Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

SIMULATOR FOR THE EVALUATION OF DISTRIBUTED NETWORK-SYSTEM PERFORMANCE

Toshio Komatsu
Yukihiro Nakamura

NTT Information and Communication Systems Laboratories
Nippon Telegraph And Telephone Corporation
1-2356 Take Yokosuka-shi Kanagawa 238-03 JAPAN

Junro Nose

NTT Software Corporation
4-40 Hon-cho Naka-ku
Yokohama-shi Kanagawa 231 JAPAN

ABSTRACT

The INSYDE simulator was developed as a tool for estimating the performance of large-scale, complex computer systems using graphical type figures to describe system operations. This paper provides an overview of an enhanced version of INSYDE that can be applied to distributed network systems. The model description method and a number of application examples are also presented. A major challenge in simulating distributed network systems that interconnect multiple computer systems over a network is how to describe the intricate, complex model conditions of computer systems including task processing and interrupts on the one hand, and network-related model conditions (e.g., specifying protocol, media access method, congestion control, broadcast communication, and so on) on the other in the most parsimonious fashion with the least amount of model description. In this work, the complex and intricate conditions of distributed network systems are simply described by creating a protocol media access method library, realizing packet splitting/merging and broadcast communication functions as nodes, adding system information that can be checked for congestion control, and multiplexing task processing.

1 INTRODUCTION

The simulator INSYDE was developed as a tool for evaluating the performance of large-scale, complex computer systems. The simulator has already been applied to estimate the performance of a number of actual system developed in-house, and the evaluation results were reflected in the system designs. Focusing on the flow of transaction processing, INSYDE describes system operations as processing flows in a flowchart-like fashion. Remaining open to modify and extend the model later, hardware and software resources conditions are separated out from the process flow and defined in table format. The descriptive power of an earlier version of INSYDE presented at the 1994 Winter Simulation Conference (See Komatsu, Nose, and Wakayama 1994) has been markedly enhanced by providing support for indirect indication of upper level equipment by files, introducing variables to node parameters, and by implementing definitions of variables in table format.

Now in our most recent work, we have extended INSYDE giving it the capability to very simply represent descriptions of complex and intricate conditions of distributed network systems.

One of the major difficulties of simulating distributed network systems that interconnect multiple computer systems across a network is how to represent intricate and complex computer model conditions (inter-task processing and interrupts) and network model conditions (specifying protocol, media access method, congestion control, broadcast communication, etc.) simply with the least amount of description. When computers are deployed in distributed topologies, this raises the issue that, even when the transaction flows are identical, if the processing computer systems are different, then a different number of process flows must be represented.

The extended version of INSYDE also represents the operations of distributed network systems as flowchart-like process flows, with the distinguishing characteristics of resources, number of distributed units, performance, and so on, represented in a graphical format and using tables. However, four extensions had to be implemented before network-specific conditions could be represented as process flows in the same way as computer systems:
(1) CSMA/CD(carrier sensitive multiple access with collision detection), FDDI(fiber distributed data interface), and other lower-layer protocol media access methods were organized as a library.
(2) Network-specific processes including packet splitting/merging and broadcast communication functions were defined as nodes.
(3) System information accessible within INSYDE was added to make it possible to describe congestion control, and the abilities to dynamically modify device

performance and to terminate transactions were implemented as nodes.
(4) Finally, the ability to represent buffer controls and other functions within devices was implemented.

