CITI-TR-90-1 June 15, 1990

NetMod: A Design Tool for Large-Scale Heterogeneous Campus Networks

David W. Bachmann
Mark E. Segal
Mandyam M. Srinivasan
Toby J. Teorey

Center for Information Technology Integration (CITI)
The University of Michigan, Ann Arbor, MI 48105-2016

Center for Information Technology Integration

The University of Michigan
Information Technology Division
2901 Hubbard
Ann Arbor, MI 48105-2016

Electrical Engineering & Computer Science
and
Center for Information Technology Integration
The University of Michigan
Ann Arbor, MI 48109-2122
(313) 763-5216

NetMod: A Design Tool for Large-Scale Heterogeneous Campus Networks

David W. Bachmann
Mark E. Segal
Mandyam M. Srinivasan
Toby J. Teorey

June 15, 1990

NetMod: A Design Tool for Large-Scale Heterogeneous Campus Networks

Abstract

The Network Modeling Tool (NetMod) uses simple analytical models to provide the designers of large interconnected local area networks with an in-depth analysis of the potential performance of these systems. This tool can be used in either a university, industrial, or governmental campus networking environment consisting of thousands of computer sites. NetMod is implemented with a combination of the easy-to-use Macintosh software packages HyperCard and Excel. The objectives of NetMod, the analytical models, and the user interface are described in detail along with its application to an actual campus-wide network.

1. Introduction

The Network Modeling Tool (NetMod) assists the development of an in-depth understanding of the performance of state-of-the-art local area network (LAN) technologies in either a university, industrial, or government campus networking environment consisting of thousands of computer sites. A principal application of the tool is to help the campus network designer to configure a potential user's data network using proposed hardware and software components.

NetMod can analyze an existing or proposed network in terms of its basic performance characteristics, e.g. component utilization, throughput, and packet delay times. It provides a capability for sensitivity analysis of the performance based on changes in the workload parameters and either minor or major changes in the network topology and connectivity. Thus it is possible to quickly evaluate and compare several alternate configurations: a feature which will greatly assist the network design process.

NetMod models popular LAN technologies such as token ring and Ethernet and some of their variations, as well as special network components such as routers, bridges, gateways and adapter cards. Network operating system details are beyond the scope of the existing analytical models, but are the subject of ongoing research.

The tool enables the designer to configure the current or proposed system topology graphically on the screen. This is currently done through a HyperCard interface. Using the topology currently displayed, the designer can evolve to a system configuration based on his or her experience or understanding of the user's requirements, and knowledge of the conceptual model. The tool then estimates the performance of the design and displays it to the designer.

The facilities provided by NetMod include ease of subnetwork definition, the flexibility to easily include/delete subnetworks, the ability to provide substantial user interaction for either simple parameter changes or major reconfiguration of LANs and backbones, and the provision of a hierarchical modeling capability for extremely large models. For example, a model of the University of Michigan's proposed network includes over 160 submodels representing different types of LANs, interconnect devices, mainframes, and workstations/personal computers. About 25,000 devices are represented in this model.

The basic approach taken by NetMod is to systematically decompose a model of the network into several related submodels, analyze the resulting submodels, and then consolidate the results of these analyses [CST88]. Currently, all the submodels in NetMod use closed form analytic expressions, considering the effects of the lowest two OSI layers, namely the Physical and Data Link (MAC sublayer only) layers. Future versions will analyze the Logical Link Control (LLC) sublayer and network and transport layers. Simulation is used to study some network components in more detail and to validate/modify the analytic models [CST88]. Real network measurements are also being made to validate the various submodels [BST89].

NetMod provides a graphic interface that corresponds to the world view of the network designer, with icons that represent rings, buses, routers, workstations, etc. This is combined with the quick feedback which is possible using the analytic approach, to provide a capability for interactive analysis of potential network designs, quickly ruling out trouble-prone configurations and identifying potential bottlenecks. The designer does not need to be an expert in any modeling discipline, nor does he/she need to keep a bookshelf lined with the specifications of various network media and protocols. One merely needs to know what devices are present in the networks and how they are connected.

It is useful to compare NetMod with other analytical and simulation tools that currently exist for modeling and evaluating computer networks. There are many simulation program generators in the literature, several using graphic front ends (GIST [SDM85], SIMF [SWM86], PAW [MeMo85], BONES [Frost88]). Recently some simulation-based tools have become available that allow the designer to layout the network as a system of ethernets, token rings, and routers (NETWORK II.5 [Garr88], PLANS [NMMT84]). However these tools usually require programs to be written in order to adequately describe the network. While simulation models could provide more detailed statistics about the performance of the networks, they are very time consuming. Indeed, even for one replication of a single network design, the model construction and its subsequent execution could require several hours before any meaningful statistics are obtained.

There are very few graphically oriented analytic tools, the notable exceptions being RESQME [GMW86] and NUMAS [Muel84]. The majority of analytic tools require the user to describe a physical network in terms of a network of queues (QNAP2 [VePo84], MANUPLAN II [SuTo88]). Consequently, this requires a more sophisticated user. There has also been some work based on stochastic petri networks (GSPNA [MBCC84]) and stochastic activity networks (METASAN [SaMe87]).

NetMod is implemented on a Macintosh to make use of the existing HyperCard user interface environment together with the simplicity of the Excel macro facility for computations. It is a tool which can be used in the early stages of network design to rapidly evaluate several alternate configurations, and select a few promising candidates. The candidate configurations may then be simulated in detail to obtain more extensive performance statistics.

