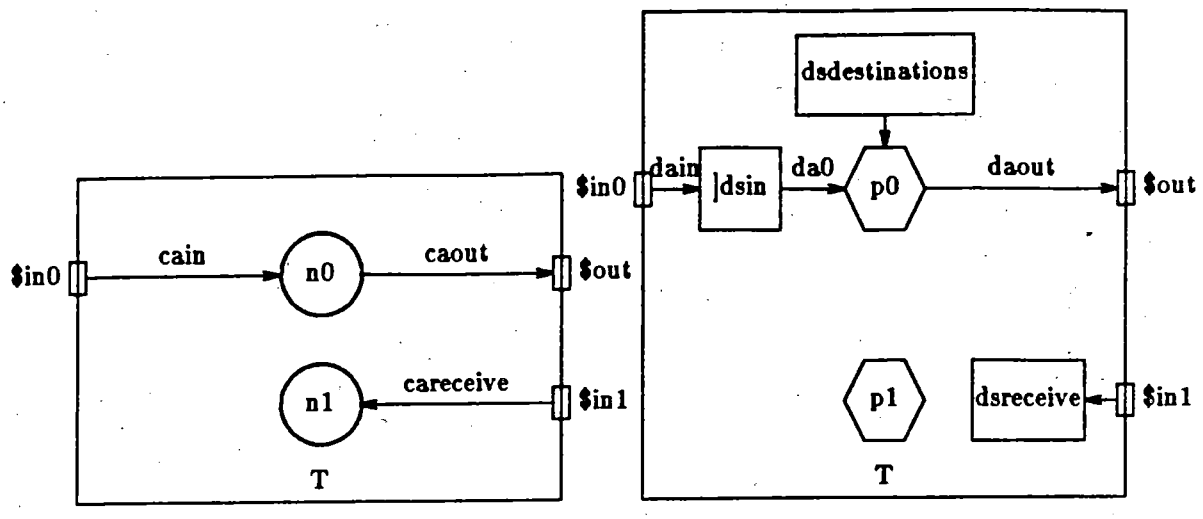


Figure 5. GMB Model of T node

It should be noted that one T node behaves differently from all other T nodes. Node 7 is required to alternate sending messages to P8 and P9. All other T nodes (and P nodes) always send messages to the same destinations. Because of this difference, two models for a T-node are required: one model for node 7 and another for all the other T-nodes. The difference between the two models is small and is limited to the data graph and interpretation models.

The parameters of the T-node and P models are the destinations for data messages. These destinations are modeled as a three element array in dataset DSDESTINATIONS. P-node models also include a parameter (dataset DSRATE) for the processing rate.

Model of an S-node

Switching nodes are responsible for routing of messages through the network. Each S-node has several incoming and outgoing channels. Incoming messages from all channels are enqueued for processing. The processing of a message consists of determining the "next" destination of the message, and transferring the message to the appropriate outgoing channel. Since routing tables are identical for all switching nodes, we chose to model the routing information as an external function called from within the model of each S-node. The routing table could have easily been included as a dataset in the model, thereby allowing for the modification of the information it contains during simulation.

The model of an S-node (see figure 7) consists of a queue (control arc QUEUE) and a server(node NSERVER), with the server responsible for routing. The largest portion of the S-node model deals with mapping the "next destination" to an outgoing channel.

