CITI Technical Report 91–7

NESTMOD: The NetMod–NEST Interface

U. Amin
aminu@citi.umich.edu

D. Bachmann
dave@citi.umich.edu

K. Deboo
deboo@citi.umich.edu

T. Teorey
teorey@citi.umich.edu

ABSTRACT

NESTMOD is a combined analytical modeling and simulation tool, based on the existing tools NetMod and NEST. It provides both transient and steady-state response statistics from models of interconnected local area networks that can execute at any level of detail desired. This gives users the potential to model both networks of extremely large scope (hundreds of thousands of nodes), and to look at great detail for any combination of the ISO layers. This paper describes the interface implementation, and presents an example to illustrate the potential power of the tool.

June 1991

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

NESTMOD: The NetMod–NEST Interface

U. Amin, D. Bachmann, K. Deboo, T. Teorey

June 1991

1. Introduction

Both analytical modeling and feel-time simulation may be useful when designing and evaluating computer networks. The goal of the NESTMOD project was to create a tool that would permit simultaneous performance and simulation modeling and analysis of network behavior, including the synthesis of appropriate network control strategies using feedback. As a starting point, we used existing components from NetMod (The University of Michigan) and NEST (Columbia University). The dynamic behavior capability of NEST, together with the steady state analysis capability of NetMod, should provide a powerful tool for both large-scale and detailed performance analysis of individual LANs and components. Such a tool would be able to analyze large systems at the overall level and provide details where required, as well as acting as a valuable addition to a network management facility. The resulting combined tool has been named NESTMOD.

The NetMod tool is a collection of simple analytical (queuing) models, used to increase understanding of the performance of new network technologies in university or industrial network environments [1,2,6,7,8]. It is currently implemented in HyperCard for the Macintosh, and is under development for MS Windows 3.0 for the PS/2 and for the UNIX environment. Its principal application is to assist the network designer configure a potential user's data network, using proposed network hardware and software components. It can also perform network evaluation and point out bottlenecks. However, for network and, transport layer details, such as flow control, it is better to simulate the network at a greater level of detail.

The NEST tool is a collection of routines that can be used for simulation of arbitrary network topology and communications characteristics [5]. Only point-to-point corrections can be specified; consequently, to simulate a real network topology a significant number of detailed routines must be provided.

NetMod models steady state behavior, allowing the designer to rapidly weed out potential designs to a produce manageable set, while NEST allows the designer to compare that set in much greater detail and to look at transient behavior. When integrated, the two approaches provide a powerful tool for both large and detailed individual device performance.

As a scenario, a user can call NetMod to set up a blank window, and use the icon menus to build a simulated network configuration with LANs, network interconnect devices, and workstations. When large-scale systems are to be analyzed, the analytic model is executed; when the details of an individual LAN or workstation are to be examined, the NEST simulator is called and the performance statistics are displayed on the screen, overlaying the network configuration. Both steady state and transient behavior can be displayed, and sensitive analysis quickly performed with simple user interactions.

A library of NEST routines is being developed to simulate the higher abstractions of local area networks and other intermediate devices, each of which is represented by a single submodel (an icon on the screen) in NetMod. Each of these higher abstractions is simulated with more granularity at a lower level of abstraction within NEST. More computation-intensive simulations can be done on some powerful machine, while most of the modeling could be done in a user-friendly environment (e.g., HyperCard). An interface has been developed to pass the network model represented by NetMod to NEST along with the library functions.

Thus, there are two main parts to the project (see Figure 1):

1. Interpret a network model from NetMod, combine it with the required library routines, and pass it on to NEST (Analysis Server; get back the results and pass them back to NetMod after reformatting.
2. Create the library of routines for the higher abstractions of the local area network and other devices. (Library Routines).