2. NetMod Features

The user enters the network description through a Hypercard interface. Alternately, the user could choose to enter the description using Excel dialog boxes. In the input process, the user decomposes the network into submodels (for example a departmental Ethernet might be one submodel). These submodels are analyzed using Excel spreadsheets. Parts of the spreadsheet calculate the level of traffic that passes through a particular submodel, while other parts use these traffic levels to calculate the output statistics for that submodel. Section 3 presents a detailed discussion on how NetMod is used. In this section the parameters specified by the user, the output statistics generated, and the submodels used, are described.

2.1. User Inputs

The inputs to NetMod are i) the topology parameters, ii) the workload characteristics, and iii) the characteristics for the devices/transmission media in the network. These are entered using either the HyperCard interface or the Excel dialogues. Following this, they are passed to Excel to build the formulas modeling their performance. At this stage, the user enters the number of nodes of each type. With this input, the output statistics for the entire model are generated. In the following, the term node will frequently be used to refer to a device or a LAN.

2.1.1. Topology parameters

To describe the network topology, the different node types, namely host computer, workstation, LAN, or other network device such as gateway, router, etc. are entered. In addition, their connectivity must be specified, namely which nodes are connected to each other.

2.1.2. Device/transmission medium characteristics

For each device in the network, information on the device throughput capacity such as packets per second (pps) is specified. For each LAN, the type of LAN such as IBM Token Ring, Ethernet, etc., and the transmission medium capacity in Megabits per second (Mbps) is given. NetMod provides the known capacity of the device or LAN as a default value, which the user may replace if desired.

2.1.3. Workload characteristics

The user now enters data on the packet generation rate for each end node type (typically a workstation or mainframe) in packets per second, the average packet size in bits (either averaged over messages and acknowledgements (ACKs), or obtained from some given distribution), and the routing probability for messages arriving at a LAN or network interconnect device. The routing probability specifies the probability that a packet arriving at this node has a destination beyond this node. The average packet generation rate for each workstation defaults to a value that is dependent on the workstation type. The default value can be changed by the user. For all the other nodes the average packet generation rate is computed based on the routing probabilities. The generation of packets at the workstations is assumed to follow a Poisson process.

The hierarchical model of the entire network is decomposed into a number of interrelated submodels each of which is analyzed in greater detail. Typically, four parameters are passed to each submodel:

λ: the mean number of packets per second,
size: the mean size of packets in bits,
size2: the second moment of packet size in bits,
speed: the transmission speed of the medium in bits per second.

In addition, for those submodels that implement bulk arrivals, the mean number of packets per message, c, and its second moment, $c^{(2)}$, is passed. Currently it is assumed (except for the interconnect devices mentioned later) that the buffer capacity for the nodes in the network is practically infinite, i.e. it is sufficiently large that it is not overflowed. Additional features targeted for future implementation include the modeling of finite buffer sizes and flow control mechanisms.

2.2. Output Statistics

The principal output statistics provided by NetMod are the mean delay experienced by a packet that is transmitted across the network (obtained as the sum of mean packet delays in milliseconds across each submodel), and the effective utilization of the transmission media for each submodel. If the submodel is saturated, i.e. unstable, the packet delay function will pass back the word "saturation".

The utilization function displays the utilization of the submodel as a percentage. If the submodel is saturated, this percentage will be a value greater than 100%. Although technically inaccurate, this feature allows the designer to assess the degree to which the submodel is overloaded (e.g. if the utilization of a token ring is shown as 200%, the traffic through that submodel must be cut by at least one half).

2.3. The Submodels

An implicit assumption currently pervading all of the submodels in NetMod is that the arrival of packets to the submodel are Poisson. This assumption is reasonable whenever the arrival stream is a superposition of many independent renewal processes [Lav83]. Such a superposition is typical in the systems being modeled. This assumption will be relaxed where necessary in future versions of NetMod. The following submodel types are implemented in NetMod.

2.3.1. The Polling submodel

The polling submodel is based on a cyclic server queueing system with non–exhaustive service [BoMe86, Srin88] and is used to model the token ring mechanism, as well as some network devices such as NTI's LANSTAR [NTI86]. This submodel is also currently used as an approximation for FDDI under normal load. Determining the throughput (and thereby the utilization) of this polling system is trivial, since the transmission is based on a "deterministic access" mechanism. Thus once a node is allowed to transmit, then assuming that the network is reliable, the packet transmitted by this node is guaranteed to reach its destination. However, calculating the exact mean waiting times at the nodes appears to be extremely difficult in general. A number of approximate analytical models exist for estimating these mean waiting times. The implementation in NetMod uses the expression for mean waiting times developed by Srinivasan [Srin88: equations (12) and (19)] for systems where arrivals occur one at a time, and subsequently extended by Srinivasan et al. [SBTC89] to systems where arrivals can occur in batches. This algorithm appears to be reasonably accurate over a wide range of parameter values, compared to other approximation schemes proposed.

2.3.2. The Ethernet submodel

Ethernet uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol with binary exponential backoff. This is also used for LocalTalk, StarLan, and Sytek networks. Most analyses of CSMA/CD protocols have either treated the nonpersistent version or are too computationally intensive to be handled by an interpretive language such as Excel on a microcomputer. However, there are two treatments that are both accurate and tractable. Hammond and O'Reilly [HaOR86] extend Tobagi and Hunt's analysis [ToHu80] to the 1-persistent case and produce a formula for throughput that compares well with the measured performance of Shoch and Hupp [ShHu80] and compares even better with simulation results. Lam [Lam80] provides a formula for delay (mean time in system until successful transmission) that is tractable and compares quite well to the simulations and measurements obtained by Gonsalves and Tobagi [GoTo88]. In the NetMod implementation, the formula for utilization given by Hammond and O'Reilly [HaOR86: equations (9.82) through (9.84), page 332], and the formula for delay given by Lam [Lam80: equation (6)] are used.