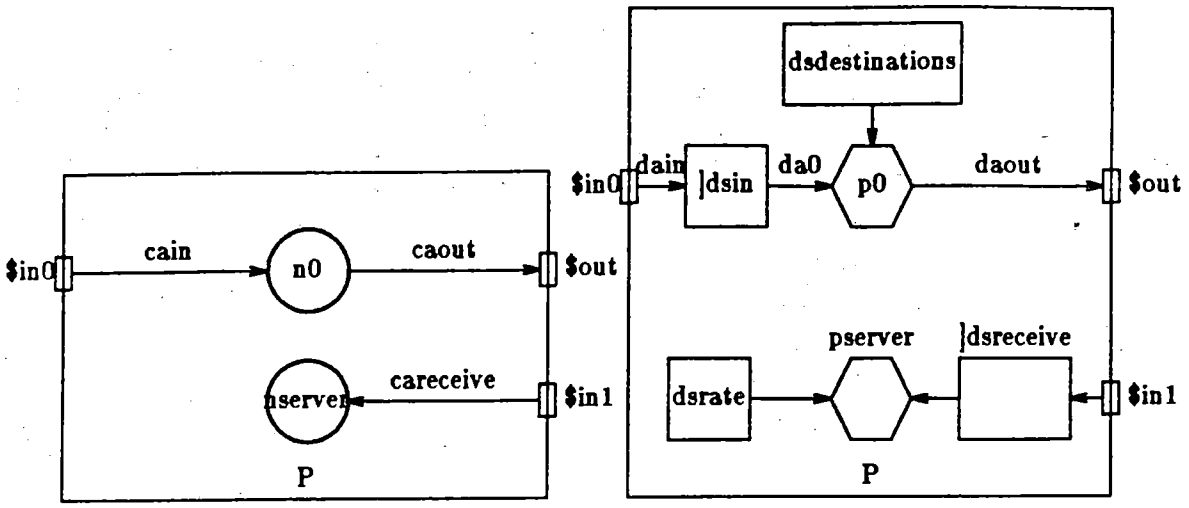


Figure 6. GMB Model of P node

The parameters of the S-node model are: dataset DSRATE which represents the processing rate of the node; dataset DSCURRENTNODE which is the identity of the node itself; dataset DSROUTINGINFO which is used to determine the mapping between destinations and outgoing channels.

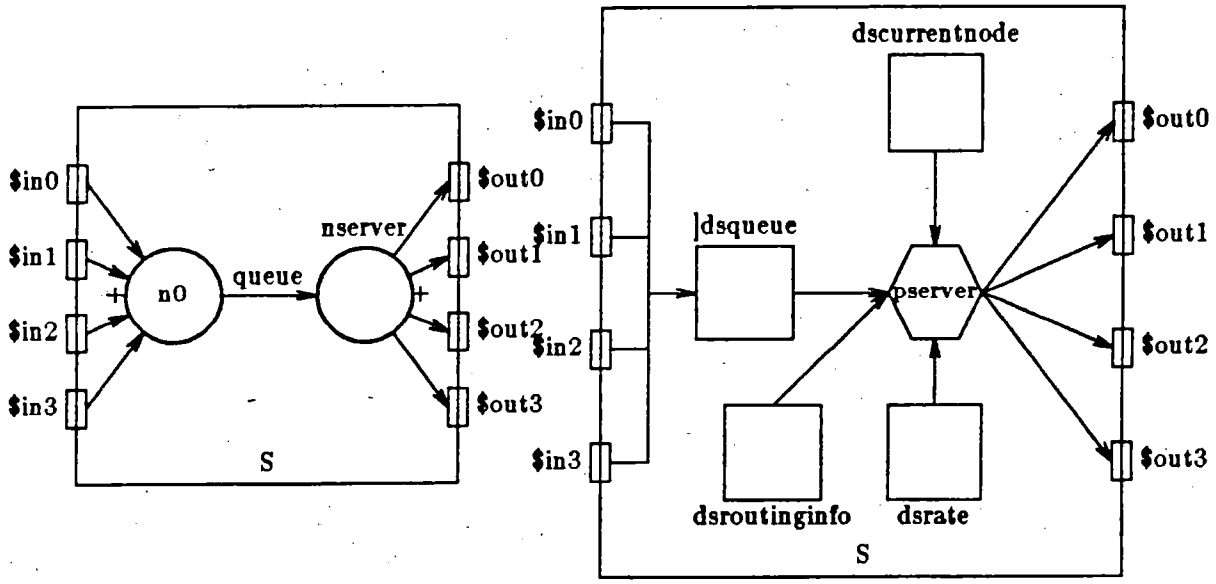


Figure 7. GMB Model of Switch Node

Model of a Channel

There are four types of full-duplex channels in the network being modeled: P-S channels, CP-S channels, S-S channels and T-S channels. All the channels behave similarly and differ only in channel transfer rate. A single model (see figure 8) is used to describe all channels. The model has been parameterized to handle the alteration of transfer rates at simulation time.