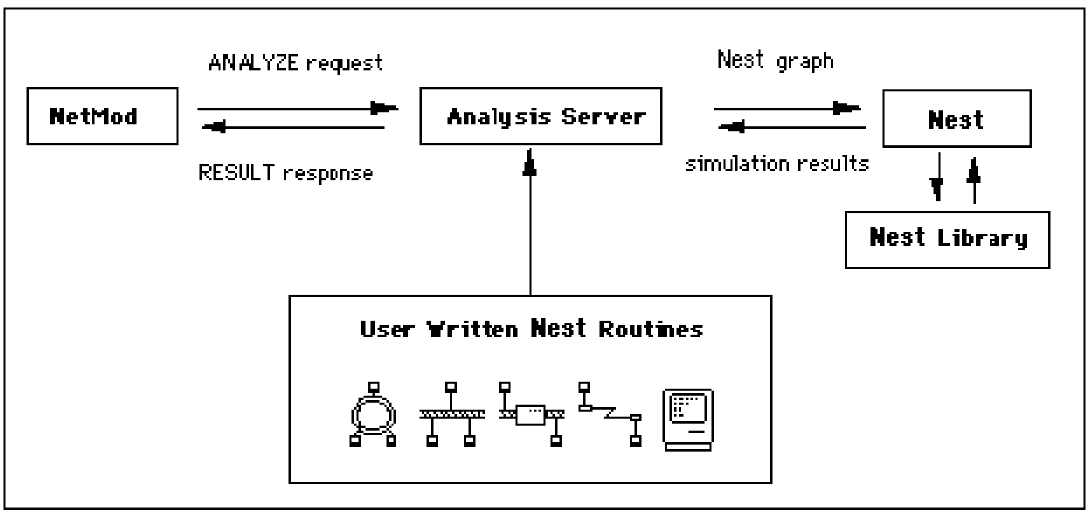

Figure 1. Overview of NetMod-NEST interface

2. NetMod-NEST Interface Component

2.1 NetMod

NetMod analyzes an existing or proposed network in terms of its basic performance characteristics (e.g., component utilization and packet delay times), and supports a thrifty analysis of performance based on changes in the workload parameters and other small changes in the system specifications. It also analyzes the efforts of a major reconfiguration on performance. Comparisons of alternate configurations will be possible as part of the network design process. Currently, the tool provides models of the most popular local network technologies, such as routers ring and Ethernet and their various, and special network components such as routers, bridges, and gateways. Workloads are specified by designating user types at the individual or aggregate workstations, personal computers, or mainframes. Each user type is described by a default set of applications (e.g., a secretary is estimated to spend a certain percentage of time using word processors, spreadsheets, databases, electronic mail, etc.). This data can be customized by the user of the tool [6].

When the user requests that a network model be simulated, NetMod opens a TCP connection to the appropriate host and port (specified by the user), and sends the ANALYZE request across that connection, traversing the model and generating a design for each submodel. It then adds that connection to the list of connections to monitor. Whenever the user is not doing something, NetMod checks its list of connections to see if any has incoming data. If one does, and it is a RESULT response, NetMod reads the response to find out which model the data pertains to, and goes to that model. It then parses through the response, and puts the results for each submodel into the corresponding display field. Whenever a connection is closed by the host, NetMod removes it from its list of outstanding connections and disposes of it. Currently, NetMod only closes a connection if it gets a serious error on that connection.

2.2 NEST

NEST (Network Simulation Testbed) is a tool for simulating and prototyping distributed algorithms and systems. NEST allows the simulation of arbitrary network topologies and communication characteristics.

NEST is provided as a C library of functions which are linked together with the users code. There are no built-in functions for statistics gathering, but since the user can easily propagate the functions that run on the nodes and pass data over links, relevant statistics can be easily gathered.

The entire simulation runs within a single UNIX process. NEST schedules the associated with the nodes in the network and ensures that messages arrive in the correct order in simulated time. Due to the usage of a single UNIX process, it is possible to simulate systems with large numbers of individual entities.

2.3 Analysis Server

Communication between NetMod and NEST is done through a program called Analysis Server, which talks the network model from NetMod and ensures a suitable model for simulation by NEST. It also communicates with NEST and submits the model for analysis; once the analysis is done, it gets the results from NEST. These results are processed and sent back to NetMod in be required format. See Figure 1.

