UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

DTIC FILE COPY

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30 1986

DTIC ELECTE OCT 0 2 1989 B

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b RESTRICTIVE MARKINGS

2a SECURITY CLASSIFICATION AUTHORITY

DD FORM 254, DATED JUNE 1986

ULE

DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release
distribution unlimited

AD-A213 203

ER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
SRNTN 64

6a NAME OF PERFORMING ORGANIZATION	6b OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
ROCKWELL INTERNATIONAL COLLINS DEFENSE COMMUNICATIONS	M/S 460-340	DEFENSE ADVANCED RESEARCH PROJECTS AGENCY INFORMATION SCIENCES & TECHNOLOGIES OFFICE

6c. ADDRESS (City, State, and ZIP Code)	7b ADDRESS (City, State, and ZIP Code)
3200 E. RENNER ROAD RICHARDSON, TEXAS 75081-6209	1400 WILSON BOULEVARD ARLINGTON, VIRGINIA 22209-2308

8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
DARPA	ISTO	CONTRACT NO. N00140-87-C-8903

8c ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
1400 WILSON BOULEVARD ARLINGTON, VIRGINIA 22209-2308				

11. TITLE (Include Security Classification)
PC-NETSIM: A PC BASED NETWORK SIMULATION

12 PERSONAL AUTHOR(S)
YOUNG, DAVID AND BAUSBACHER, PETE

13a TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
SRNTN	FROM OCT 87 TO JUL 89	1989, JULY, 18	13

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	05.07		
25	02.05		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

PC-NETSIM is an IBM PC/XT/AT/ 386 based packet radio network simulation with a number of interesting and innovative features. Since the target system (the packet radio) has the same CPU as the PC, the simulation is able to run the target software without modification. This leads to a test environment which is so realistic that the communication protocols behave for the most part exactly as they would in a real network, even down to the level of processing delays and multiple access channel interference. It also provides a development environment that has many advantages over using real network hardware, such as network-wide state information during trace breakpoints, exact experiment repeatability, and a graphical representation which gives a very dynamic view of network variables. A description of the PC based real time operating system under which PC-NETSIM runs and the Ethernet based test-bed environment is also provided.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

89 10 2 135

UNCLASSIFIED

894207

PC-NETSIM: A PC Based Network Simulation

SRNTN 64

July 1989

Prepared by: C. David Young
Pete Bausbacher
Technical Staff Members
Advanced Technology and Engineering
Collins Defense Communications
Rockwell International

Prepared for: Major Mark Pullen
Major Brian Boesch
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

Mr. Paul Sass
US Army CECOM
Attention: AMSEL-RD-C3-AC-B
Ft. Monmouth, New Jersey 07703-5202

Sponsored By: Defense Advanced Research Projects Agency (DoD)
Information Sciences and Technologies Office
Survivable Adaptive Networks (SURAN) Program
ARPA Order No. 6067/9142
Issued by NRCC under Contract # N00140-87-C-8903

PC-NETSIM: A PC Based Network Simulation

David Young
Pete Bausbacher

ABSTRACT

PC-NETSIM is an IBM PC/XT/AT/386 based packet radio network simulation with a number of interesting and innovative features. Since the target system (the packet radio) has the same CPU as the PC (Intel 8086 family), the simulation is able to run the target software without modification. This leads to a test environment which is so realistic that the communication protocols behave for the most part exactly as they would in a real network, even down to the level of processing delays and multiple access channel interference. It also provides a development environment that has many advantages over using real network hardware, such as network-wide state information during trace breakpoints, exact experiment repeatability, and a graphical representation which gives a very dynamic view of network variables. A description of the PC based real time operating system under which PC-NETSIM runs and the Ethernet based testbed environment is also provided.