2.3.3. Interconnect device and adapter card submodels

Interconnect devices

The following terminology is used for the interconnect devices:

Bridge - used to connect LANs of the same type to span a larger area. A bridge will typically only transfer data which is not destined for the local LAN. An example is an Apollo bridge between two Apollo rings.

Router - used when two LANs use the same network (or internetworking) protocol, but different physical or link-level protocols. An example is a Proteon router between a Proteon ring and either an Ethernet or token ring.

Gateway - used to store and forward packets between dissimilar networks [Tane88]. It translates protocols between the dissimilar systems. An example would be a Mac-IP gateway between LocalTalk and Ethernet (developed at the Center for Information Technology Integration).

Bridges, routers and gateways are devices which connect LANs in a variety of ways. In NetMod, these devices are considered as variations of a generic interconnect device, which is modeled as an M/D/1/K system: Poisson packet arrivals, a deterministic service time, single server, and finite buffer(K packets). This is a "loss" system in the sense that once the buffer is filled, additional packets that arrive to the node are lost. The service time for the packet at the node depends on the speed of the device and on whether the two LANs being connected are the same or different . If they are different, additional service time is needed for protocol conversion. The basic parameters used in determining the service time are:

1. Throughput capacity as measured from some existing systems in packets per second (pps). The inverse of this throughput capacity determines the "base" service time. (The cpu times for packetizing overheads and protocol conversions are added to this.)
2. Packetizing overhead (cpu time for packet setup and teardown).
3. Protocol conversion for gateways only (cpu time for encapsulation).

Measurements and claims for some representative devices are given below:

Mac-IP gateway: 480 pps for 60 Byte packets (approx. 230 Kbps), and 203 pps for 1320 Byte packets (approx. 2.1 Mbps).

Proteon router claims: 500 pps per direction for packets in the 46 - 1500 Byte (plus 18 Byte header) range. This translates to 256 Kbps - 6.1 Mbps.

DEC LanBridge is reported to take 50 microseconds to process one packet, giving a throughput of 20,000 pps.

Adapter cards

Adapter cards can be modeled in a similar manner as interconnect devices in that measurements are needed for throughput rates (pps and bps) as a function of bus capacity and packet size. As before, the service time includes the cpu time for packetization etc.

3. Using NetMod

There are two versions of NetMod: a dialog–based version using Excel only, and a graphic version using both Hypercard and Excel. Both versions provide a user interface, which presents the user with the following options:

1. Create a new network model
2. Open an existing network model
3. Save a network model to disk
4 Print a diagram of a network model
5. Add a submodel to an existing network model
6. Delete a submodel from a network model
7. Modify a submodel in a network model
8. Execute the model and collect performance statistics
9. Invoke on-line help
10. Platform-specific options (escape to Excel or HyperCard, format cells, etc.)

3.1. Dialogue-based technique - using Excel

The version which uses Excel only starts with a physical diagram of the network on paper (Fig. 1), followed by a description of that network to Excel via dialogue boxes (forms). The model description process is based on the assumption that most of the network can be modeled as a hierarchy of subnetworks, starting with, say, a backbone network down to department-level LANs. The modeler first describes the highest level in the hierarchy and who its immediate children are. He/she is then prompted for information on each child submodel in turn: the equivalent of a breadth-first search of the network hierarchy. A submodel is typically a local area network, e.g. an Ethernet, or a gateway or workstation, but can be any component of the network that might have a significant impact on performance, or that the user might have a special interest in. For example, if part of the network is linked via T1 to the rest of the network, one could model the T1 and associated CSUs (modem-like devices) as a submodel with 1.544 Mbps capacity. If, however, the gateways hooked to that link were known to be slower (e.g. an AppleTalk router), one could just model the gateways. A typical submodel window is shown in Fig. 2.

After the user has described every submodel, he/she can add other submodels that did not fit into the hierarchical model (e.g. a link to a WAN) using the same procedure and dialogues. If it is desired to go back and change some information about a submodel, such as adding a child, or changing the spelling of a name, a Modify menu item will recall the information for that submodel and put up the regular dialogue box with the original information for any necessary changes. Following this, NetMod will rebuild that submodel, and prompt the user for infomation about any new submodels that may have been mentioned as children. A Help menu item brings up a series of Help dialogues similar to the Help facility provided by Excel. These describe the function of all menu items, explain error messages, and give hints on model-building. Fig. 3 shows a snapshot of performance statistics for the network application in Fig.

1 involving a Proteon ring as the backbone, with Ethernets connected to the backbone via Proteon routers.

3.2. Visualization techniques - using HyperCard

The version which uses both HyperCard and Excel uses HyperCard as the front end. This allows the user to draw the network logical diagram on the screen rather than on paper. For the physical network diagram (Fig. 1) drawn manually, Fig. 4 shows the corresponding NetMod logical network diagram created using the HyperCard user interface.

A new network model is initiated by selecting "New Model" from the Options Menu (Fig. 4). New submodels are created by selecting one of the icons from the palette on the left side of the window. These icons can be thought of as a "library" of pre–defined submodel types. Each icon is, in fact, a pop-up menu of alternative submodel types within a class. As more submodel classes are added to the icon palette, the arrows at the top and bottom of the palette will scroll the entire set of icons up or down. The topmost icon in the palette shown in Fig. 4 is a ring icon. If this icon is selected with a mouse, it pops up a menu with entries for FDDI, IBM Token Ring, Apollo Ring, Proteon Ring, etc. If one of these menu items is selected the corresponding submodel button appears on the screen. Each submodel in Netmod represents a generic "class" of networks or nodes. These buttons have a set of attributes such as network type, speed, name, etc. each of which has a default value which may be changed by the user. The submodel type selected will determine the Excel template that is used to model the performance of that submodel. Once the model is transferred to Excel, then the cardinality of each class (i.e., the number of nodes) is input by the user.