Within the Analysis Server, the following steps are performed:

1. Parse the model given by NetMod (ANALYZE request)
2. Determine which library routines are required for this model
3. Create the Graph structure, as required by NEST
4. Attach appropriate library functions to appropriate nodes
5. Submit the model to NEST for simulation (NEST graph).

Once the simulation is complete (i.e., the time allowed for simulation to run is over or a specified number of passes are completed), the following steps are performed:

1. Get results from NEST (simulation results)
2. Format them according to the requirement of NetMod
3. Send them to NetMod (RESULTS response).

2.4 Links

1. The ANALYZE request brings the data from NetMod to the Analysis Server, by physically taking it from the computer running NetMod to some UNIX workstation for simulation over a network connection. Table Al in Appendix A describes the format of this request, which provides the information required to simulate any given submodel of the network.
2. The NEST graph is the means by which the model is supplied to NEST for simulation. The Analysis Server constructs a NEST graph structure that represents the given Network. The main information provided is the number of nodes; their connection, depending on the topology; and the library functions required to simulate them. A brief description of the Nest graph structure is given in Table A4.
3. The simulation results are the means by which the results of the simulations are collected. Since the Analysis Server and Nest routine are combined and run as a single process, the results are obtained from shared global variables. No real transformation of data is done in this link: it is displayed purely for conceptual understanding.
4. The RESULT response sends results back to NetMod over the network connection using the format shown in Table A5. It differs from the ANALYZE request primarily in the replacement of submodel information by simulation results, i.e. the Utilization, Delay, and Queue Length for each model. Keywords such as “UTIL,” “DELAY” etc. are used to pass back the results, rather than making them position-dependent, so that any other type of required result can easily be returned in the future, simply by defining new keywords.

2.5 Library Routines

The network is represented by higher abstractions in NetMod. The Analysis Server is provided for this network model. However, NEST could simulate only point-to-point connections. Hence, for each higher abstraction of network represented in NetMod, some routines must be made to simulate the higher abstraction model of the network in NEST.

Following is the guideline for writing such a routine for any higher abstract model of the network:

- Create the topology of the model:
  1. Create the required number of nodes (depending on the model).
  2. Create edges among the nodes, according to the structure of the model (star, node, fully connected etc.).
- Attach appropriate functions from the library to the nodes:
  3. Attach a node routine to each of the newly created nodes.
  4. Attach an edge routine (called a <u>channel function</u> in NEST) to each of the newly created edges.
- Other details:
  5. Provide a mechanism for appropriately (according to parameters provided by NetMod) generating traffic.
  6. Provide default values for the diffaent parameters required by NEST.

Whenever a new type of model is to be simulated, only the library routines required to attach to the nodes and the edges are to be added to the library; also the appropriate default values for the parameters.

Following is an example of one such model.

3. Example of a Library Routine–Token Ring

The following routine is to be used to create each token ring present in the network being modeled.

1. Create a number of workstations on the ring. For each workstation, create a workstation node (NEST node).
2. Make links between successive nodes, and create a link between the last and the first nodes, to form a ring structure.
3. Attach a node routine to each node.
4. Attach an edge routine to each newly created edge, depending on the reliability of the ring.
5. For each workstation, create a dummy node which simulates generation of traffic at that workstation. Connect this dummy node to the workstation node. Attach the dummy node routine to this node.
6. Provide NEST default parameters such as delay time, reliability, utilization, etc., either from parameters provided by NetMod, or some generic value.

The routine for each workstation node (the node function that is to be kept in the library) does the following:

- Creates its own node id
- Waits until there is no message
- Receives a message
    - If the message is from a dummy node, it keeps it in a queue.
    - If the message is a frame addressed to itself, it receives the message and forwards the token.
    - If the message is a frame addressed to another node, it passes the flame to the next node.
    - If the message is a token, it checks to see if there are any messages in its queue. If there is one, it sends it; if the message is greater than the frame size, it sends part of the message.