INTRODUCTION

PC-NETSIM is a quasi-identity simulation [1] made to run on an IBM PC/XT/AT or compatible. The term "quasi-identity" refers to the fact that only part of the system is simulated, whereas the other part consists of the real, unmodified target software. By realistically mimicking its execution environment, the software is bluffed into behaving as it would in a real system. Developed for DARPA (Defense Advanced Research Projects Agency) on the SURAN (Survivable Adaptive Networks) program, PC-NETSIM is used to test and characterize the protocols and peripheral network hardware in the absence of the target communications equipment, the LPR (Low-cost Packet Radio), whose delayed production schedule has put it several years behind the protocol development. Although the idea of developing PC-NETSIM was initially born out of necessity caused by this skewed schedule, the final product has proven to be immensely worthwhile, even if the LPR had come out early enough to be useful in the software development process. The reasons for this are fundamental to the whole concept of simulations, namely that they often provide certain advantages over their real counterparts, usually in terms of expense, control, and availability. PC-NETSIM embodies not only these characteristics, but also includes a number of innovative technical features which make it of interest to the software engineering community.

This paper will concern itself with those aspects of PC-NETSIM which are innovative and of general interest. Among the innovations is a

hierarchical approach to simulation design and a new technique for incorporating nodal processing delays into the simulation. Software designers will find the real time operating system and mouse/graphics driven user interface to be of interest. System engineers will discover how a PC based simulation can be effectively used in a distributed testbed environment connected via Ethernet. Also, a description of the CSMA (Carrier Sense Multiple Access) channel model taking into account the effects of multiple access interference, forward error correction, and wideband jamming will be of interest to communication engineers. It is hoped that the description of these and additional features will allow other engineers to recognize parts of PC-NETSIM which might be useful in their work.

SYSTEM ARCHITECTURE

PC-NETSIM consists of four primary subsystems in a layered architecture. Each subsystem will be described here.

SURAP - Survivable Radio Network Protocol

This is the "real" part of the PC-NETSIM quasi-identity simulation, namely, the actual communication protocol code. There is no difference between the SURAP that runs in the LPR and the SURAP that runs in PC-NETSIM. In fact, since PC-NETSIM is really just a means of running SURAP in a different environment, it is actually more of an emulation than a simulation at this level. This is probably the most important feature of PC-NETSIM from a protocol development point of view, which is why PC-NETSIM evolved in the first place. That is, if the same protocol code can run unmodified in either the simulated or target system, there will be fewer bugs and greater confidence that the protocols will work as expected in the target system.

LPROS - LPR Operating System

The "real" LPROS resides in the radio along with SURAP and provides operating system services such as scheduling and input/output to the radio hardware. The PC-NETSIM version of LPROS is for the most part the same, except that it has been modified where necessary to take into account the differences between the simulation and the LPR hardware. For instance, whereas the transmission of a packet by the radio is initiated by the LPROS through a complex series of DMA initiations and interrupts, the same thing is handled in the simulation with a pair of procedure calls, one to the simulation master to queue the transmission and one from the simulation master after the transmission is complete. Likewise, whereas in the LPR terminal i/o takes place via RS232 hardware, in PC-NETSIM this gets passed on to DOS. It will be revealed later that LPROS is actually a specialized version of the more general real time operating system, SOS, on top of which the entire simulation runs.

PC-NETSIM - Collection of Simulation Processes

PC-NETSIM is a collection of processes which run in the SOS environment and form the kernel of the simulation. One process manages communications with the testbed environment, another handles storage and retrieval of the connectivity and field view matrices from the disk and testbed controller, while the simulation master process (SIMMAN) manages the simulation clock and simulates the radio hardware and transmission channel.

Of the PC-NETSIM processes, SIMMAN is by far the biggest and most complex. Simulating the radio hardware means performing those tasks normally handled by the radio firmware and hardware, at least to the extent needed to fool the LPROS and SURAP into thinking they are talking to a real LPR. Simulating the transmission channel involves handling all packet i/o to and from the LPROS, performing CSMA, and detecting collisions. (A description of this process can be found in the section entitled "LINK PERFORMANCE MODEL".) Managing the simulation clock involves coordinating and synchronizing with the testbed environment, maintaining the clocks of the individual network nodes (see the section entitled "PC-NETSIM Clocks and Event Queues" for a detailed discussion), and slowing or stopping the simulation upon request of the user. In fact, as far as the average PC-NETSIM user is concerned, SIMMAN is the primary user interface since it handles most of the commands to control the simulation.