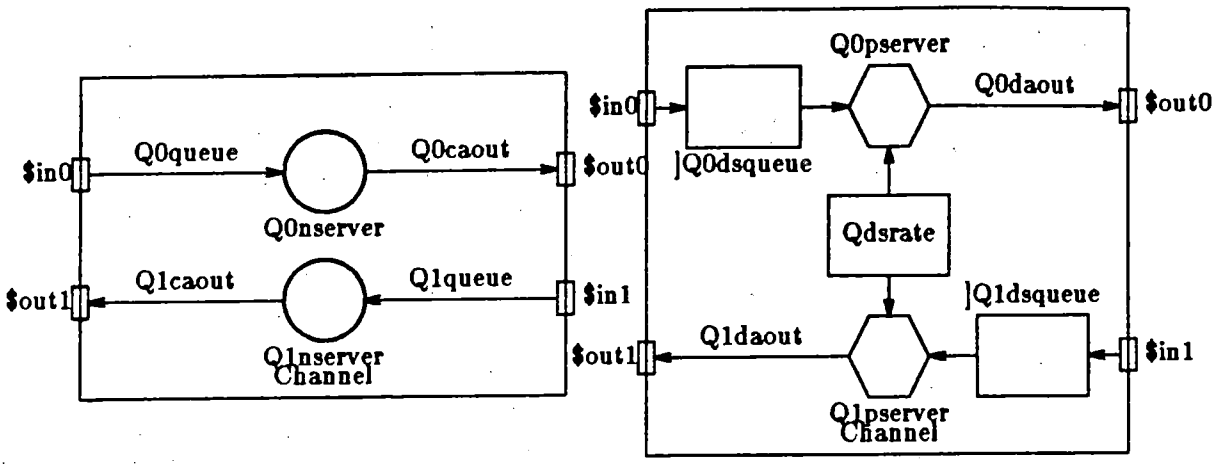


Figure 8. GMB Model of Communication Channel

A full-duplex channel behaves similarly to two queue/server pairs, one for each direction of transfer. The model of a channel includes two such pairs with one minor modification: The two servers have the same transfer rate (modeled using dataset QDSRATE). This change is not *necessary*, but makes the model more realistic and easier to alter.

Results of the simulation

The individual models described above were mapped onto individual structural modules using the GMB Translator and the PLIP Processor. The individual GMB models were composed into a single large model using the GMB Linker. The composed model was then used by the GMB Simulator to generate simulation results.

The SARA simulator was directed to measure queue lengths, queueing delays, processor utilization and processor throughput for every node and channel in the network. The measurements were compiled every 200 messages and a confidence level of 90% was required. The simulation had to be run for 1800 simulation seconds (approximately 3800 messages) in order to achieve the required confidence level with reasonable accuracy of the measurements (variation of $\pm$ 7%).

The SARA simulation results provided most of the required measurements: SARA provided processor utilization, throughput, queue length and queueing delay measures, but did not provide transit time measures. The confidence interval measures available from SARA were found to be very useful in light of the fact that earlier simulations (in other languages) were run for only 300 simulation seconds and 900 simulation seconds. The appendix of the paper shows an edited transcript of a SARA session.

Enhancing the SARA model

The simulation runs described in section 1 vary in three important aspects:

1. Network topology.

2. Use of data compression.

3. Channel capacities.

Only three topologies are required for the seven runs. Network topology is the most difficult aspect of the SARA model to change because it requires changing the structural model completely. There is no simple way to change the structural model, even if the model is highly repetitive. Topological changes also require the following minor modifications to the behaviroal (GMB) models:

a. The message generator must be changed to distribute messages to more T-nodes.

b. Two more models of switches must be developed to model switches which have three or five ports.

c. The function containing the routing table must be altered.

Changes in data compression can be dealt with at simulation time through the modification of datasets in the models of the message generator and the CP model. Changes in channel capacities can also be dealt with at simulation time through changes to datasets in each channel module. The seven simulation runs require three simulation models to be built (one for each topology). All remaining runs require only minor, simulation-time, changes and require no re-compilation.

Transit Time Measures

Transit times (from source to destination) can be measured by altering the models of switches, processors and the central processor. The selected problem *requires* that transit times for data messages addressed to T-nodes be measured at the destination switch. Transit times for data messages must be accumulated at the destination processor and central processor. Transit times for display requests must be measured at the *originating* switch. Such model-specific measures cannot be *automatically* supported by any simulation system. SARA can support these measures if the following changes are made to the models:

1. Switches must be modified to time-stamp messages arriving from attached terminal clusters and processors.

2. Switches must be modified to record transit times for messages addressed to attached terminals.

3. The central processor must be modified to propagate time-stamps of display requests onto the display output messages. In this way, the transit-time for display requests, as measured at the destination switch, will include the round-trip transit time.

