Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

PACKSIM

A Personal Computer Implementation of a Packet Switched Network Simulation Model.

Horst E. Ulfers
Defense Communications Agency
Center for
Command, Control and Communications
AHS, Arlington, Virginia

ABSTRACT

PACKSIM is a recent personal computer implementation of a large scale simulation model of a packet switched network simulator which had been previously developed for use on the IBM System/370 mainframe. This paper addresses special modeling techniques used in simulating the flow of various types of packets through a packet switched network. Techniques to keep the overhead associated with discrete simulations low are even more important for personal computer implementations then for the large main frame. The model is written in the PC-SOL simulation language interspersed with special PASCAL procedures and functions for traffic generation and alternate routing. The alternate routing algorithm is implemented as a special module, such that it can easily be replaced with a different one when desired. The model simulates in detail the specific protocols within a typical packet switched network. The paper also discusses the procedure to be followed in compiling and executing the model. Inputs to the model are parameterized. Consequently, a recompilation of the model becomes necessary only when major changes are to be implemented. A pre-processor will generate the input data for the model in interactive fashion with the user. During execution the model generates a log file that contains records of all significant events of the simulation. The PC-SOL analytical routines are used to analyze the log tape and generate statistics as well as presenting graphics of the simulation. The paper concludes with a description of the simulation of a typical, but small, network to demonstrate the capabilities of PACKSIM.

1. MODEL DESCRIPTION

The PACKSIM simulation model has been designed to simulate the flow of information packets and control packets through a packet switched network as realistically as needed for the evaluation of protocols, routing algorithms, and system control actions as they affect network performance. The design goal was to build a model that simulates the flow between backbone packet switches to a fine degree while handling the access area traffic flow only to the degree that it contributes to backbone network congestion. In general, the arrival of messages at hosts is modeled on a per line basis. The model was implemented in two versions. One with an idialistic routing algorithm, assuming instant routing updates available at all switches. This version also ignores the access line problem and assumes nonblocking. It is,therefore, better suited for network engineering, where analytical methods cannot provide the desired degree of accuracy and other more detailed modelling would be too time consuming. The other version contains an actual routing algorithm, implemented in modular form, simulating the time delay usually encountered between dynamically updating tables driving the routing algorithm at each packet switch.

The following discussion pertains to both versions of the model. Both simulate the packet flow across the entire network. Messages composed of single packets and multiple packets are generated on a host-to-host basis. The nearest packet switch is determined by a homing table. File transfer messages consist of a constant number of packets, in this case eight. They are packetized at the origination switch and transmitted across the network simultaniously after an opening packet has established the connection. Messages consisting of a single packet are sent immediately since a buffer for one packet is always reserved and does not require a confirmation. The routing algorithm represents the algorithm implemented in the ARPA network. The basic model consists of two processes: CONTROL and LOAD. In the second version the routing table update is implemented by two additional processes: UPDATE, to periodically update the routing tables based on least delay and shortest number of hops, and REVISE, to periodically update the GODOWN and COMEUP tables for switches.

1.1. Components Modeled

Figure 1 illustrates the basic components modeled. Both versions of the model simulate the basic packet switch with central processor (CPU), the input queue (INQUE), the buffer pool (BUFFER), the task queue (TASKQUE), and the transmission links (LINK) with associated modem queues (MODQUE). The second version of the model also simulates the access lines (LINES) and output queues (OUTQUE). In it's current implementation the model collects the following data to be presented as histograms: The time it takes for packets to travel through the network (PKTDELAY), the time it takes for a file transfer message to be packetized, transmitted, and re-assembled (MSGDELAY), aborted packets are recorded with the time they were in the system (ABORTTABLE), and finally the number of hops across the network for each packet is recorded in table HOPTABLE.