In Fig. 1 there are two Ethernets connected to the Proteon Ring. The logical diagram on the screen (Fig. 4) presents these networks as a single class "Building Backbone". If the two Ethernets are not really identical in terms of detailed connections and/or parameter data, then they must be modeled as two separate buttons with different names.

Buttons with icons representing rings, buses, stars, gateways, and workstations may be linked and moved around on the screen. Buttons representing other subnetworks may be used to reduce the clutter on one card; double-clicking the button (see "Departmental LAN" in Fig. 4a) takes the user to the card for that subnetwork (Fig. 4b). Double-clicking on any of the other buttons brings up a form displaying the current attributes of that submodel such as network type, speed, name. etc. which can be changed by the user. A submodel can be deleted by merely dragging it into the trashcan icon.

The cards making up the model may be printed for presentation-quality graphics at the click of a button. A submodel may be moved by positioning the mouse cursor over the submodel, holding down the mouse button and moving the mouse; the submodel icon will follow the mouse around the screen for as long as the button is held down. Once the button is released, any links between that submodel and others are redrawn. To link two submodels together, e.g a network and a workstation, the "connect" button is clicked. Alternately, the "Connect Submodels" can be chosen from the options menu, then the higher-level submodel (the "parent") is clicked followed by the lower one (the "child"). A line will then be drawn between the two icons. Links are broken in a similar manner.

Clicking on the question mark icon brings up an identical card whose buttons describe their function when clicked on. For example, in this new card, clicking on the "Connect" button pops up a message saying "Click on me to connect two submodels together."

3.3. Comparison of computing platforms: HyperCard and Excel

At the present time, HyperCard is primarily used as a presentation platform. Scripts have been written to allow the user to:

* move icons around,
* link them together,
* associate lists of attributes with them and to edit those attributes,
* create new icons and models,
* print a model, and
* transfer the model to Excel for execution and analysis.

Although it is possible to do the calculations in HyperCard, Excel is currently better suited for the job at present, both in speed and in functionality. Excel is mainly used as a calculation engine. The end result of a model-building session is an Excel spreadsheet which uses built-in formulas to capture the response of the network being modeled to changes in its various input parameters. The input parameters that can be most easily changed are workstation characteristics such as mean packet size and rate, as also network characteristics such as speed/capacity and percent of traffic leaving the network, and cardinality relationships (e.g.

number of workstations per department Ethernet, number of routers per campus backbone) between submodels.

A change in any of the input parameters triggers immediate recalculation of all affected output variables. For a medium-sized network model with 10-20 submodels, it takes about 4-8 seconds on a Mac II to calculate network performance. More fundamental changes, such as replacing an Ethernet with a token ring, or adding a new class of workstations to a network, involve the use of additional spreadsheet formulas, a task which is handled by the NetMod code once the user has specified the change in the input parameter section.

In contrast to the HyperCard–based NetMod, where practically all the code deals with the user interface, the majority of the Excel–based NetMod code (about 75%) is concerned with manipulating the formulas making up a network model. This has been changing as increased functionality has been added, with much of the growth occurring in the user interface area.

3.4. Transformation between the platforms

When a model is transferred from HyperCard to Excel, a temporary file is created which contains a small database describing all the submodels in the network model being transferred. For each submodel, a record in the database contains information such as submodel name, type, speed, and routing probability for networks; and mean packet size and rate for workstations. The HyperCard platform invokes the Excel platform which reads in the transfer file and proceeds to build a network model, treating each record in the database as if that information had just been entered by the user. Some sanity checking is also done at this point, e.g. any submodel without children is assumed to have the default workstation attached. This allows use of the same lower-level routines and ensures that the resulting model is identical to that created by entering all the information via the dialogues normally used by the Excel tool alone. The resulting model is ready to execute and analyze, and may be modified by the normal dialogue-based procedure. Alternatively, the user may return to the HyperCard platform, make changes there and retransfer the model to Excel. Future work will make it possible to transfer changes from Excel back to the HyperCard topology. On a Mac II the transfer process takes approximately 30 seconds of overhead plus 20 seconds per submodel; thus a system with 10 submodels takes about 230 seconds (almost 4 minutes) for the transfer.

4. Example Campus Network Analysis

The following example illustrates the capability of NetMod in the design and analysis of a medium-sized campus network with approximately 3500 workstations and personal computers. The campus model is based on a real industrial campus covering several square miles. Our investigation of various industrial, governmental, and academic institutions has revealed a common set of objectives and campus-wide network topology designs that are illustrated in the example shown here.

4.1. Modeling the campus network

A typical campus network model, illustrating the hierarchical network design process with Hypercard, is given in Fig. 1. It contains a single campus backbone (a Proteon Pronet 80 token ring), 40 building backbones (Ethernets), minicomputers connected directly to the building backbones, and personal workstations connected via LANs (Faralon Star Controllers or alternatively, Ethernets). The LANs are connected via routers to the building backbones, which are in turn connected via routers to the campus backbone. Minicomputers service 10 terminals or personal workstations; and the LANs connect 24 personal workstations plus a file server and a print server. The total number of connections for this configuration is 3560 source devices.

Network topology is specified in general by defining the different LAN types, source device types, interconnect device types, and the size (cardinality) of their interconnections. Network device capacity is measured in terms of either LAN speed in Mbps or interconnect device maximum throughput in pps.