SOS - Scheduling Operating System

From a practical standpoint, one the most important features of PC-NETSIM for many software engineers is the general purpose operating system on which it is based. SOS is a multitasking, prioritized operating environment that allows real time processing on an IBM compatible PC/XT/AT or (with slight modification) in any Intel 80x86 based system. SOS provides services which are utilized by SOS and the applications running on top of SOS (e.g., PC-NETSIM) to schedule process execution, allocate resources, and perform I/O to the PC keyboard/screen, PC disk storage, COM 1 mouse interface, COM 2 modem port, and Ethernet drivers. It also has many program development tools including symbolic debugging, command stack, automatic command file execution, trace breakpoints, etc. While PC-NETSIM demonstrates the usefulness of SOS for discrete-event simulations, other forms of SOS have also been used as the operating system for both the packet switched network controller and modem controller of an HF network (Survivable Skywave IR&D), a multimedia controller (SIST: Survivable Information Systems Technology IR&D), and a UHF packet radio (LPROS). Other real time and packet switching applications are sure to be found for this versatile operating system.

Data Flow Diagram

A simplified data flow diagram of the PC-NETSIM execution environment is as follows:

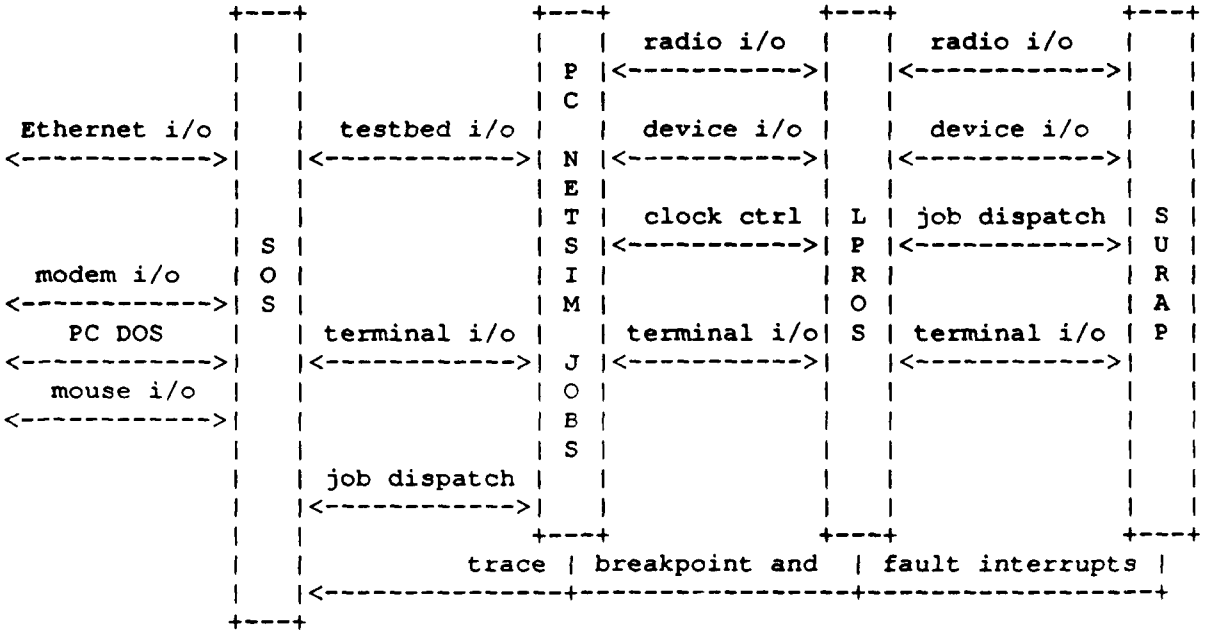

Figure 1. Simplified PC-NETSIM data flow diagram

At the highest level, SURAP generates radio i/o intended for other radios, device i/o intended for the radio's "attached" device (which in the PC-NETSIM environment resides externally in the testbed), and terminal i/o intended for an operator. All i/o passes through the LPROS on its way to SIMMAN, which then processes radio i/o via the channel model, device i/o via SOS's Ethernet drivers, and terminal i/o via SOS's (and ultimately DOS's) screen and keyboard routines.