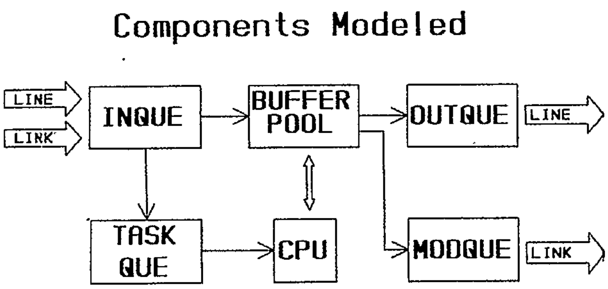

Figure 1: Components Modeled

2. MODEL DESIGN

This discussion is based on the second, more complex version of the model. However, the reader should keep in mind that the simpler model simply has the processes UPDATE and REVISE eliminated and does not implicitly model the access lines and their associated queues. Model PACKSIM was implemented for use on the IBM Personal Computer using the SOL-PC simulation system. SOL-PC uses the Simulation Oriented Language (SOL) to formulate the model. Since the Language allows the inclusion of PASCAL code, several special functions have been implemented as PASCAL procedures; most notebly the routing algorithm.

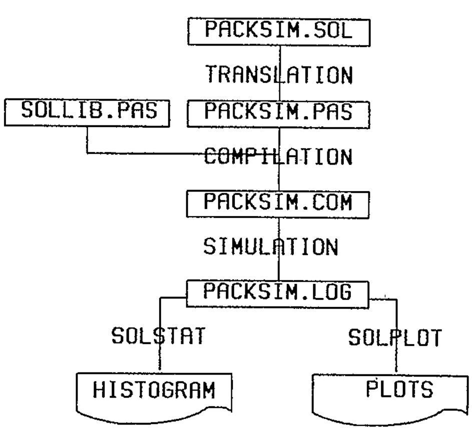

Figure 2: Operational Flow

Figure 2 illustrates the tasks involved from coding the model to analysing the simulation results. SOL-PC translates the source code written in SOL into PASCAL, then compiles it into an executable module. During model execution a Log File is generated which then is analyzed by two tools the SOL-PC Simulation System provides, a STATISTICS and a GRAPHICS program. Both are interactive and can optionally provide hard copies. The model will run on PC configurations with 512k of RAM or more. It can be operated with two floppy disk drives. However, a hard disk drive, better still a removable hard disk, is desirable for increased execution speed as well as for accomodating log data of long runs.

Figure 3 illustrates the structure of model PACKSIM. The upper horizontal block represents the global entities of the model. Each of the four blocks below represents one of the processes which are executed in a parallel fashion.

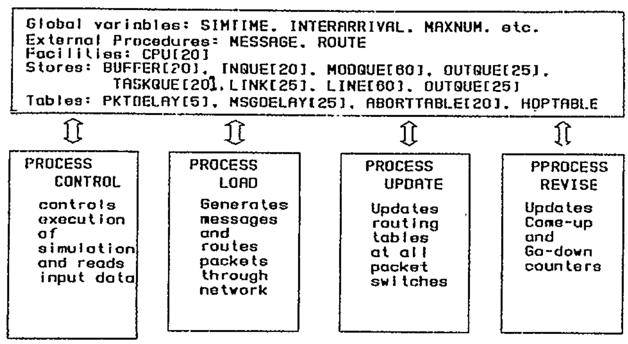

Figure 3: Model Structure

SOL requires that in the source code all global entities are declared ahead of the first process. All processes can make use and compete for the resources (FACILITIES and STORES) declared globally. A FACILITY is the SOL resource used to model a time shared device like a central processor unit (CPU). A store is used to model a space shared device like memories, buffers and queues; A store is declared with a certain capacity. Also a transmission link can be modeled as a STORE, it's capacity representing the number of circuits. Tables are used to collect special statistical data during the simulation. Standard statistics on all resources are collected automatically and sent to the log file.

At the begin of the simulation - at time 0 - one transaction is started at the beginning of each of the four processes. The SOL conventions specify that the processes are handled in the order in which they have been declared, as illustrated in Figure 1 from left to right. Consequently, Process CONTROL will start first, following by Process LOAD, Process UPDATE, and Process REVISE.

3. MODEL OPTIMIZATION

Before going into the detailed discussion of the model logic, it might be worthwile to explain some techniques that can be used in simulation of communication systems in general and of packet switched networks in particular to keep the size of the model and the execution time down.