Each type of source device (terminal, workstation, or server) can generate a different workload of packets on the system. Workload is characterized by packet size distribution and input rate from a source computer or terminal. The other workload parameter specified here is the routing probability which can be derived from live test data or from user profiles that specify the source and destination of all applications executed by each type of user.

4.2. Types of output statistics

There are two types of output statistics from the campus model: intermediate and final. Intermediate output displays the number of packets per second being processed by the LAN, interconnect device, or source device under steady state conditions. This is useful as a debugging tool for the model because it traces the packet flow from multiple sources to multiple destinations, following the accumulation of multiple sources and routing probabilities through the system.

For instance, in Fig. 1, if each personal workstation connected to the Star Controller generates 4 pps, and each server generates 8 pps, then a total of 24*4 + 2*8 = 112 pps are input to the LAN. If the routing probability is 40% at that LAN (representing a 40% remote access rate), then 40% of 112 packets or 44.8 pps leave the LAN. This LAN is assumed to be symmetric or balanced, so that packets leaving are replaced by packets returning; thus we have 44.8 pps returning from other sources in the network, making the total input to the LAN 112 + 44.8 = 156.8 pps. This final figure is shown in the model intermediate output. If the LAN is asymmetric, for example if it contains a sink for packets (which may be sent to a wide area network without acknowledgement), then a figure of 112 pps would be correct.

Final statistics are shown, in Fig. 5, in terms of LAN and router utilization, throughput in pps, average packet delay at each LAN or interconnect device in milliseconds, and implicitly the average end-to-end delay which is calculated as the sum of the individual average packet delay times. For most of the nodes, bottlenecks are identified when their utilization is near 100%. These are nodes whose throughput is deterministic, based on incoming packets rates. In the case of nodes modeling Ethernet (and its variations), however, such bottlenecks are identified even when the utilization is significantly lower than 100%, but the packet delay is rapidly increasing with workload packet rate and has become the dominant source of delay in the network. This is because Ethernet is affected by packet size and network cable lengths, which influence the number of collisions, and thereby the maximum effective throughput achievable.

4.3. Sensitivity Analysis : Reconfiguration to Eliminate Bottlenecks

In Fig. 5 the bottleneck is clearly at the departmental LAN level (see the extremely large delay time) due to its low throughput capacity and large set of personal workstation and server connections. The most obvious solution to eliminate this bottleneck without reducing the workload is to either replace it with a faster LAN or to reduce the number of connections per LAN, i. e. purchase more LANs for the same total number of connections.

We investigate the first solution, a faster departmental LAN (Ethernet). In Fig. 6 we modify the departmental LAN to reflect the higher speed of Ethernet and the model shows all network components to run at very low utilization. Fig. 7 shows the effect of increasing the global packet workload rate from 4.5 to 54 pps. At this point the departmental LAN is still the bottleneck of the system , but the system is much more balanced: the campus backbone has 17% utilization, the campus router is utilized 21%, the building backbone 23%, and the building router 42%. Although the departmental LAN is utilized only 37%, it is the nearest to saturation because of greater Ethernet sensitivity to cable distance and packet size characteristics. In this case the packet delay is the better indicator of the saturation condition.

NetMod can also produce statistical data in graphical form as shown in Figs. 8 and 9. In these figures we see, respectively, the delay of the various network components as a function of global packet input rate and global (remote) routing probability. Note that as the remote activity increases, the delay at backbone components increase dramatically.

In general, parameters we can vary easily are the global and local packet input rates, packet sizes, and routing probabilities; the topology cardinality values, and the speed of each network component. The topology can also be restructured almost as easily, but requires more than mere change of numbers. The analyst must return to the HyperCard logical network diagram and rearrange the icons to the new topology. NetMod then recalculates the network performance based on the new configuration.

5. Network Model Validations

NetMod has been validated extensively through simulation and measurement. The results of these validations are described below.

5.1. Validation through simulation

In general, it was extremely difficult to measure the end–to–end packet delays and utilizations across interconnected LANs. However, we have made extensive simulations [CST88] which, confirm both the effectiveness of the decomposition approach as well as the accuracy of the analytical expressions used in the submodels. Table I compares the mean packet delays at the various components in a representative interconnected network with over 750 nodes, obtained from simulation with the results obtained from the analytic model. As may be seen from the table, the correspondence between the two results is reasonably good. For more details on this validation, the reader may refer to [CST88].

5.2. Validation through measurement

In addition, we were able to validate the Ethernet model predictions on packet delays and utilizations through extensive measurements. The experimental setup and the test results for this part of the validation are now presented in some detail.

Table I goes here

5.2.1. Ethernet Test System Configuration

A series of network measurement experiments were conducted in order to test the accuracy of some of the analytical models in NetMod, in particular, the Ethernet model [BST89]. The experiments attempted to determine how long it would take to copy a 100 MegaByte (MB) file between two computers on a local area network under varying network load. For this purpose, a number of assumptions were made about the hardware and software used to perform the copy. Experimental data was obtained by performing actual file copies between two Vax computers running Ultrix 3.0 via the Network File System (NFS) implementation. For this exercise, we were not interested in the "theoretical best performance" achievable with special-purpose, highly optimized copying programs, but rather how long a file transfer would take using standard tools under normal conditions and near saturation.

Files were copied from the machine named Ionia to the machine Venice using standard Ultrix file copy commands. Experiments were run under contention-free, normal load, and heavy contention conditions. For the contention-free experiment, Ionia and Venice were on their own separate Ethernet. For the normal load experiment, both machines were attached to an Ethernet that was otherwise in use by our research unit. To achieve the heavy contention conditions, four other workstations were set up to exchange UDP packets at varying rates. With this setup, we achieved an offered load of up to 98%. The experimental setup for the heavy contention case is shown in Fig. 10.