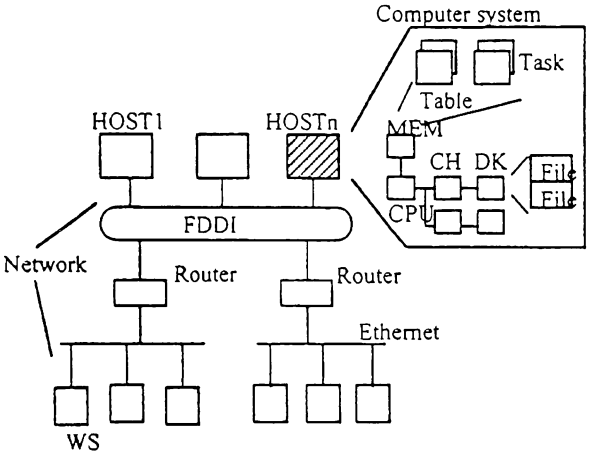
Moreover, in order for the new version of INSYDE to represent distributed topologies with relatively few process flows, inter-task processing and variable definition capabilities were extended. Finally, INSYDE was extended so it could represent various communication processes between computer systems. Implementing these various methods, we ended up with 73 types of nodes for representing process flows, and 8 types of definition tables, giving us the ability to simply describe the complex and intricate conditions associated with distributed network systems using a combination of process flows and definition tables.

In the network field, there are a number of fairly good tools available that are capable of representing the minute conditions of networks in graphical and table format, but when computer systems are factored in together with networks, it is exceedingly difficult to represent a detailed internal process model of a computer system (See Bharath and Kermani 1984, Marsan et al. 1990, and Dupuy et al. 1990). There are quite a number of simulation tools available in the computer field as well, but none of these are very good at representing some of the most characteristic processes of computers such as interrupts, inter-task processes, and so on. Of course, computer-specific tools are also incapable of representing network model conditions (See Okada and Tanaka 1991).

These were some of the considerations motivating our efforts to extend the INSYDE system platform. The rest of the paper will be organized as follows. Section 2 will outline some of the unique attributes of distributed network systems. Section 3 will lay out the issues involved in model description, which will be followed in Section 4 with the approach to model description that we have adopted in the extended version of INSYDE. Finally, Section 5 will present a number of application examples and the results that were achieved using the enhanced INSYDE simulator.

2 DISTINCTIVE FEATURES OF DISTRIBUTED NETWORK SYSTEMS

A typical distributed network system configuration is illustrated in Fig. 1. Let us briefly consider four distinctive features of this topology in terms of how these features affect simulation.
(1) Topologies in which multiple computer system are interconnected by a LAN generally include a range of devices including processors and file storage devices and various software resources such as tasks, tables, and files. In addition, the variety and number of these hardware and software elements is usually quite large.



FDDI : Fiber Distributed Data Interface
WS : Work Station
MEM : Main Memory
CPU : Central Processing Unit
CH : Channel Device
DK : Disk Device

Figure 1: Overview of Distributed Network System

(2) The transactions flowing through networks and computer systems assume many different varieties and are multiplexed. And even when the same transactions are being processed, the number of resources accessed and the amount of processing will vary depending on the nature of the transaction.
(3) There are also various distinctive kinds of processing associated with networks such as processing associated with protocols, routing, broadcast communications, congestion control, and so on. Other kinds of distinctive network-related processing support inter-center communications.
(4) There is distinctive processing in computer systems to handle interrupts, exclusive control, inter-task communications, and other functions.

3 MODEL DESCRIPTION REQUIREMENTS

Having looked at the distinctive attributes of network systems, here we will list some of the requirements in order for INSYDE to simply represent the intricacy and logical complexity of networks and distributed network systems.

3.1 Requirements to Represent Networks

(1) Integrate the network model descriptive method with that of the computer model.
(2) In order to reduce the number of processing flows, if

the processing flows are the same, make it possible to represent them as a single processing flow, even though the number of LAN devices being accessed and amount of processing may vary.
(3) Enable LAN-specific functions such as retry control after collisions, congestion control, and broadcast communication to be represented by process flows.
(4) Enable buffer control and other detailed device internal operations to be described.

3.2 Requirements to Represent Distributed Systems

(1) Integrate the distributed network model descriptive method with that of the computer model.
(2) If processing flows are the same, make it possible to represent those flows as a single processing flow, even though the hardware and software configurations making up the computer system are different.