3.1. Transactions

As mentioned before, the transaction is the unit of flow, in this case representing a single packet. Each packet has to carry along a number of descriptors which are called LOCAL VARIABLES in SOL terminology. These may describe items like the originating host, terminating host, time of message generation, packet identifier, etc. A transaction also maintains its own record of the SOL resources it is using. When a transaction enters a queue condition, e.g. when it executes a conditional or unconditional WAIT statement or when it encounters blocking on a resource, this transaction with all it's descriptors is entered into one of many queues until it is time to re-activate it. This is the process which places the heaviest load on the queue management routines and the physical queue space in RAM. The strategy to follow is to keep the number of transactions one must simultaneously maintain in queue as small as possible and to allocate as few local variables to a transaction as possible. Furthermore, it is advisable to carefully plan the Resource parameter R in the process declaration that reserves space in each transaction for resource management. The more resources one particular transaction is allowed to use the more space must be reserved for a transaction. Other techniques which lead to conservation of transactions are discussed with the model logic in each process.

3.2. Reentrant Submodels

Another technique, when applied will keep program memory size down by designing the submodels in re-entrant fashion. All resources are dimensioned as indicated by the value in brackets. This SOL feature, allowing dimensioned resources is significant for modeling a network with many identical nodes and links.

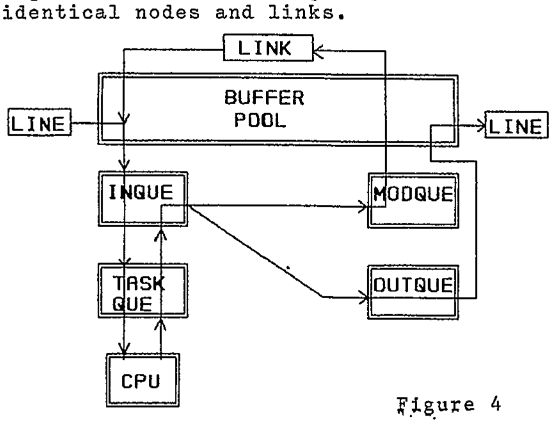

Figure 4

Figure 4 illustrates the flow of packets through the model as contained in Process LOAD. A transaction, representing a packet, traveling through the backbone network may traverse the switch model several times, each time with a different switch identifier. This switch identifier, a local variable to the transaction, is used as a subscript for the resources modeled at the switch. This method allows use of the same switch model numererous times in a re-entrant fashion, simply by re-setting the switch identifier. This is accomplished by the routing algorithm which does not only determine the next node but also the identifier of the link conecting the two nodes. Each different subscript of a resource then represents the same resource but at a different nodes or link. Consequently, the dimension of the resource specifies the maximum number of each the model can handle.

There is another advantage to this technique, it gives the model designer a simple means to increase the size of a particular model. This model, for instance, could be easily modified to handle 200 switches by changing the dimension 20 for all resources modeled in the switch to 200.

4. MODEL IMPLEMENTATION

The following paragraphs adress the implemntation of the model and discuss in detail the logical flow through each process.

4.1. Process CONTROL.

Process CONTROL is the first SOL Process and is used to control the simulation run. This process first reads a file which contains the simulation parameters and other input data used in initializing arrays that control the simulation. These data contain the connectivity matrix, the host homing table, and the traffic matrices. The input file has to be in a specific format, e.g. traffic matrices must be cumulative as required by the sampling algorithm. A special interactive program is available to the user to easily generate this file. Process CONTROL uses three time parameters to control the simulation: (1) SIMTIME, the simulated time. When expired the simulation is terminated and all files are closed. (2) MONTIME, the monitor time interval between writing a transaction record to the system output file. (3) BRKTIME, the time interval between performing checkpoint/restarts. This function is easily implemented by just one transaction which cycles in a loop executing WAIT statements until the simulated time has expired and then executes a STOP statement that terminates the simulation.

4.2. Process LOAD.

Process Load simulates the flow of information across the packet switched network. Each individual packet is simulated as it travels across the network, enters a queue, is re-activated, until it is finally delivered at the destination. It is this process which is critical for the build-up of large queues.