4. The central processor must be modified to record transit times for incoming data messages.

5. Processors are expected to record transit times for all incoming messages.

6. Switches must be altered to record transit-times for display output messages addressed to attached processors.

The modifications above involve changes in the interpretation domain only (with only one minor change to the datagraph of a switch). Time-stamping messages is easily accomplished by adding a node at the input of each queue to read the simulation clock and record its value as part of the message. Recording of transit-time statistics requires some processing to be added to the *server* node in processors and switches.

Partitioning of Messages

The GMB allows for modeling of message partitioning in a straightforward manner. Only the behavioral models of terminal clusters and the central processor are affected. Partitioning can be accomplished by adding a loop in the interpretation of each terminal cluster and in the central processor. The number of iterations in the loop determines the number of control tokens inserted in the control arc queue and the number of data values written into the dataset queue.

One minor complication is introduced when transit time measures are combined with message partitioning: Each message must be tagged to indicate whether it is a complete message or a message partial. This can be accomplished easily by adding a one-bit flag to the last block of each partitioned message. Only the last block is time-stamped and used in transit-time measures.

Storage utilization

The problem required measurement of memory utilization within each switch and processor. Obtaining this measure is very difficult in SARA (and was found to be difficult in earlier simulation efforts as well). While the simulator can monitor items as they enter a queue (in this case messages), it is not possible to monitor attributes of those items. The desired measurement requires that the "available storage" in each node be decremented by the length of the message when that message enters the queue, and be incremented by the same amount when the message is removed from the queue. No mechanisms exist in SARA to perform such measurements short of modeling the details of each queue (resulting in an increase in the size of the model by a factor of 3 to 4).

4. Evaluation of SARA

Below, SARA is evaluated according to three important criteria:

1. The technical expertise required to use the system. Issues of user-friendliness are also included in in this discussion.

2. The ability of the system to accurately and easily model the class of computer systems of interest (in this example, packet-switched long-haul computer networks).

3. The support provided to the user in collecting performance measures and in generating useful reports.

Required Expertise

SARA requires a good deal of programming expertise. With PL/I as the interpretation language for the GMB, it is difficult to avoid programming. Because no graphics interface is currently available for SARA, model construction is time consuming. It should be noted, however, that once the basic *building blocks* of the model were constructed, the task of assembling and executing a model in SARA became considerably easier.

SARA is interactive in nature and attempts to be user-friendly. SARA attributes most of its user-friendliness to integral help facilities [FenR 80] which provide uniform on-line access to all user manuals. SARA's innovative help system and its flexible I/O interface is a major benefit. SARA does, however, suffer greatly from the absence of a graphics interface. Structural models, in particular, are very lengthy and could be made considerably simpler with a graphics I/O package. The descriptions of behavioral models, while not excessively bulky, could be simplified through the use of graphical interfaces. The advantage of the textual descriptions in SARA is that they permit the use of "dumb" terminals over ARPANET, a major goal in the original design.

Modeling Power

The computer network being modeled can be easily represented as a network of queues. Such problems map easily to the GMB and can be modeled very concisely †. This is the direct result of recent extensions to SARA by M. Vernon and E. e Silva [VerM 83a]. Control arcs and nodes have been extended to model queues and servers in a very natural way. For this reason the GMB was found to be ideally suited for this application.

The structural model in SARA (SL/1) provides a good mechanism for partitioning the problem. Mapping of behavioral models to structural modules is easy, and supports the concept of "Building Block Models". However, as currently implemented, the system requires compilation of several copies of a behavioral model if it is mapped to several structures.

---

† The reader should note that the largest model presented above contains only four control nodes.

The major difficulty in using SARA lies in the fact that the models (structural and behavioral) are static, i.e. their topology remains constant. As a result, structural changes to the system being modeled are sometimes difficult to handle. For example, a small change in the network topology manifests itself as a major change to the structural model. Simple structural changes to individual modules (e.g. increasing the number of channels connected to a switch) require significant changes to the control graph, data graph and interpretation. However, non-structural changes (such as changing the transmission rate) to the model are handled elegantly in SARA. Simulation-time commands allow the user to modify such parameters, even in the middle of a simulation experiment.