Table 1: Functions of Nodes for Processing Flow Description

Type of Node		Number	Functions
Time delay		9	These delay transactions with resources in the reserved or unreserved status.
Control	Start, End	8	These indicate the beginning or end of processing flow. For termination,continuous restart and queuing to other processes is possible,except when the transaction is deleted.
	Branch	14	Unconditional branching and branching according to various conditions is possible. Conditions can be set for the work area of each transaction or variable,etc.
	Processing Transfer	9	These transfer processing between system processes and task processes or between task processes.
	Resource management	13	These handle resource reservation and release. Multiple resources can be reserved or released simultaneously.
	Splitting , Merging	8	These split transactions into multiple processes that may proceed in the same direction or in different directions. It is possible to specify that the split transaction be complete reintegrated or partially re-interred.
LAN		4	These split/merge packets, support broadcast communication,etc.
User		1	This is defined by the user.
Macro		2	This calls a pre-defined macro process flow.
Measurement		3	These collect statistical data over a freely specified measurement interval.
Connection Node		2	These indicate the processing flow interconnection on input screen.

4 DISTRIBUTED NETWORK SYSTEM MODEL DESCRIPTION METHOD

4.1 Modeling Overview

In order to integrate the representation of networks with that of computer systems, we focus on the transaction flow of system operations, and describe them as processing flows. Processing flows are described in two tiers: as system processing flows covering an entire process from the time a transaction is created until it is terminated, and as groups of task processing flows that execute individual work processes.

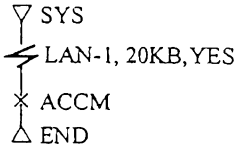
Processing flows are described through the use of 73 different kinds of nodes as shown in Table 1. In addition, to facilitate unified modification and correction of models, the full range of model conditions including load conditions, device conditions, task and other software conditions, and LAN conditions are defined using eight tables that encompass types of items.

4.2 Network Description Method

In the extended version of INSYDE, lower-layer protocol network descriptions are provided in the form of a library, while higher-layer protocol network descriptions are supported with additional functions enabling processing flows to be simply described.

4.2.1 Creating a Lower-Layer Library

The various functional capabilities of the protocol media access method including CSMA/CD, token passing, and FDDI will be provided in the form of a library. Figure 2 shows that the access method is not directly described in the LAN node, but rather is defined using a LAN definition table. The detailed parameter items will differ depending on the access method.



WW-XX,YY,ZZ: Media access and packet splitting
WW: LAN name
XX: LAN number
YY: Data length
ZZ: Specifies packet merging
× ACCM: Combines packet segments of the same generation

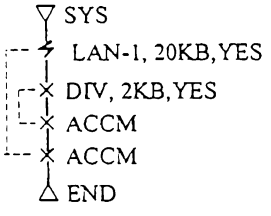
Figure 2: Example of Using a LAN Library

(1) CSMA/CD : The main defining parameters of the CSMA/CD method are slot size, frame interval, number of retransmissions, and cable length. When a frame is sent, frame collisions are detected. If a collision occurs, the frame is retransmitted a prescribed number of times according to the prescribed algorithm. If the number of retransmissions exceeds the prescribed value, statistical processing is invoked, the number of retransmissions is reset, and the frame is resent

(2) Token Passing : Both token bus and token ring topologies are supported by INSYDE's library. Here, the main defining parameters are THT (token-holding time) and the repeat extension bit. In order to allocate tokens in the proper order in the LAN interconnection node, the hierarchical position of the interconnection node must be specified. INSYDE approximately allocates tokens to interconnected nodes in the order in which LAN nodes are executed.
(3) FDDI : The token allocation rules and link delay time-related provisions are largely the same as in * for token passing. In addition, a TRT (token rotation time) value is employed to support both synchronous and asynchronous transmission. In order to achieve efficient transmission of files and other data, INSYDE provides that the token is not released until all the data have been transferred.