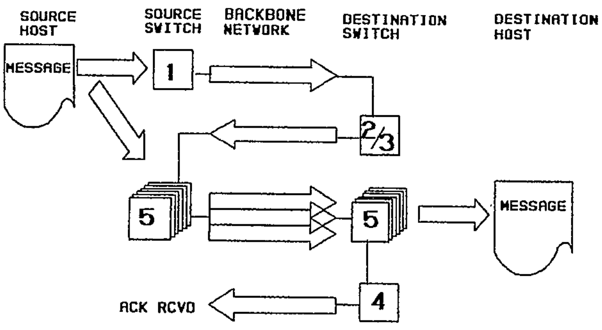

Figure 5: Typical Packet Flow Protocol

The transmission of a message causes five different types of packets to be sent between the originating node and the destination node (see Figure 5). At the beginning of communications between two packet switches, after the message has been packetized, an opening packet (Type 1) is sent from the originating switch to the destination switch. If the destination switch has sufficient buffer space to accept the communications, a positive RFNM (Request For New Message) packet (Type 2) is returned, otherwise the transmission is denied by a negative RFNM packet (Type 3). The originating switch starts sending packets (Type 5) in rapid succession. The reception of each packet at the destination switch is confirmed by piggy-backing a short message on to another packet to the originating switch. Since this process does not add to the load of the network it is not modeled explicitly, but is represented by simulating the transmission time delay encountered. After the last packet of a message has been received an acknowledge packet (Type 4) is returned to the originating switch. This terminates the transmission of a message. Analysing this process one can see that all packets except the imformation packets (type 5) are sent sequentially. The normal approach to simulate the packet flow of a message would be to simulate each individual packet and then put the packets into a holding queue until the last packet, the acknowledge packet has been processed. This approach, however, would lead to a rapid build-up of the internal transaction queue. As a result the model may run out of RAM space besides substantially slowing down the execution . The approach implemented in this model takes advantage of the fact that all but the information packets (Type 5) are sent sequentially. Therefore, one and the same transaction simulates all packets, types 1 through 4, and the first information packet. If there are more than one information packet then the remaining packets are sent in parallel. This approach is implemented by attaching the packet type as a Local Variable to the transaction, starting it out at the originating node with Type=1, then returning the same transaction to the originating node with Type=2 or Type=3, and again sending the same transaction back to the destination node as a Type=5 packet. Other transaction are now being generated to simulate the parallel transmission process of the remaining information packets. The original transaction is held in queue at the destination node until the last information packet of a transmission has been received. Then it is set to Type=4 and is returned to the originating node. The other information packets are immediately cancelled when they have arrived at the destination. While transactions are queued up, usually, the PC-SOL Simulation System keeps track of loads on buffers and other modeled resources. This modeling approach is superior to the straight forward method, but requires some additional bookkeeping to maintain the proper load on buffer space at the destination node. Otherwise, the PC-SOL Simulation System would have kept track of the load on buffers while transactions are queued up.

Other functions of process LOAD are the generation of messages, finding routes through the network, and the simulation of transmission links. There are two external procedures, written in PASCAL language: (1) Procedure MESSAGE generating pairs of nodes, the originating and the destination node. This is accomplished by sampling process two traffic matrices. (2) Procedure ROUTE finding the next node toward the destination node by executing the routing algorithm. Both procedures can easily be changed or replaced, allowing for implementation of other traffic generation schemes and routing algorithms.

4.3. Process UPDATE

Process UPDATE simulates a system control function that automatically updates the routing tables at each node. It closely resembles the functions used in the ARPA routing algorithm. It performs two basic functions: Determination which other destination nodes are reachable, and a calculation of the route of the least delay for the nodes which are reachable. The process maintains a set of tables for each node in the network. Each table has information necessary to derive the optimum route to any other node in the network, like least number of hops and shortest delay. Each node re-computes these tables in fixed intervals based on information received from it's neighbour nodes. It then formulates a routing update message that is sent to it's neighbouring nodes after the next update interval has expired.

4.4. Process REVISE.

There are two tables at each node which maintain the status of each other node. The tables are kept current about switches going down and switches coming up. Process REVISE updates these tables in fixed intervals. The update interval is a parameter that can be set at the begin of the simulation.

5. A SAMPLE NETWORK

Although this paper concentrates on special implementation techniques of a packet switched model, it was thought worthwhile to take the reader through the major steps of preparing the inputs to the model and means available to analyze the results. In the following the simulation of a sample packet switched network is demonstrated to show the capabilities of the PACKSIM simulation model.