The routine far each dummy node (Message Generator) does the following:

- Generates a message for the workstation node
- Sleeps for specific time. This will determine the traffic on the ring.
- Repeats the loop.

4. Example of NetMod-NEST

In this example, we create submodels for an IBM Token Ring and 10 attached PCs. After setting the relevant parameters for the PC submodel (user type, number of PCs, % active), we choose “Other Analysis...” from the Options menu.

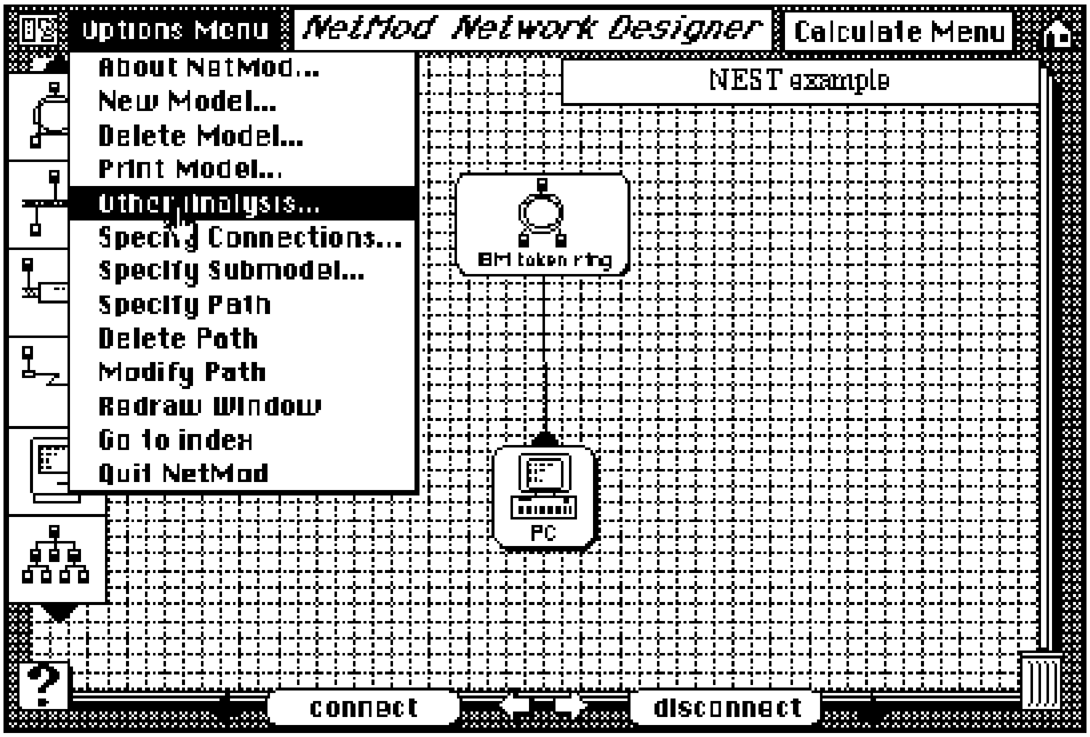

Figure 2. Requesting Model Simulation

This takes us to the Analysis card, where we click the NEST button and set the Host to citi.umich.edu and port to 3401, where the Analysis Server is listening. We then set the number of passes of the simulation. Clicking on the OK button starts transmission (the Cancel button would just return us to the model card).