SARA also places emphasis on a clean separation between the system being modeled and the environment with which it interacts. SARA uses the same primitives to model both the system and the environment, but insists on a structural partitioning which clearly separates the two. The main justification for using the same primitives to model the system and its environment, is the desirability of taking a model of the environment to implementation. This would permit a "test environment" to be implemented and used to determine whether the implementation of a system is consistent with its model [RVE 79].

Measurement and Report Generation Capabilities

SARA provides capabilities to capture most of the required measures. The SARA simulation environment has been recently altered to provide a great deal of assistance in performing queueing measurements. SARA provides most of the classical queueing measures such as queue length, queueing time, utilization and throughput. Model-specific measures are not provided by the system, but can often be added to the interpretation model. In some cases (e.g. storage utilization measure) extensive changes to the model are required to obtain desired measures.

A useful feature of the SARA simulator is the automatic calculation of confidence intervals. These measures are valuable in determining the accuracy of the simulation results. SARA's greatest deficiency in this area is its inability to present graphical plotting of measurement results. The ability to plot results during and following a simulation experiment, can be extremely valuable.

CONCLUSION

A network simulation study was used to evaluate the SARA simulation system. SARA was found to be well suited for modeling computer networks. Its' strengths are in a uniform and thorough help system, in a clean separation of structure from behavior and in its simulation and measurement facilities. SARA's greatest weaknesses were found to be its use of text, rather than graphics, for model construction and for displaying results. In the area of queueing measurement gathering, some more sophisticated measurement mechanisms were found to be needed.

Acknowledgments:

I would like to thank Dr. Mary Vernon for her advice during model construction and for her assistance in explaining the newest features of the SARA simulator. Her help and patience were appreciated.

References

[EstG 78] Estrin, G. "A Methodology for Design of Digital Systems - Supported by SARA at the Age of One," AFIPS, *Proceedings of the National Computer Conference*, 1978.

[FenR 80] Fenchel, R.S "Interactive Systems with Integral Help," Ph.D. Dissertation, Computer Science Department, University of California, Los Angeles, July 1980.

[GorG 78] Gordon, G., *System Simulation*, 2nd Edition, Englewood Cliffs, N.J.: Prentice-Hall Inc., 1978.

[PetJ 81] Peterson, J. *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Inc., 1981.

[RazR 80a] Razouk R. and G. Estrin "Validation of the X.21 Interface Specification Using SARA," *Proceedings of the* 1980 *Trends and Applications Symposium: Computer Network Protocols*, Gaithersburg, Maryland, May 1980.

[RazR 80b] Razouk R. and G. Estrin "Modeling and Verification of Communication Protocols in SARA: The X.21 Interface," *IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980, pp. 1038-1052.

[RVE 79] Razouk, R., M. Vernon, and G. Estrin, "Evaluation Methods in SARA - The Graph Model Simulator," *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, August 1979, pp. 189-206.

[VerM 82] Vernon, M. "Performance-Oriented Design of Distributed Computer Systems," Ph.D. Dissertation, Computer Science Dept., University of California, Los Angeles, 1982.

[VerM 83a] Vernon, M., and E. de Sousa e Silva, "Performance Evaluation of Asynchronous Concurrent Systems: The UCLA Graph Model of Behavior", *9th International Symposium on Computer Performance Modeling, Measurement, and Evaluation*, College Park, Maryland, May 25-27, 1983.

[VerM 83b] Vernon, M., and G. Estrin, "The UCLA Graph Model of Behavior: Support for Performance-Oriented Design", *Proceedings of IFIP* 10.1 *Working Conference on Methodology for Computer System Design*, Lille, France, September 1983.