4.2.2 Higher-Layer Capabilities

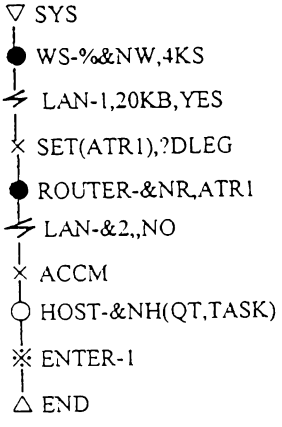
Higher-layer capabilities that can be represented in the extended version of INSYDE include the following.
(1) Packet Splitting/Merging : The LAN node not only has media access capability, that is also endowed with a packet splitting function. The DIV node is also supported that only does packet splitting. The user can then specify whether packets split by either node are to be merged by an ACCM node. Using these nodes, packet splitting and merging can thus be represented in nested fashion as illustrated in Fig. 3. Since generation management is imposed when packets are split, there is no need to describe merged packets at the ACCM node in terms of merge packet number and generation parameters.



× DIV,XX,YY:DIV, XX, YY: Divides a packet
XX: Packet partioning size
YY: Specifies packet merging

Figure 3: Example of Nested Packet Splitting/Merging

(2) Routing : In cases where the processing flow is the same (i.e., where the same kinds of devices are passed through the flow), Fig. 4 shows that the routing can be represented by a single process flow and variable definition tables even though the number of devices accessed and the amount of processing differs. One will observe in Fig. 4 that, in order to process split packets at a router, the current packet length for each transaction is set by the SET node.
(3)Broadcast Communication: Broadcast communications can be represented using the PARA, BROAD, and SPLT nodes, which correspond to different broadcast topologies. For example, where there are relatively few broadcast destinations, a topology can be represented with a branch splitting off at the PARA node. Or, in cases where there are many broadcast destinations and a simple flow is involved, the topology can be described with the BROAD node. In cases where processing will continue after the broadcast, a transaction that is first split and then continues in the same direction can be represented with the SPLT node.



Variable Definition Table

&NW	Probability	&NR
1	10	1
2	90	2
3	30	3

If &NW=1 at a 10% probability, &NR=1

× SET(ATR1),?DLEG :Set data length in the call work area
XX : Resource name
& : Display of variable
YY : Resource number
ZZ : Processing amount

● XX-&YY,ZZ : Time delay at the specified resource
ATR1 : Transaction work area
? DLEG : Function indicating packet length

○ WW-&XX(YY,ZZ) :Task processing request
WW : Host name
XX : Host number
YY : Task queue name
ZZ : Task name

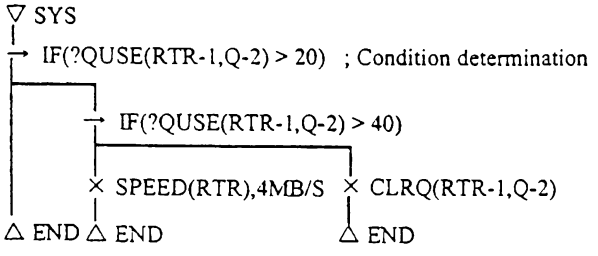
※ENTER-1 : Point of return from task processing

Figure 4: Example of Routing Description

(4) Congestion Control : In order to simply represent congestion control in a process flow, two additional things are required: system information that can be referenced, and nodes for terminating and dynamically modifying the performance of devices. These capabilities are illustrated in Fig. 5 where a stay situation is observed, device performance in incrementally modified, and the transaction is terminated.

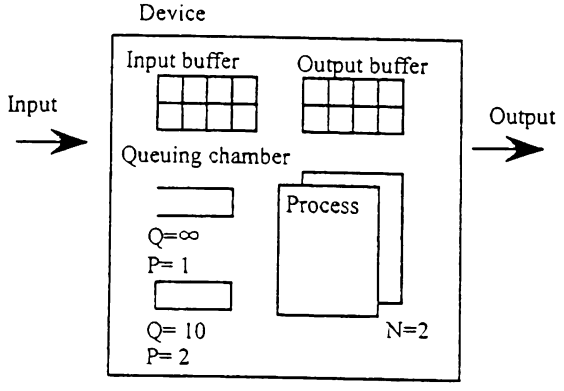
4.2.3 Defining Detailed Device Configurations

In the process of investigating a new protocol or configuration of a router, bridge, switch, or other piece of equipment, there are many cases where a researcher wants to model the internal aspects of a device in detail. Figure 6 illustrates that a range of device characteristics can be easily defined with a set of queuing compartments and corresponding device definition tables. The management of input and output buffers can then be represented using the tables.