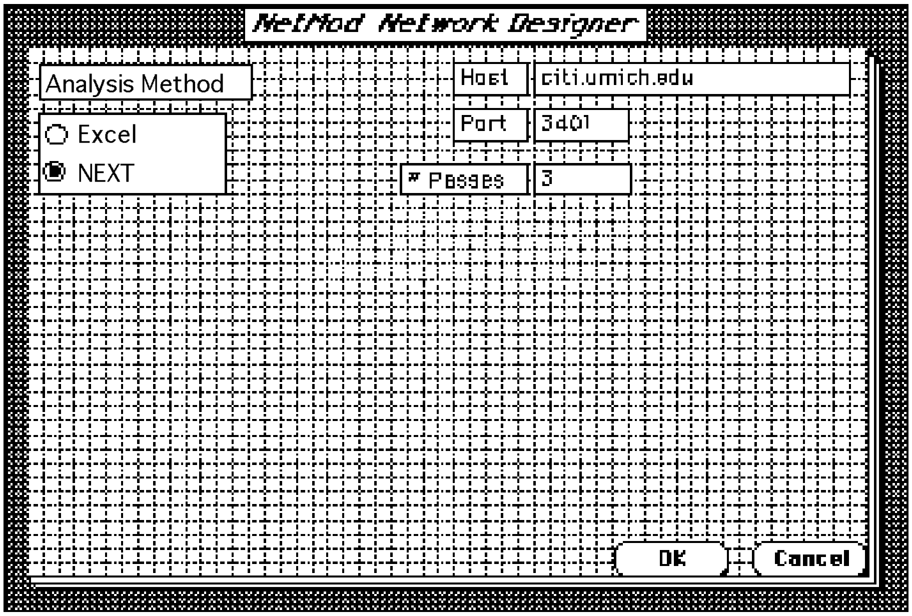

Figure 3. Specifying Analysis Parameters

NetMod now opens a TCP connection to the Analysis server, and sends the description.

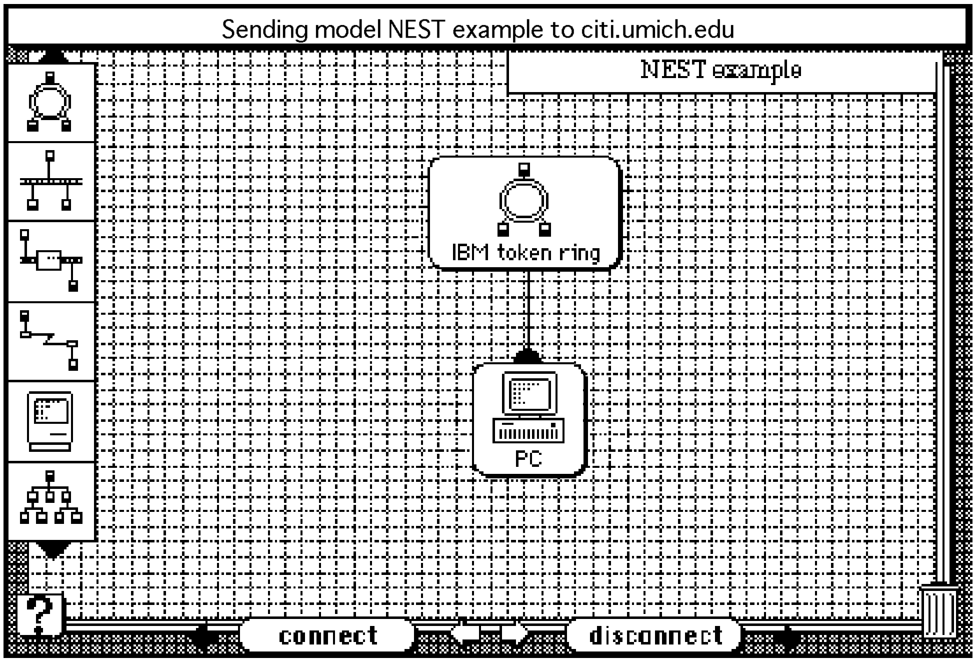

Figure 4. Sending the Request

```
ANALYZE
{
NEST
PASSES 3

MODEL NEST Example
{
card id 9060

SUBMODEL IBM token ring
{
231,120
Icon ID: 18155 “Ring”

PC,2963,
IBM token ring
4000000


1

201

0
}

SUBMODEL PC
{
230,198
Icon ID: 31638 “PC”
IBM token ring,2963,

workstation
34.96
800,1280000

10,

Student-engineering & science
100,

PC & Token Ring
0
}
}
}
```

Table 1. Analysis Request Sent

While NEST is simulating the model, we are able to continue working either with NetMod, or within any of the models, or to create new ones. We can also request more simulations to be run either on the same or on other hosts if they are available. NetMod keeps track of all outstanding analysis requests, and takes us back to the approbriate card when the results are returned for a particular model. As the results come back, they are put into the results fields of the model.