APPENDIX
Transcript of Simulation

The following is an edited transcript of a SARA session. The transcript shows portions of SL/1, and GMB models (particularly of a channel module). The transcript also shows sample output from the simulator

```
ec > udd> SARA> SARA_system> ec> sara
SARA Selector  February 9, 1980
New or modified news:
  No news changes
Auto start in progress
> /* Specify a working library */
> &library netsim
Working library now > user_dir_dir> Hughes> Razouk>
SARA_library> netsim
> /* Enter the SL-1 description of the network */
> @sl1 15
SARA.Structure
SL1 Translator  September 21, 1980
> &input case3.sl1
Input from source > user_dir_dir> Hughes> Razouk>
SARA_library> netsim> case3.sl1 started
universe ( environment, system )
universe ( i3 : system - environment,
        i6 : system - environment,
        i7 : system - environment,
        i8 : system - environment,
        i9 : system - environment )
environment ( message_generator )
environment ( i3 : message_generator - $i3,
          i6 : message_generator - $i6,
          i7 : message_generator - $i7,
          i8 : message_generator - $i8,
          i9 : message_generator - $i9 )
          ●
          ●
> /* Store the model */
> @store case3
model stored under "case3".
> @end
End of SL1 Translator
SARA
> /* Now we can enter the GMB model for each module */
> @beh
SARA.Behavior
> @gmb
SARA.Behavior.GMB
> @transl 15
SARA.Behavior.GMB.Translator
GMB Translator  June 26, 1982
> @load_sl1_model case3
model loaded from "case3"
  creation date: 821228
          ●
          ●
> @system T_S3
> &input channel.gmb
Input from source > user_dir_dir> Hughes> Razouk>
SARA_library> netsim> channel.gmb started
@control_graph
  @nodes Q0nserver
  @arcs Q0queue, Q0caout
  Q0nserver ( Q0queue : Q0caout )
  Q0queue ($in0 : )
  Q0caout ( : $out0 )
@end

@data_graph
  @controlled_processors Q0pserver(Q0nserver)
  @arcs Q0dain,
        Q0daqo, Q0daout, Q0darate
  @datasets ]Q0dsqueue, Qdsrate

  Q0dain ( $in0 : Q0dsqueue )
  Q0daqo ( Q0dsqueue : Q0pserver )
  Q0daout ( Q0pserver : $out0 )
  Q0darate ( Qdsrate : Q0pserver )
@end
          ●
          ●
> @endsys
model stored under model name T_S3
model stored under "case3".
          ●
          ●
> /* We can now exit the GMB translator and
> /* prepare to compose the individual models into
> /* one single large model
> @end
percentage of gmbplx tables used =  1.6%
End of GMB Translator
SARA.Behavior.GMB
> @linker case3 15
SARA.Behavior.GMB.Linker
GMB Linker  November 27, 1979
model loaded from "case3"
  creation date: 821228
Current SL1 model: case3
          ●
          ●
> @compose universe
universe successfully linked
> @store_sl1 case3
model stored under "case3".
Current SL1 model: case3
> @store_gmb case3
Current GMB model: case3
> @end
End of GMB Linker
SARA.Behavior.GMB
> /* We can now use the PLIP processor to
> /* provide interpretation for the GMB models
> @plip case3 15 -x
SARA.Behavior.GMB.PLIP
GMB PL1 Preprocessor  February 21, 1980
model loaded from "case3"
  creation date: 821228
Current model:  case3
          ●
          ●
> @system T_S3
Current SL1 system:  universe.system.T_S3
> &input channel.plip
Input from source > user_dir_dir> Hughes> Razouk>
SARA_library> netsim> channel.plip started
@template (Q0dain, Q0daqo, Q0daout, Q1dain, Q1daqo,
Q1daout)
      1 queue,
        2 source fixed bin(15),
        2 destination fixed bin(15),
        2 type fixed bin(15),
        2 length float bin(53);
```

```
@template (Q0darate, Q1darate)
        rate float bin(53);

@processor Q0pserver;
  @read rate @from Q0darate;
  @read job @from Q0daqo;
  @delay = (job.length*8) / rate @sec;
  @write job @to Q0daout;