5.2.2. Comparison of Live Test Data with Analytical Tool Predictions

When comparing live test data with results from NetMod we note that for each workstation type we needed to know the rate at which it generates traffic and the average size of the packets generated. To do this the monitoring tool, tcpdump, was used to collect statistics for each run of the experiment. The sending workstation generated an average of 60 packets/sec, with average size 1460 bytes, and the receiving workstation generated an average of 10 packets/sec, all 138 byte acknowledgements. Entering these values into the model, we obtained results for the contention-free case as displayed in Fig. 11.

Fig. 11 gives the delays due to software overhead and network transmission. The delay due to software overhead (termed simply as *software delay*) is estimated by dividing the number of packets making up a 100 MB file by the rate at which they are generated. The delay due to network transmission time (termed the *Ethernet delay*) is estimated by multiplying the per-packet delay calculated in the Ethernet submodel by the number of packets. It is evident from the figure that software delays on the sending and receiving workstations are much greater than delay attributable to transmission across the network for low background traffic conditions. The delay experienced by the file transfer, (which we term as *total delay*) is calculated as max(software delay, Ethernet delay).

Using the model generated by NetMod, we can predict at what point the Ethernet delay will exceed the software delay, which is the point at which we would expect to see an increase in total delay. This occurs with a background traffic of 1034 packets/sec (Fig. 12) which corresponds to a load of 8.7 Mbps (accounting for 87% of the network capacity). The file transfer from Ionia to Venice corresponds to a load of 0.7 Mbps (which is 7% of the network capacity).

Fig. 13 contains a graph comparing the predictions of the model to the measured delay times for the 20 MB file copy. As we can see, the model is in reasonably close agreement with the measurements for the cases of low, medium and high loading of the network. However, it is a little off actual measurement around the region where the transmission delay begins to overtake the software delay. It appears that better accuracy at high loads will require an accounting of the interplay between the transmission delay and the software delay.

6. Conclusions / Future Work

The Network Modeling Tool (NetMod) has been shown to be a very pragmatic tool for the design and analysis of medium to large interconnected campus networks, involving hundreds or even up to tens of thousands of computer workstations. NetMod forces the system analyst to thoroughly understand the device interrelationships in the network and design an initial configuration which can be analyzed. NetMod also provides basic statistics for the most idealistic case, commensurate to traveling in a car between two cities and only having to cope with speed limits and traffic signals with normal traffic conditions, so that bottlenecks detected by NetMod for the ideal case will occur in a real system with absolute certainty.

To obtain the desired performance measures, NetMod uses simple analytic formulae that can be readily programmed in Excel. Currently, all the formulae are expressed in closed form. This has been possible due to two factors.

One factor is the assumption of Poisson arrivals to every submodel. As noted earlier, this assumption is reasonable whenever the input is a superposition of many independent renewal processes; this is typical in the systems being modeled. A number of simulation studies indicate [CST88] that the Poisson approximation is quite reasonable. Also, a number of measurement studies undertaken in-house [Wei89] showed that the departure process from the Ethernet gateway was very nearly Poisson. Although one cannot infer anything from just one set of measurements on a specific installation, these findings do lend support to the Poisson assumption. The assumption of Poisson arrivals usually leads to simple analytic expressions.

The second factor is due to the fact that currently we have only modeled the lowest two layers in the OSI hierarchy (up to the MAC layer, only). We are currently working on enhancing the tool to include the layers up to the transport layer. These additional enhancements are expected to result in analytic formula that are not always in closed form. We are working on some iterative schemes for some of the submodels and higher network layers based on G/G/1 approximations [SrTe89]. In addition, we expect to interface, through external calls from within either Hypercard or Excel, some of the simulation models that we have developed for the higher network layers.

A number of analytical formulae have been developed in-house for the tool. For example, motivated by the fact that in real systems, the arrivals are messages which are broken up into one or more packets, we have developed accurate approximations for cyclic server systems with non-exhaustive service fed by simple/compound Poisson arrivals [Srin88, SBTC89].

The performance predictions made by NetMod have been tested for their accuracy primarily through simulation, although in the case of Ethernet they have been validated through actual measurements. However, it has been extremely difficult to validate NetMod for large interconnected networks through actual measurements, and this remains a topic for future work.

The tool described in this paper uses a combination of Hypercard and Excel. A new version of the tool has been developed, which uses only Hypercard. This version is currently being tested. Future versions of the tool will introduce details of network operating system algorithms and higher layer network protocols (LLC, network, and transport layers in the OSI model or equivalent) and network operating system overhead.

NetMod has been field tested with a variety of university and industrial campuses and has been found to be extremely useful in the initial design stages, particularly in feasibility studies. Later versions of the tool will assist in the detailed design phase that would follow when some of the network components have already been purchased.

Acknowledgements

The authors gratefully acknowledge the programming support from Tom Unger and critical review from Peter Honeyman. Guangping Wei provided live test data for packet input rates, packet sizes, and routing probabilities; Bruce Hebbard designed the large-scale model for the University of Michigan campus. This work was entirely sponsored by Northern Telecom Inc.

References

[BST89] D. W. Bachmann, M. E. Segal, and T. J. Teorey "On the performance of copying large files across a contention-based network," submitted to *IEEE Network*.

[BoMe86] O.J. Boxma and B. Meister, "Waiting Time Approximations for Cyclic-Service Systems with Switch-over Times," *Performance Evaluation Review*, vol 14, pp. 254-262.