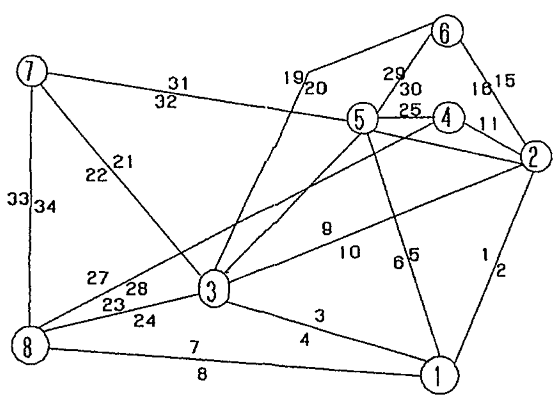

Figure 6: Packet Switched Sample Network

Figure 6 shows the network. For simplicity it is assumed that only one host is connected to each node, consequently the hosts are mapped into the nodes. Nodes are numbered sequentially. Each link is assigned two numbers, an even and an odd number. The odd numbered link has the traffic flowing to a node with a higher numbered ID while traffic through even numbered links flows into the opposite direction.

5.1. INPUT FILE PREPARATION

The user must prepare a connectivity matrix, an entry of '1' represents a connection between a pair of nodes, a '0' no connections. The user must also prepare two traffic matrices, one for single packet messages the other for multi-packet file transfer messages. Fortunately, there is an interactive input preprocessor that leads the user systematically through all steps in preparing the input file, and processes the information into the format required by the model. Figure 7 lists the simulation parameters and arrays as they have been generated in the input file. This listing is produces each time at the beginning of model execution.

```
START TIME AND STOP TIME OF TRACE:
50000 50000
MAXNUM=100 SIMTIME=50000 BRKTIME=5000
MONTIME=50000
NODES=8 HOSTS=8
HOST TO SWITCH HOMING
  2  3  4  5  6  7  8  1
CONNECTIVITY MATRIX:
  0  1  1  0  1  0  0  1
  1  0  1  1  1  1  0  0
  1  1  0  0  1  1  1  1
  0  1  0  0  1  0  0  1
  1  1  1  1  0  1  1  0
  0  1  1  0  1  0  0  0
  0  0  1  0  1  0  0  1
  1  0  1  1  0  0  0  0
TRAFFIC MATRIX FOR SINGLE PACKETS:
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10
TRAFFIC MATRIX FOR FILE TRANSFERS:
  0  2  2  2  2  2  2  2
  2  0  2  2  2  2  2  2
  2  2  0  2  2  2  2  2
  2  2  2  0  2  2  2  2
  2  2  2  2  0  2  2  2
  2  2  2  2  2  0  2  2
  2  2  2  2  2  2  0  2
  2  2  2  2  2  2  2  0
INTERARRIVAL TIME= 500
AVERAGE LINK TRANSMISSION TIME= 100
AVERAGE LINE TRANSMISSION TIME= 500
CPU PROCESSING TIMES ARE: 100 100 100 100
THE LINK ASSIGNMENTS ARE AS FOLLOWS:
  0  1  3  0  5  0  0  7
  2  0  9 11 13 15  0  0
  4 10  0  0 17 19 21 23
  0 12  0  0 25  0  0 27
  6 14 18 26  0 29 31  0
  0 16 20  0 30  0  0  0
  0  0 22  0 32  0  0 33
  8  0 24 26  0  0 34  0
COMBINED CUMULATIVE TRAFFIC DEMAND MATRIX:
 752    94   188   282   376   470   564   658   752
  94    10    22    34    46    58    70    82    94
 188    12    22    34    46    58    70    82    94
 282    12    24    34    46    58    70    82    94
 376    12    24    36    46    58    70    82    94
 470    12    24    36    48    58    70    82    94
 564    12    24    36    48    60    70    82    94
 658    12    24    36    48    60    72    82    94
 752    12    24    36    48    60    72    84    94
CUMULATIVE FILE TRANSFER TRAF DEMAND MATRIX:
 112    14    28    42    56    70    84    98   112
  14     0     2     4     6     8    10    12    14
  28     2     2     4     6     8    10    12    14
  42     2     4     4     6     8    10    12    14
  56     2     4     6     6     8    10    12    14
  70     2     4     6     8     8    10    12    14
  84     2     4     6     8    10    10    12    14
  98     2     4     6     8    10    12    12    14
 112     2     4     6     8    10    12    14    14
```