@endprocessor;

@processor Q1pserver;
  @read rate @from Q1darate;
  @read job @from Q1daqo;
  @delay = (job.length*8) / rate @sec;
  @write job @to Q1daout;
@endprocessor;

> @dataset Qdsrate @like rate @init (9600*12);
          ●
          ●
> @store
***  0 errors
***  0 warnings
Do you want to compile the PLIP output?> yes
PL/I compilation in progress
PL/I 27d
Current model:  case3
> @end
End of GMB PL1 Preprocessor
SARA.Behavior.GMB
> /* We can now run the simulation */
> @sim case3 15
SARA.Behavior.GMB.Simulator
GMB Simulator  August 19, 1982
Time scale not specified.  Default time scale used
model loaded from "case3"
  creation date: 821228
Warning: the mapping between control_nodes and processors
is incomplete.  Dummy processors with unit delays will be
used as needed.  The cumulative delay time for the simulation
may be misleading.
>
> /* Ask for queueing measures to be accumulated */
> @calculate_queueing_measures S3/queue:S3/nserver
> @cqm S6/queue:S6/nserver
> @cqm S7/queue:S7/nserver
> @cqm S3_S7/Q0queue:S3_S7/Q0nserver
> @cqm S3_S7/Q1queue:S3_S7/Q1nserver
> @cqm S6_S7/Q0queue:S6_S7/Q0nserver
> @cqm S6_S7/Q1queue:S6_S7/Q1nserver
> @cqm CP_S/Q0queue:CP_S/Q0nserver
> @cqm CP_S/Q1queue:CP_S/Q1nserver
> @cqm P8_S/Q0queue:P8_S/Q0nserver
> @cqm P8_S/Q1queue:P8_S/Q1nserver
> @cqm P9_S/Q0queue:P9_S/Q0nserver
> @cqm P9_S/Q1queue:P9_S/Q1nserver
> @cqm P8/careceive:P8/nserver
> @cqm P9/careceive:P9/nserver
> @cqm CP/queue:CP/nserver
> @set_qmeasure_jnterval 200:1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16
> /* set a breakpoint to observe the simulation */
> @break_time_jnt 300 sec
> @set_default_time_unit sec
> @start
          ●
          ●
break_time_jnt breakpoint
time = 1800.000000000 sec

> @display_avg_qmeasures *  /* display all performance
measures */
Q.N.  S. Utilization  Throughput  Q. Length    Q. Wait  Q.
 1
    1  7.50e-002  1.02e-009  2.35e-002  2.26e+007  1
   ±  1.02e-002  6.94e-011  6.09e-003  5.50e+006  ±
 2
    1  6.91e-002  9.76e-010  2.67e-002  2.69e+007  1
   ±  1.02e-002  6.74e-011  7.78e-003  7.91e+006  ±
 3
    1  1.74e-001  1.89e-009  1.38e-001  7.21e+007  1
   ±  1.36e-002  8.41e-011  3.00e-002  1.34e+007  ±
 4
    1  7.30e-002  3.77e-010  6.91e-003  1.74e+007  1
   ±  1.02e-002  2.47e-011  2.09e-003  4.87e+006  ±
 5
    1  5.92e-001  2.79e-010  1.72e+000  5.93e+009  1
   ±  8.76e-002  2.81e-011  1.09e+000  3.63e+009  ±
 6
    1  6.91e-002  3.58e-010  7.85e-003  2.17e+007  1
   ±  8.42e-003  2.80e-011  2.92e-003  7.80e+006  ±
 7
    1  5.41e-001  2.63e-010  8.87e-001  3.15e+009  1
   ±  8.26e-002  2.24e-011  3.98e-001  1.20e+009  ±
 8
    1  6.34e-002  5.42e-010  1.13e-002  2.02e+007  1
   ±  5.45e-003  3.02e-011  3.77e-003  6.08e+006  ±
 9
    1  8.35e-003  1.04e-009  6.74e-004  6.53e+005  1
   ±  6.82e-004  4.62e-011  1.66e-004  1.67e+005  ±
10
    1  1.61e-002  8.47e-011  1.25e-002  1.24e+008  1
   ±  6.86e-003  1.72e-011  6.53e-003  5.91e+007  ±
11
    1  2.37e-001  2.97e-010  3.09e-001  9.83e+008  1
   ±  5.29e-002  2.26e-011  1.56e-001  4.94e+008  ±
12
    1  1.74e-002  9.47e-011  1.30e-002  1.27e+008  1
   ±  4.76e-003  1.62e-011  4.22e-003  4.28e+007  ±
13
    1  2.37e-001  2.97e-010  2.58e-001  8.15e+008  1
   ±  4.61e-002  2.57e-011  1.25e-001  3.84e+008  ±
14
    1  7.57e-003  2.97e-010  1.42e-004  4.71e+005  1
   ±  9.45e-004  2.26e-011  1.47e-004  4.87e+005  ±
15
    1  7.39e-003  2.97e-010  2.72e-004  9.42e+005  1
   ±  8.42e-004  2.57e-011  1.66e-004  6.05e+005  ±
16
    1  2.47e-001  1.04e-009  2.55e-001  2.48e+008  1
   ±  2.23e-002  4.52e-011  8.01e-002  8.23e+007  ±
> @end
End of GMB Simulator
SARA.Behavior.GMB
> @quit
End of SARA Selector
```