?QUSE(XX,YY) : Function indicating use state
XX : Device name
YY : Device queue name
× SPEED(XX),YY : Modify device performance
XX : Device name
YY : Performance value
× CLRQ(XX,YY) : Reduce queuing transactions
XX : Device name
YY : Devise queue name

Figure 5: Example of Congestion Control Description



Q : Queuing capacity
P : Device acquisition priority
N : Degree of multiplexing

Figure 6: Example of Detailed Device Modeling

4.3 Distributed Network Description Method

In earlier work we significantly enhanced the descriptive power of INSYDE for modeling computer systems by indirect indication of upper level equipment by files, introducing variables as node parameters, defining variables in table format, and so on (See Komatsu, Nose, and Wakayama 1994). Building on this descriptive approach, we have extended the INSYDE environment to support multiple computer systems in distributed topologies as follows.

(1) Resource Identification Method

In the majority of cases where the computational load is distributed over a distributed computer system, each interconnected computer exhibits the same kind of processing flow. In order to reduce the number of processing flows in cases such as these, we have identified distributed equipment, tasks, tables, and other resources with a resource name and a resource number. Although the number of configured devices, the file configurations, and the number of tasks will vary for each computer system, and even though the processing computer systems will differ depending on the transaction, Fig. 7 shows that the processing flow can be represented by a single processing flow and set of variable definition tables.

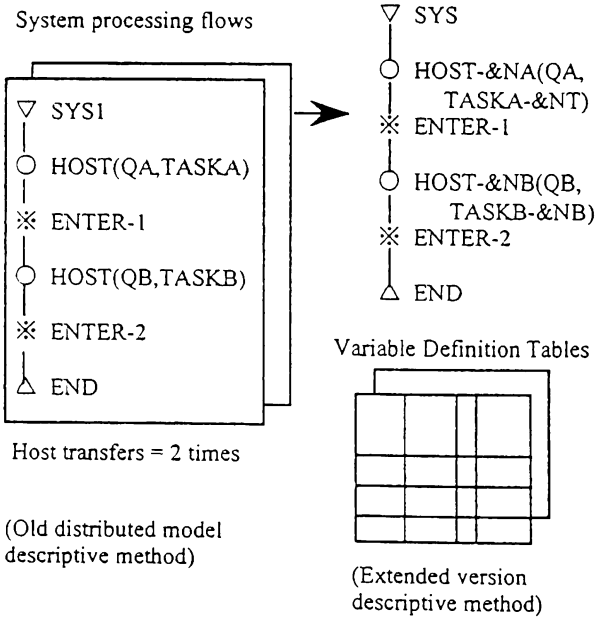


Figure 7: Reducing the Number of Processing Flows

(2) Communication Between Computer Systems

Since communications between computer systems involves communications moving in both directions, two types of communications are supported in the extended version of INSYDE. Cases where a response is required from the recipient of the communication and where no response is required are represented by LINK nodes and SEND nodes, respectively. Since each and every

communication is controlled by the initiator of the communication, there is no need for the response destination to be represented as a parameter in the LON node which indicates that a response has been returned to the communication initiator.

5 APPLICATION EXAMPLE

5.1 Model Conditions

We applied the extended version of INSYDE to a distributed network system having the same kind of topology as shown schematically in Fig. 1. The main conditions of the model were that the distributed system interconnected eight computers that were concurrently executing ten different kinds of jobs. Here we assume that the same kind of work would produce the same processing flow. Among the various kinds of work that were performed, the processing flow of one of the jobs is briefly described as follows.

A transaction was generated at a workstation. The transaction was divided into packets and sent via Ethernet, router, and FDDI to host computer system HOSTn where the packets were reassembled. The destination HOST varies depending on the nature of the transaction. Here n signifies the number of the HOST, and the transaction is processed by transferring the task from TASK1 to TASK2 at HOSTn. After the processing at HOSTn is completed, the packets are again split up and transferred to HOSTm over an FDDI link for further processing. Here, the value of m varies depending on HOSTn and the nature of the transaction. After the processing is completed, the transaction is terminated. The processing of each task involves CPU processing and file processing. Here, we have simplified the actual flow which was longer and more complex in order to facilitate the explanation.