Figure 7: Input File Listing

5.2. TYPICAL STATISTICS AND GRAPHICS OUTPUTS

During model execution all significant events are recorded in the LOG file, e.g. when a transaction enters or leaves a Store, seizes or releases a Facility, and when data are entered into a Table. The PC-SOL Simulation System provides two standard analysis tools, SOLSTAT and SOLPLOT. SOLSTAT produces statistics on the resources modeled as well as histograms on data collected during the simulation, in table or bar-graph format. More detailed load analysis is possible by SOLPLOT, a graphics routine that graphs load over time, and has the capability to zoom in on any segment of the graph and expand it to its limits for the finest details. The following figures give some examples.

Figure 8 lists the standard statistics output on resources modeled. Note, that resources that are modeled, but have not been used, are not listed. This feature is convenient it suppresses statistics on nodes 9 through 20 which are modeled but have not been used in this 8 node sample network.

FACILITY NAME	TIME	UTILIZATION
CPU[1]	10000	0.1480
CPU[2]	10000	0.1900
CPU[3]	10000	0.1800
CPU[4]	10000	0.0980
CPU[5]	10000	0.0880
CPU[6]	10000	0.1200
CPU[7]	10000	0.0740
CPU[8]	10000	0.0800

STORE NAME	TIME	CAPCTY	MAX	OCCP	UTLZN
BUFFER[1]	10000	1000	5	8177	0.001
BUFFER[2]	10000	1000	13	30225	0.003
BUFFER[3]	10000	1000	31	47284	0.005
BUFFER[4]	10000	1000	23	89211	0.009
BUFFER[5]	10000	1000	15	87537	0.009
BUFFER[6]	10000	1000	21	23932	0.002
BUFFER[7]	10000	1000	5	4591	0.000
BUFFER[8]	10000	1000	3	5110	0.001

Figure 8: Standard Statistics Listing

TABLE PKTDELAY[5] TIME=50000 ENTRIES=137
SUM OF ALL ENTRIES=56839.446 MEAN=414.886
STANDARD DEVIATION=414.585

UPPER LIMIT	COUNT	PERCENT	CUMULAT	MULT/MEAN
0	0	0.00	0.00	0.0000
100	3	2.19	2.19	0.2410
200	63	43.99	48.18	0.4821
300	17	12.41	60.58	0.7231
400	35	25.55	86.13	0.9641
500	9	6.57	92.70	1.2051
600	2	1.46	94.16	1.4462
700	2	1.46	95.62	1.6872
800	4	2.92	98.54	1.9282
900	0	0.00	98.54	2.1693
1000	0	0.00	98.54	2.4103
1100	0	0.00	98.54	2.6513
1200	0	0.00	98.54	2.8924
1300	2	1.46	100.00	3.1334

Figure 9: Histogram Table

Figure 9 shows the data collected in table DELAYTABLE[5] in histogram format, while figure 10 below presents the same data in bar graph form.

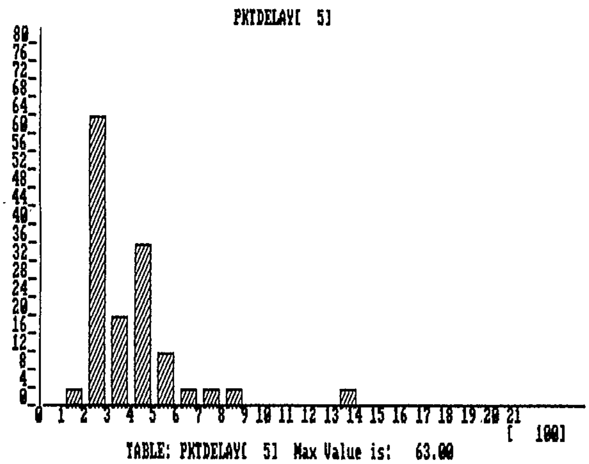

Figure 10: Histogram Bar Chart

Figures 11 and 12 have been produced with the SOLPLOT utility.