At the lowest level, the testbed provides connectivity and timing information via SOS's Ethernet drivers to SIMMAN, which in turn sends timing and statistical information back to the testbed. SIMMAN manages the master clock, selects nodes for execution (according to the algorithm described in the section entitled "PC-NETSIM Clocks and Event Queues"), and calls the LPROS's dispatcher to allow it to execute process calls. SIMMAN also uses SOS's mouse handlers to allow a local operator to control the field view (a graphical representation of the network) for standalone testing and debugging.

One can also see from the diagram that all fault and trace breakpoint interrupts are handled by SOS, which causes SOS to freeze the entire simulation. Even when the simulation is frozen, the operator can continue to talk to all nodes. This gives the operator access to all data contained in the nodes as he explores the events that led up to the freeze. This is an very powerful tool and one that is not usually available in a real network. That is, in a real network only the frozen node would go into a suspended state, while the rest of the network would continue to run, usually overwriting much of the network state information germane to the fault or trace breakpoint. Also notice that SOS has support for modem i/o which allows an operator to remotely control and debug SOS and the processes it is executing (in this case PC-NETSIM). This same remote control capability is available over the Internet using the Ethernet drivers and the Internet Protocol (IP).

PC-NETSIM CLOCKS AND EVENT QUEUES

Discrete event simulations often treat service times as negligible or use random distributions to estimate them. However, even as processor performance continues to improve, the computational requirements seem to increase just as fast, making service times in many applications still very significant. Also, for a system characterized by many independent stochastic processes, such as a multinode packet switching network, where random packet transmissions, queuing delays, prioritized process execution, and complex routing algorithms make it virtually impossible to predict the outcome of an event at any one node, the service times are very hard to estimate. The technique described here, "Realistic Service Time Simulation" or "RESTS", overcomes the problems of incorporating realistic processing delays into a simulation very simply: it measures them directly. How this is achieved and the resulting technique used to manage the simulation is the subject of this section.

RESTS can be applied to any quasi-identity simulation, that is, the service times associated with the real part of the system can be measured directly. For instance, in a packet switching network simulation, the real communication protocol code can be executed by the simulation CPU, allowing the execution time to be measured, while the radio hardware and communication channel are simulated. The service times of the latter tend to be easy to calculate since they are based primarily on a deterministic channel access technique, the data rate of the channel and the length of the packets. Thus, for any quasi-identity simulation, the service times of the real, variable parts of the system can be measured, while those of the simulated part of the system can often be calculated.

If there were only one node in a network, it would be straightforward to maintain its clock and event queue using RESTS:

- o If there are outstanding events to be processed, measure the processing delays and add them to the clock.

- o If the node is idle, move the clock ahead to the next scheduled event or i/o event. This is a classical discrete event simulation technique.

To maintain the clocks and event queue for a multinode network, RESTS uses a new hierarchical approach, whereby many of the bookkeeping tasks normally performed by the simulation manager are pushed out to the individual nodes. In classical discrete event simulation techniques, the simulation manager maintains the queue(s) for all events. In RESTS, each node of the network is responsible for maintaining its own event queue(s), while the simulation manager just maintains an array of clocks, one for each node, and an i/o queue to manage the active i/o events (that is, the transmissions and receptions currently occupying channel bandwidth and access hardware, as opposed to the future i/o events queued up within each node). By taking this hierarchical approach, much of the potential complexity of a network simulation is avoided.

To maintain the array of node clocks, RESTS uses the measured service times to increment the clock of each node. A node is not eligible for processing again until the clocks of all other nodes have caught up to or passed its own clock. Another way of looking at this is that the node furthest behind is always the next one to be processed. Thus, this technique could be described as "catch-up node processing". If a node has no task scheduled, then its clock is set to the time of its next scheduled event and that node will not be processed again until then, unless a packet reception preempts its idleness. Also, the exact time of a packet transmission spawned by the processing of an event can be measured with respect to the clock of that node (instead of merely being treated as if it occurred at the beginning or end of the processing, as is typical). Thus, catch-up node processing allows RESTS to easily manage the independent clocks of a multinode network simulation.

As described, RESTS consists of three complementary techniques: direct measurement of processing delays, a hierarchical simulation architecture, and catch-up node processing. Each technique can be used independently of the other with some success. Taken together, they represent a unique and elegant solution to the problem of realistically simulating the unpredictable service times encountered in a packet switching network.