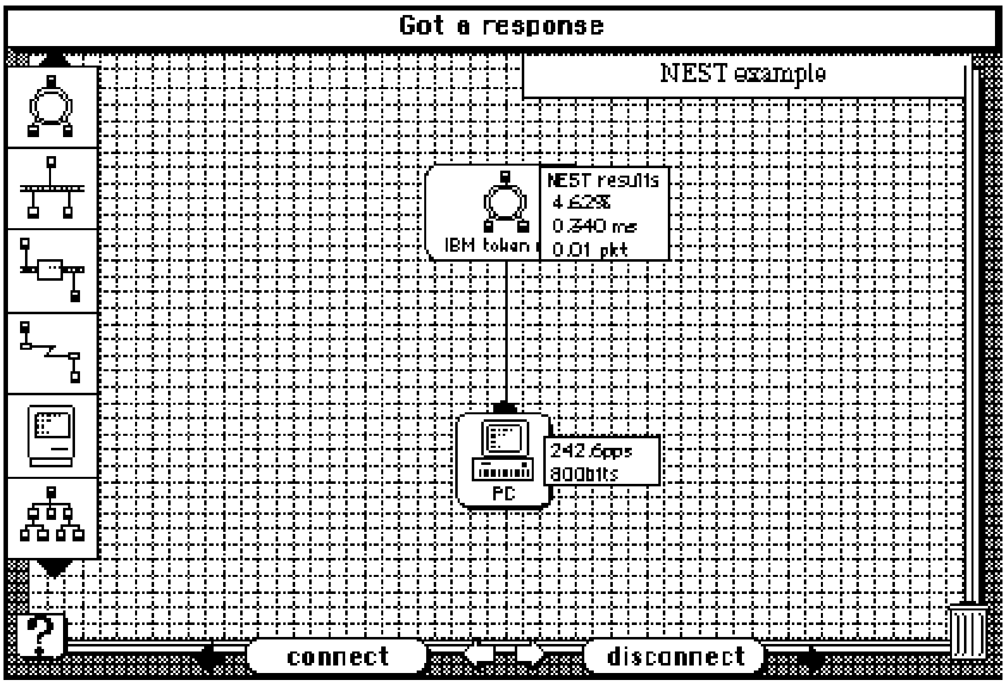

Figure 5. Returning Results to NetMod

```
RESULT
{
NEST
PASSES 3

MODEL NEST Example
{
card id 9060

SUBMODEL IBM token ring
{
UTIL 0.064516
DELAY 616236
QUEUE 1.566790
}
}
}
```

Table 2. Table 2 - Analysis results returned

5. Current Status and Future Directions

The Analysis Server and the library routines for token rings and Ethernet have already been implemented, and are successfully working; library routines for other models, such as routers, are currently under development. The NetMod tool is capable of submitting a model to the analysis Server for simulation. This tool can keep track of multiple outstanding simulation requests, so it is currently possible to submit simulation to several analysis servers (possibly on different machines) for concurrent execution. As each set of results comes back, NetMod will go to the appropriate model and fill in the results on the screen. This allows the user to create models for several alternate scenarios, submit them all for simulation on the same or different boosts, and go out to dinner. The results will be put into each model as they arrive, and the user may compare them at his or her leisure. This provides a simple means for analysts to harness the power of all the available machines on a network Because each analysis request uses a different port on the Mac, the Analysis Server has no difficulty distinguishing between multiple requests from the same machine.