[CST88] J. Chiarawongse, M.M. Srinivasan, and T.J. Teorey, "Performance Analysis of a Large Interconnected Network by Decomposition Techniques," *IEEE Network*, Vol. 2, No. 4, July 1988, pp. 19-27.

[Frost88] V. Frost, "Block-Oriented Network Simulator," presented at 2nd IEEE Workshop on CAMAD of Comm. Links and Networks.

[Garr88] W.J. Garrison, "Network II.5 tutorial - Network II.5 - Without programming," *Proceedings 1988 Winter Simulation Conference*, pp. 152-158.

[GMW86] R.F. Gordon, E.A.MacNair, and P.D.Welch, "Examples of Using the Research Queueing Package Modeling Environment (RESQME)," *Proceedings 1986 Winter Simulation Conference*, pp. 404-503.

[GoTo88] T.A. Gonsalves and F.A. Tobagi, "On the Performance Effects of Station Locations and Access Protocol Paramters in Ethernet Networks," *IEEE Trans. Commun.*, Vol. COM-36, No.4, April 1988, pp.441-449.

[HaOR86] J.L. Hammond and P.J.P. O'Reilly, "Protocols That Use Carrier Sensing With Collision Detection," *Performance Analysis of Local Computer Networks*, Addison-Wesley, 1986, pp.320-340.

[Lam80] S.S. Lam, "A Carrier Sense Multiple Access Protocol for Local Networks," *Computer Networks*, Vol. 4, No.1, February 1980, pp.21-32.

[Lav83] S.S. Lavenberg, "Mathematical Prerequisites," *Computer Performance Modeling Handbook*, Academic Press, New York 1983.

[MBCC84] M.A. Marsan, G. Balbo, G. Ciardo, and G. Conte, "A Software Tool for the Automatic Analysis of Generalized Stochastic Petri Nets Models," *Proceedings International Conference on Modelling Techniques and Tools for Performance Analysis*, May 1984, pp. 155-170.

[MeMo85] B.Melamed and R.J.T.Morris, "Visual Simulation: The Performance Analysis Workstation," *IEEE Computer*, Aug. 1985, pp. 87-94.

[Muel84] B. Mueller, "NUMAS: A Tool for the Numerical Modelling of Computer Systems," *Proceedings International Conference on Modelling Techniques and Tools for Performance Analysis*, May 1984, pp. 141-154.

[NMMT84] T. Nishida, M. Murata, H. Miyahara, and K. Takashima, "PLANS: Modelling and Simulation System for LAN," *Proceedings International Conference on Modelling Techniques and Tools for Performance Analysis*, May 1984, pp. 235-258.

[NTI86] Northern Telecom, Inc., "Meridian SL1 Integrated Services Network - LANSTAR Data Services: Usage and Connectivity," Dallas, TX, 1986.

[SaMe87] W.H. Sanders and J.F. Meyer, "Performability Evaluation of Distributed Systems using Stochastic Activity Networks," *IEEE International Workshop on Petri Nets and Performance Models*, pp. 111-120.

[ShHu80] J.F. Shoch, J.A. Hupp, "Measured Performance of an Ethernet Local Network," *Communications of the ACM*, Vol.23, No.12, Dec. 1980, pp.711-721.

[SDM85] J.B. Sinclair, K.A. Doshi, and S. Madala, "Computer Performance Evaluation with GIST: A Tool for Specifying Extended Queueing Network Models," *Proceedings 1985 Winter Simulation Conference*, pp. 290-299.

[Srin88] M. M. Srinivasan, "An Approximation for Mean Waiting Times in Cyclic Server Systems with Nonexhaustive Service," *Performance Evaluation* 9, 1988, pp. 17-33.

[SrTe89] M. M. Srinivasan, and T. J. Teorey, "Hierarchical Modeling of Interconnected Local Area Networks," CITI Technical Report 89-1, The University of Michigan, Ann Arbor, MI, February 1989.

[SBTC89] M. M. Srinivasan, D. W. Bachmann, T. J. Teorey, and J. Chiarawongse, "Mean Waiting Times for Cyclic Service Systems with Batch Poisson Arrivals: The Non-Exhaustive Service System," CITI Technical Report 89-6, The University of Michigan, Ann Arbor, MI, July 1989.

[SWM86] K.L. Stanwood, L.N. Waller, and G. C. Marr, "System Iconic Modeling Facility," *Proceedings 1986 Winter Simulation Conference*, pp. 531-536.

[SuTo88] R. Suri, and M. Tomsicek, "Rapid Modeling Tools for Manufacturing Simulation and Analysis," *Proceedings 1986 Winter Simulation Conference*, pp. 25-32.

[Tane88] A. S. Tanenbaum, *Computer Networks*, 2nd Edition, Prentice Hall, Englewood Cliffs, New Jersey 1988.

[ToHu80] F.A. Tobagi and V.B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," *Computer Networks*, Vol. 4, No.5, Oct-Nov 1980, pp.245-259.

[VePo84] M. Veran and D. Potier, "QNAP2: A Portable Environment for Queueing Systems Modelling," *Proceedings International Conference on Modelling Techniques and Tools for Performance Analysis*, May 1984, pp. 25-64.

[Wei89] G. Wei, "Measurement studies on Ethernets," Unpublished technical notes.