5.2 Example of Model Description

As an example of the model description, portions of the system processing flow and task processing flow representing the job described in the previous section are shown in Fig. 8. In the workstation generating the transaction, since the device numbers will vary depending on the nature of the transaction, it is defined by variables. Descriptions are omitted, but since the HOST number, the task number, and so on are described using variables, the variables must be defined.

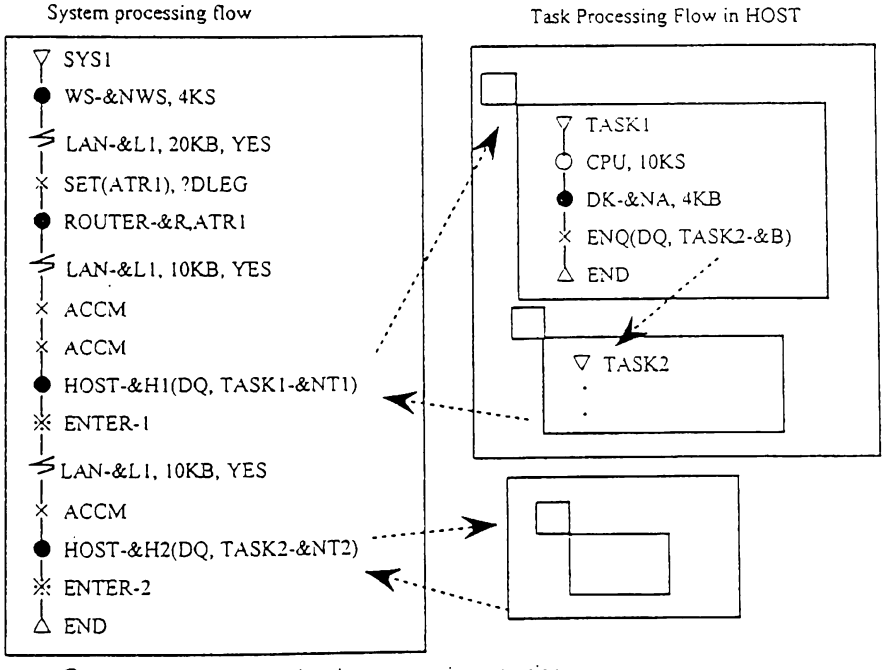


Figure 8: Example of Processing Flow Description

5.3 Application Results

Although we are not able to generalize since the results in any particular case will differ depending on the model conditions, as a broad indication we will compare results using the old version of INSYDE designed for computer systems and the new descriptive method outlined here in representing the work outlined above in Section 5.1.

5.3.1 Reducing the Processing Flow Number

Since there are two transfers to the host, the system process flow number described by the old version is given by the number of jobs × the number of hosts × the number of hosts, or 10 × 8 × 8 = 640. In the extended version of INSYDE on the other hand, since the host number is defined by variables, the transaction can be terminated with a system processing flow number equivalent to the job number of 10. Of course, the variable definitions have to be added to this. Since there are transfers of task processes, the task processing flow number using the old version is given by the number of jobs × the number of hosts × the number of hosts × the number of times the task processing is transferred = 1,280. In the expanded version, using variable definitions as in the previous example, the transaction can be terminated with a task processing flow number equal to the number of jobs × the number of times the task processing is transferred = 20. Although variables must be defined in order to reduce the processing flow number, and this increases the complexity somewhat, the variables are represented in a table format which reduces the complexity.

5.3.2 Number of Processing Flow Nodes

In the older version of INSYDE, packet splitting and merging must be represented using combinations of existing nodes. In cases where the processing amount is adversely affected by variables, programming is required to effect the splitting, and this complicates the description. Representing the protocol media access method also becomes more difficult. In the extended version of INSYDE, however, these various aspects can be represented using a small number of nodes; namely, LAN, DIV, and ACCM nodes.