USER INTERACTION

A friendly and flexible user interface is of great importance to any software package that requires user interaction. User input and output should be conveyed in a manner most appropriate to the type of data being represented. For example, while keyboard input is suitable for

many types of commands, positioning of icons on a graphics screen is usually most easily done with a mouse or trackball. Likewise, graphics can often be used to represent output data much more effectively than is possible with ASCII text. PC-NETSIM has a simple but powerful user interface which gives the operator considerable flexibility in how he chooses to run experiments and monitor the results.

Rather than use bit mapped graphics, text mode is used to generate a graphical representation of the network. This was a compromise designed to maximize the speed and minimize the amount of code associated with the graphics, both of which are critical resources in a PC based simulation. A node within the network is represented by an icon consisting of up to 3 hexadecimal digits which can be used to express any important piece of information stored within the node. For instance, while the icon could be something static like the node ID, it is usually more interesting to let the icon display a dynamic variable, like one used to keep track of some important piece of routing information. Likewise, connectivity between the nodes is represented with text characters that depend on the orientation of nodes to one another. The following are examples of typical PC-NETSIM graphics screens using two different representations for their node icons:

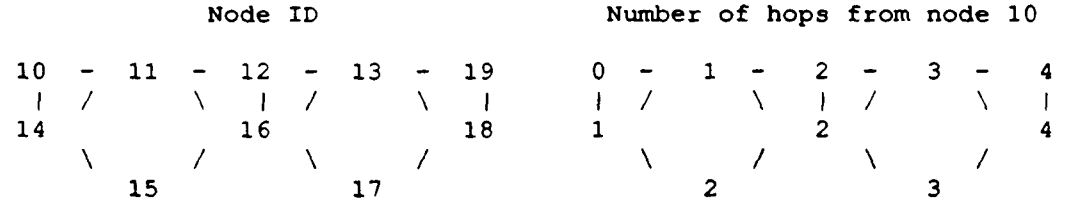

Figure 2. Typical PC-NETSIM graphics screens

Besides the ASCII values on the screen, node icons and connectivity bars are color coded to show important information about packets being forwarded through the network. For instance, a node in the process of transmitting a packet will be lit up with inverse video and the color used may be based on packet type, source or destination node, or any other user selected field within the packet. This graphical representation of the network allows dynamics to be observed as they happen, which is often impossible to do from data dumps taken after the fact. This is also something that is virtually impossible to do with a real network, which makes the simulated environment much better for studying complex network interactions.

The mouse interface allows the user to grab a node and move it around the screen. As he is doing so, in one mode the connectivity is automatically recalculated using the distance between nodes to determine path loss. In another mode the operator can use the mouse to manually adjust the path loss between any two nodes. The mouse is also used to quickly switch between different network views, that is, node

icon representations as previously described. Lastly, it can be used to select a particular node for terminal input. That is, any command typed by the user not intended to control the simulation will be passed on to the selected node to be executed as if it came from a terminal attached to a real radio. Thus, the combination of keyboard and mouse allow the user to control the simulation with a minimal amount of effort.

REPEATABILITY

In its default mode PC-NETSIM is stochastic, that is, the random nature of events guarantees that every experiment will be unique. However, there may be times when one wishes to run an experiment more than once with exactly the same results, even on different PC's. Normally there are two features of PC-NETSIM which prevent this: a hardware random number generator and the realistic measurement of processing delays, both based on the PC's hardware timer. Since this timer is driven by a different crystal than that of the CPU (except for the very earliest 4.77 MHz PC's), it is impossible to correlate the timer from run to run on the same PC, much less on different PC's. Thus, in order to make runs repeatable, PC-NETSIM uses a pseudorandom number generator (which, given the same seed, always generates the same pseudorandom sequence) and fixed processing delays. The problem with running in repeatable mode is that realism may suffer, especially if accurate processing delays play an important role in the experiment. For that reason, the non-repeatable mode is the default and gives maximum confidence in the results.

LINK PERFORMANCE MODEL [2]

A realistic network simulation requires a link model which encompasses the entire communications path, from the information source, through all encoding and modulation steps, through the transmitter and the channel, up to the receiver, and terminating at the information sink.