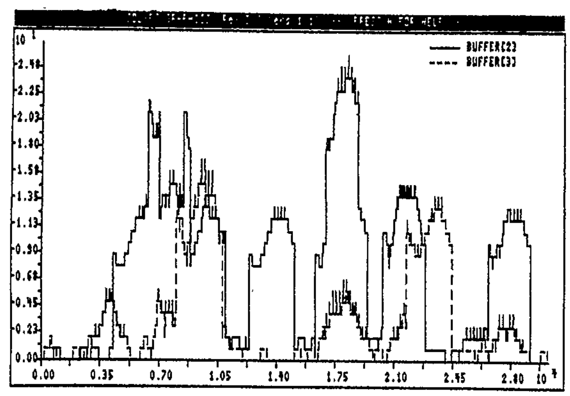

Figure 11: Loading on Stores BUFFER[2] and BUFFER[3]

During the analysis of simulation results, it often becomes necessary to pinpoint exactly the time when the heaviest load on a particular store had occured during the simulation. For instance, the overview graph in figure 11 just gives an indication that the heaviest load occured about time unit 17500. The zoom feature of the graph utility can now be used to pick a narrow time frame around the value and blow it up to show details with as fine a grain as needed, even down to the time unit level. Figure 12 illustrates the results of this procedure. Here, one realizes that the peak load consists of a series of pulses, most likely caused by a series of packets originating from a file tranfer message.

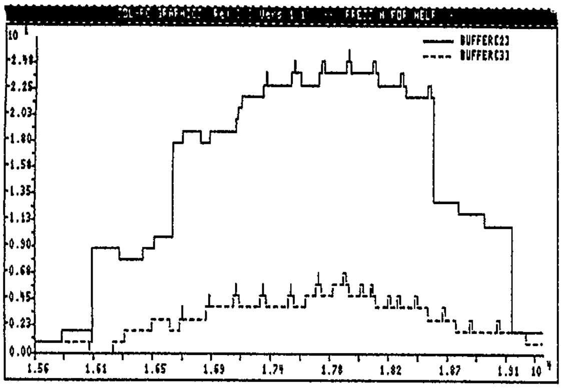

Figure 12: Illustration of Zoom Feature

6. CONCLUSIONS

It has been shown, that by using techniques presented in this paper, large network simulation models, like packet switched networks, can be implemented on personal computers. Similar Models, previously, required a large main frame to run effectively. Today's personal computers are very reliable and offer the user a much more efficient, interactive environment leading to faster simulation model implementation and turn-around. As personal computers become more powerful, increased speed and larger memory size, they are bound tobecome the preferred simulation vehicle for all but very special real time simulation models depending on large data bases.

REFERENCES

1. Horst E. Ulfers," PC-SOL, A Personal Computer Implementation of a Simulation Oriented Language", Proceedings of the Summer Computer simulation Conference, July 1986.

2. PC-SOL Language Reference Manual, Systems Simulation Consultants, 11051 Ring Road, Reston, Virginia, 22090.

3. D. E Knuth and J. L. McNeley, "SOL - A Symbolic Language for General Purpose Systems Simulation", IEEE Transactions on Electronic Computers, IC-13, No.5 (Aug. 1964).

4. J. Armstrong, H. Ulfers,, D. Miller, H. Page, "SOLPASS - A Simulation Oriented Language Programming and Simulation System", Preceedings of the Third Conference on Applications of Simulation, Dec 1969.

5. Horst E. Ulfers, "PACKNET - A Packet Switch Network Simulator," Proceedings of the 1975 ICC, June 1975.

AUTHOR'S BIOGRAPHY

Horst E. Ulfers is a Senior Electrical Engineer at the Defense Communications Agency. He obtained the Degree of Diplom-Ingenieur (MSEE) at the Technical University of Munich in 1954. After 5 years of employment with the Siemens & Halske Research Center in Munich he joined in 1958 the US Army Signal Corps Laboratory in Ft. Monmouth, New Jersey, and later in 1969 the Defense Communications Agency in Arlington, Va. During his career he specialized in the field of Communications Network Analysis and Systems Engineering. He developed numerous aimulation models and developed and implemented General Purpose Simulation Systems on Main Frame Computers and Personal Computers. He is the author of several publications in this field and is currently working in the Center for Command, Control and Communications at the Defense Communications Agency.