Another enhancement is in the direction of greater functionality. Because NetMod uses simple analytic queuing models, it is only able to report steady-static (average) results. With NESTs simulation abilities, it is possible to display the dynamic behavior of the network being modeled. In the same way that users currently specify the number of passes of the simulation to perform (on the Analysis card), it will be possible to specify how often NEST is to return snapshots to NetMod. If a user requests a snapshot every two passes, NEST will return partial results back every other pass, and NetMod will display those results on the screen. Thus, the user will be able to watch the dynamic behavior of the network being studied.

One other enhancement will be made in the near future to the NetMod-NEST partnership. This is in the direction of greater flexbility, entailing mixing NetMod's analytic solutions with NEST's-simulations in the same model. The user will create a NetMod model as usual, but may mark some submodels for simulation rather than for analysis by formula Then, when NetMod is calculating traffic and results, when it reaches a simulation submodel it will submit an analysis request for that submodel to NEST and continue on. Each submodel will be tagged with the boost to send requests to, so several different simulations on different hosts might be requested by one network model. The user will see results immediately displayed for solved submodels; and as simulation results come back those will be displayed as well.

References

1. Bachmann, D.W., Srinivasan, M.M., and Teorey, TJ. "The Network Modeling Tool: A Design for Large-Scale Campus Networks," Proc. Ninth Annual Intl. Phoenix Conf. on Computers and Comnunications, lEEE, March 21-23, 1990, Phoenix, AZ, pp. 560-567.
2. Bachmann, D.W., Segal, ME., Srinivasan, M.M., and Teorey, TJ. "NetMod: A Design Tool for Large-Scale Heterogenoous Campus Networks," IEEE J. on Selected Areas in Communications (JSAC) 9,1 (January 1991), pp. 15-24.
3. Bachmann, D.W., Bauer, M., Bennett, M., Fasulo, G., Klinge, K, Makkapati, S., Kamlet, M., Slomin, J. and Teorcy, TJ. "Analysis of X.500 Distributed Refresh Strategies," Journal of Database Administration 2, 2 (Spring 1991), pp. 1-13.
4. Chiarawongse, J., Srinivasan, M.M., and Teorey, TJ. "A Simulation Model for a Large Interconnected Local Area Network Decomposition Techniques," IEEE Network 2,4 (July 1988), pp. 19-27. [Presented at the Symposium on the Simulation of Computer Networks, Colorado Springs, CO, Aug. 4-7,1987.]

5. Dupuy, A., Schwartz, I. Yemini, Y. and Bacon, D. "NEST: A Network Simulation and Prototyping Test-bed," Comm ACM 33,10 (October 1990), pp. 64-74.

6. Srinivasn, M.M. "An Approximation for Mean Waiting Times in Cyclic Server Systems with Nonex-haustive Service," Performance Evaluation 9 (1988), pp. 17-33.

7. Srinivasan, M.M., Bachmann, D.W., Teorey, TJ. and Chiarwongse, J. "Mean Waiting Times forCyclic Service Systems with Batch Poisson Arrivals: The Non-Exhaustive Service System," CITI Technical Report 89-6, The Univerisity of Michigan, Ann Arbor, Ml, July 1989.

8. Teorey, TJ., Bachmann, D.W.. and Srinivasan, M.M. "User Profile and Workload Analysis for Intercon-nected Local Area Networks," submitted to CACM, 1991.

About the Authors

Utpal Amin is a graduate student in the Electrical Engineering and Computer Science department of the University of Michigan.

David W. Bachmann is a Ph.D. candidate in the Electrical Engineering and Computer Science de-partment of the University of Michigan.

Kerman Deboo is a graduate student in the Electrical Engineaing and Computer Science depart-ment of the University of Michigan.

Toby J. Teorey is an Associate Professor in the Electrical Engineering and Computer Science de-partment of the University of Michigan.

IBM contact for this paper is Jacob Slonim, Center for Advanced Studies, IBM Canada Ltd., Dept. 61/894, 895 Don Mills Road, North York. Ontario M3C IW3 This paper is available as IBM CAN-ADA Technical Laboratory Report TR 74.064