The PC-NETSIM link model determines the fate of packets attempting to traverse a link in the DARPA Packet Radio Network (PRNET). The primary component of the PRNET is the LPR, which is capable of omni-directional, spread spectrum transmission.

Before a packet enters the channel, three significant transmit parameters must be selected by the source LPR: transmit power, data rate, and forward error correction (FEC) rate. The transmit power affects the useful range of the signal and, hence, the number of potential receivers. The data rate determines the spread spectrum processing gain, which in turn affects the signal to noise ratio at the receivers. The FEC rate influences the ability of receivers to correctly receive a packet despite the presence of some limited number of errors.

A packet enters the channel via the LPR's antenna, which is characterized by a fixed gain. As the signal traverses the channel, it will suffer a path loss which increases with distance traveled. The signal might collide with simultaneous transmissions from other network participants; the combined effect of these friendly transmissions is referred to as multiple access interference. The signal may also collide with hostile jamming signals. The packet transmission time, which is a function of packet length, data rate, and FEC rate, affects the probability of collision.

A packet leaves the channel through the receiving LPRs' antennas. Front end thermal noise perturbs the signal by an amount specified by the receiver noise figure. The sequential decoder time-out period influences the probability that the packet will be correctly received despite some limited number of demodulation errors.

Implementation

In accordance with the event driven nature of PC-NETSIM, the fate of a packet is evaluated at four points during its journey through the link. At the inception of a new packet transmission the model evaluates the acquisition process at each radio in the network. A radio will acquire bit sync with the packet if the following criteria are satisfied:

1) The transmitter is disabled.
2) The receiver is enabled.
3) The receiver is not already in frame sync.
4) The received bit energy to noise density ratio (Eb/No) > bit sync threshold
5) The received signal power is sufficient with respect to the power sum of all interfering preambles.
6) The received signal power is sufficient with respect to the power sum of all interfering signals.

The last two critera require a knowledge of the channel condition. For this reason, each radio maintains a "state-of-the-channel" variable which represents the degree of multiple access interference in the channel at a given time. Any radio already in bit sync will re-evaluate the bit sync conditions in light of the new transmission.

The next packet event of interest occurs approximately five bit-periods after the transmission begins. At this point, any radio which did not originally acquire bit sync with the packet is given a second chance. The "late bit sync" criteria are the same as above except that criterion 4 has a higher threshold.

The end of preamble event is the next point of evaluation. At this point, any radios which are in bit sync with the packet will also achieve frame sync.

The final evaluation point is the end of packet event. At each radio in frame sync with the packet, a signal to noise ratio is calculated using the maximum value of interference which occurred during the transmission. From this ratio, the bit error rate (BER) of the packet can be estimated. The packet error rate (PER) is then approximated as a function of FEC rate, decoder speed, decoder time-out period, packet length, and BER. The subsequent comparison of the PER with a random number ultimately determines whether the packet is successfully received.

TESTBED ENVIRONMENT

PC-NETSIM was designed not only to serve as an individual development station, but also to function as the kernel of a realistic testbed where external network entities interact with the simulation in much the same way that they would with a real LPR network. Network Interface Units (NIUs) can be used to attach terminals or computers that wish to communicate across the network. Traffic generators can inject packets into the system to simulate typical traffic patterns while a network monitor gathers statistics from individual radios. The orchestration of long experiments is done from a program called SALT (SURAN Automated Laboratory Testbed) running on a SUN workstation (either locally or anywhere on the Internet) which executes script files to manipulate the network (control traffic patterns, change connectivity between nodes, etc.) while continuously collecting data from the radios for off-line analysis. This analysis results in graphs of virtually any type of performance criteria (throughput, delay, packet loss, etc.) that a network analyst could desire. The results of these experiments are fed back to the SURAN protocol developers, who then work to correct problems that may have been identified.