5.3.3 Easy Visualization of Processing Flows

Although variable definitions are added in the extended version of INSYDE, there are fewer flows and fewer nodes per processing flow, which makes the extended version process flows much easier for users to visualize.

6 CONCLUSIONS

This paper has presented an overview of INSYDE, a powerful simulation tool for evaluating the performance of large-scale, complex distributed network systems using graphical figures and tables to represent system operations. We have also discussed the model description method and an application example. The extended version of INSYDE greatly simplifies the representation and the amount of description that is required to represent the complex and intricate conditions of distributed network systems by creating a protocol media access method library, realizing packet splitting/merging and broadcast communication functions as nodes, adding system information that can be referenced for congestion control, and multiplexing task processing.

The extended version of INSYDE not only provides a powerful tool for modeling and representing distributed network systems, it also features an extremely user-friendly man-machine interface. This user friendliness, for example, permits users to input other data outside of the actual numerical values and resource names with the click of a mouse, or to use the mouse to select a resource name that was previously defined as a table from a list.

We intend to follow up on this work by expanding the media access topology library and by developing other simple, intuitive simulation tools incorporating the powerful capabilities of INSYDE.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support and encouragement of their colleagues over the span of years that the INSYDE simulator was in development. The also express their appreciation to their coworkers for valuable discussions in formulating the model conditions of actual systems.

REFERENCES

BHARATH-KUMAR, K., and P.KERMANI. 1984. Performance Evaluation Tool(PET): An Analysis Tool for Computer Communication Networks, *IEEE JASC, Vol.SAC2, No.1*: 220-225.

DUPUY,A., J.SCHWARTZ, Y.YEMINI, and D.BACON. 1990. NEST:A Network Simulation and Prototyping Testbed, *COMMUNICATION OF THE ACM, Vol.33, No.10*: 64-74.

Komatsu,T., J.Nose, and H.Wakayama. 1994. Model Description in the INSYDE Simulator for Evaluating Large-Scale Computer System Performance, *Proceedings of the 1994 Winter Simulation Conference*: 1272-1289.

MARSAN, M.A., G.BALBO, G.BRUNO, and F.NERI. 1990. TOPNET:A Tool for the visual Simulation of Communication Networks, *IEEE JSAC, Vol.8, No.9*: 1735-1747.

Okada, Y., and Y.Tanaka. 1991. FES:A Toolkit System for the Development of Visual Interactive Simulators, *Trans.IPG Japan, Vol.32, No.6*: 766-776.

AUTHOR BIOGRAPHIES

TOSHIO KOMATSU is a Senior Research Engineer in the NTT Information and Communication Systems Laboratories. He received B.S. degree in 1972 and M.S. in 1974 in electrical engineering from Kyushu Institute of Technology. Since joining NTT in 1974, he has been active in R & D on computer systems hardware architecture and computer systems performance evaluation techniques. He is currently interested especially in modelling and analyzing techniques and tools for distributed computing systems. He is a member of the Information Processing Society of Japan.

JUNRO NOSE is a Senior Manager in the Productivity Innovation Promotion Headquarters at NTT Software Corporation. He received B.S. degree in 1967 and M.S. in 1969 in electrical engineering from Kobe University. He joined NTT in 1969 and has been active in R & D on fault diagnosis technique, videotex communication system and systems evaluation techniques. He joined NTT Software Corporation in 1993 and is now responsible in the systems evaluation systems and services. He is a member of the Information Processing Society of Japan.

YUKIHIKO NAKAMURA is an Excutive Manager in the NTT Information and Communication Systems Laboratories. He received the B.S., M.S., and Ph.D. degrees in Applied Mathematics and Physics from Kyoto Univercity in 1967, 1969 and 1995 respectively. He joined NTT in 1969, from 1969 to 1980, he was engaged in R & D on computer systems hardware architecture, and since 1981 he has been engaged in R&D of high-level synthesis technology for parallel architecture design. He is a member of the technical committee of the ICCAD, ED&TC, ISSS and ASP-DAC.