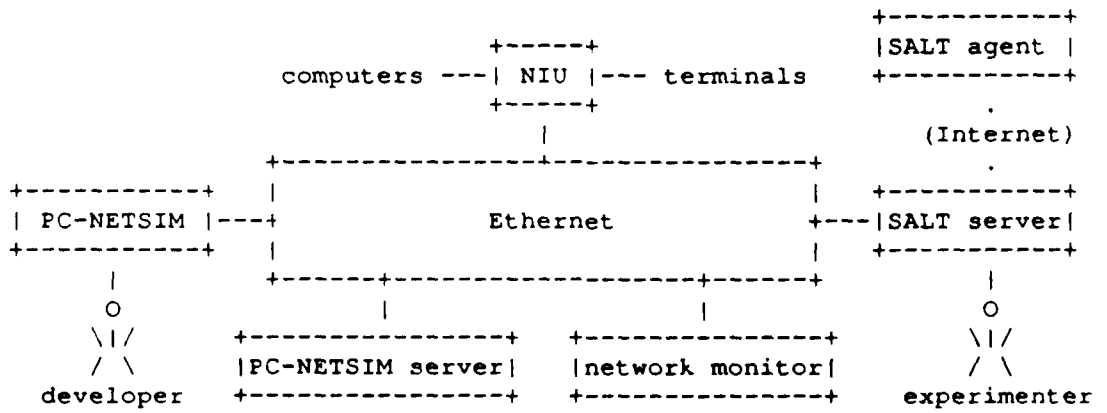

Figure 3. Testbed Environment

The major source of complexity in using a simulation in a testbed rather than real network hardware is that of synchronization: PC-NETSIM does not run in real time. It can either warp ahead or run very slow relative to real time, depending on the traffic load. To allow external testbed devices to synchronize to it, PC-NETSIM recognizes a suite of packets used by these devices to, among other things, control how long it should run before stopping to collect data, control how often it should output a synchronization packet, and change network connectivity. These packets, along with the network traffic packets, are encapsulated in IP packets (to allow safe coexistence with other LAN devices and protocols) and transmitted over the Ethernet. In the SALT environment, all packets to and from PC-NETSIM are processed by a specially programmed NIU called the 'PC-NETSIM server', whose job it is to multiplex/demultiplex testbed packets and coordinate requests for PC-NETSIM services. In this way, PC-NETSIM extends the services it offers beyond the individual protocol developer to a wider audience of men and machines capable of running more complex experiments and analysis than would be practical by the PC alone.

SUMMARY

Traditionally, large scale simulations have been done on mainframes rather than PCs for reasons of speed, memory, and mass storage requirements. However, today's PCs compare quite favorably in these respects with mainframes operating in a time-sharing environment under typical processing loads. Couple this with the fact that the IBM PC and compatibles provide a widely standardized and inexpensive program execution environment and you have a near optimal development environment. This is especially true when the target software is written in a language that can be executed on the PC, providing the opportunity for a quasi-identity simulation like PC-NETSIM to execute the target software and incorporate realistic nodal processing delays. It is hoped that the descriptions of these and other features of PC-NETSIM has struck a resonant chord in the imaginations of software engineers who might find a use for some of these ideas in their own applications.

References:

[1] Garage S. Fishman, "Concepts and Methods in Discrete Event Digital Simulation", John Wiley & Sons, New York, 1973.

[2] Pete Bausbacher, SURAN Technical Note # 62, "The Link Performance Model of a Packet Radio Network Simulator", Defense Advanced Research Projects Agency, Arlington, VA, March 1989

Biographical Sketches:

C. David Young received a Bachelor of Arts degree from Rice University in 1978 with a double major in Electrical Engineering and German Literature. He gained considerable experience designing hardware and software for microprocessor based monitoring devices prior to joining Rockwell in 1981, where he has worked on the SURAN and Survivable Skywave programs as a protocol developer. He is currently a member of the Advanced Technology and Engineering department where he is the lead software engineer on the Rockwell SURAN protocol development team.

Peter E. Bausbacher received the B.S. degree in electrical engineering from The University of Texas at Austin in 1986. He is currently pursuing the M.S. degree in electrical engineering at The University of Texas at Dallas. Prior to Rockwell, Mr. Bausbacher worked for IBM, where he aided in the design of high density gate arrays, and LTV, where his responsibilities included RF and electro-optic analysis as well as analog and digital circuit design. In 1988 he joined Rockwell's Advanced Technology and Engineering department, where his primary emphasis has been on analytical modeling of communications systems with particular attention to packet radio networks employing spread spectrum techniques.