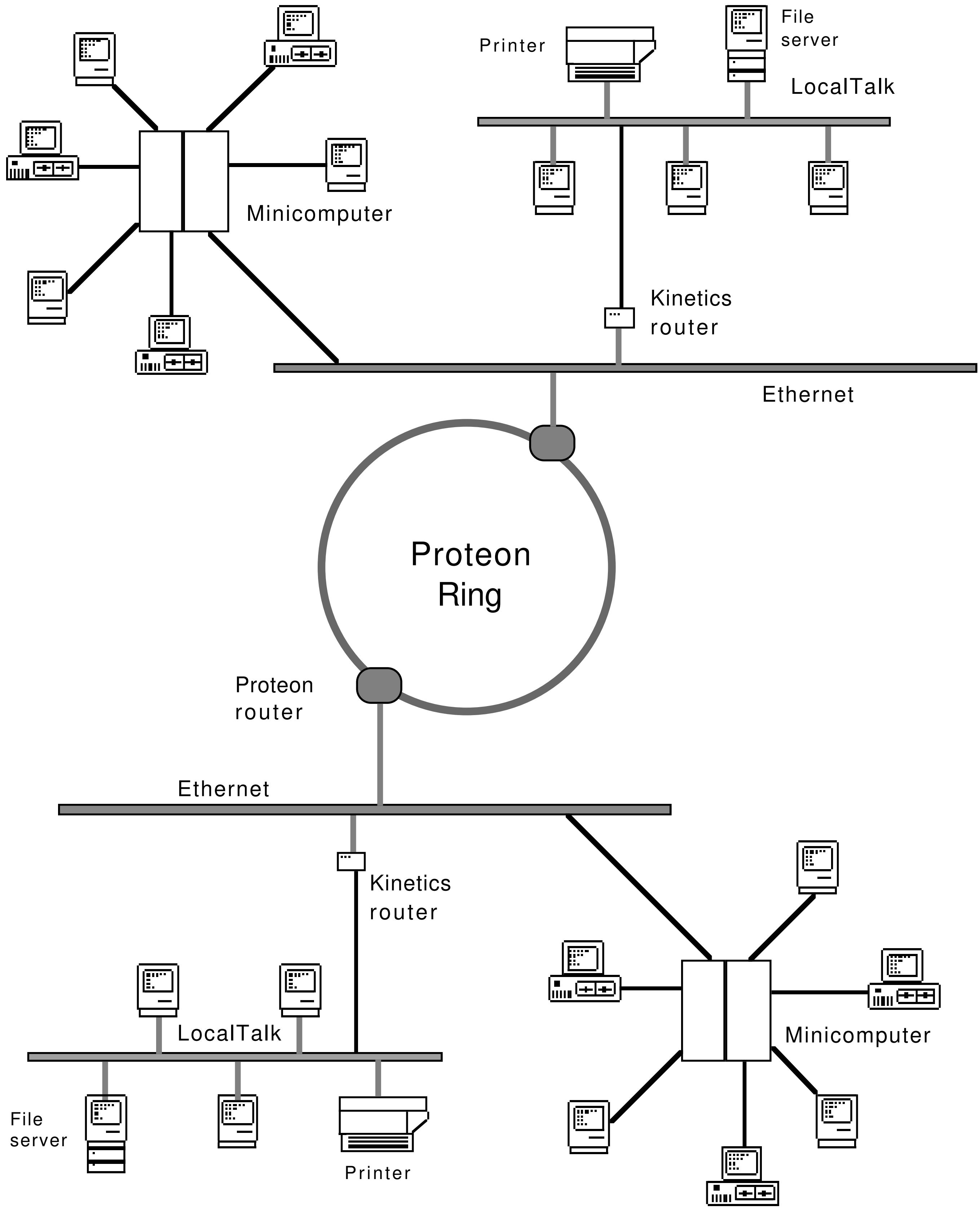


Fig. 1. Physical network diagram

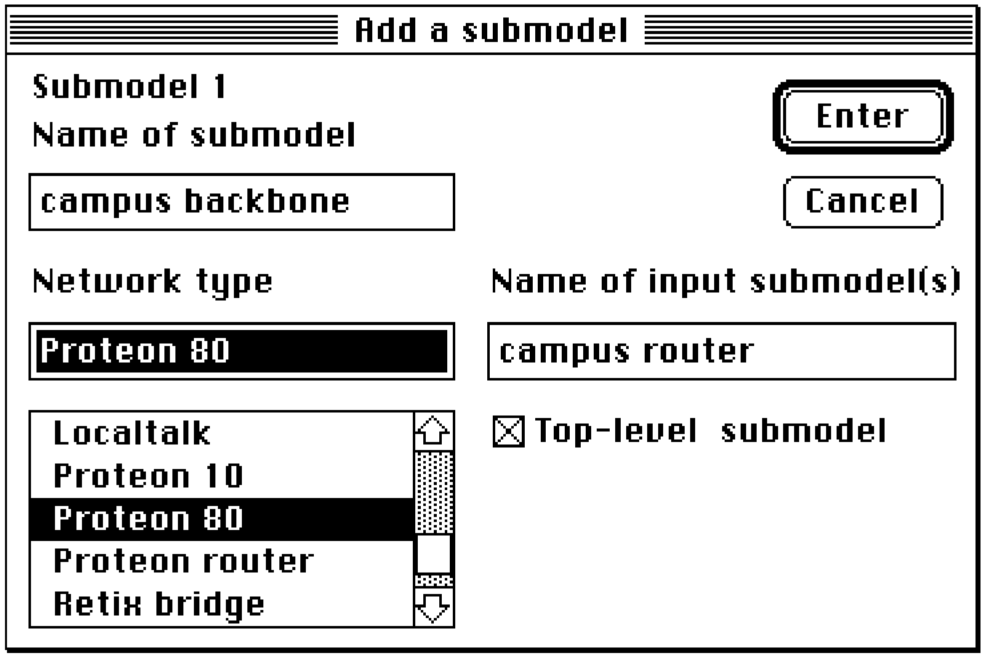


Fig. 2. - Submodel definition dialogue

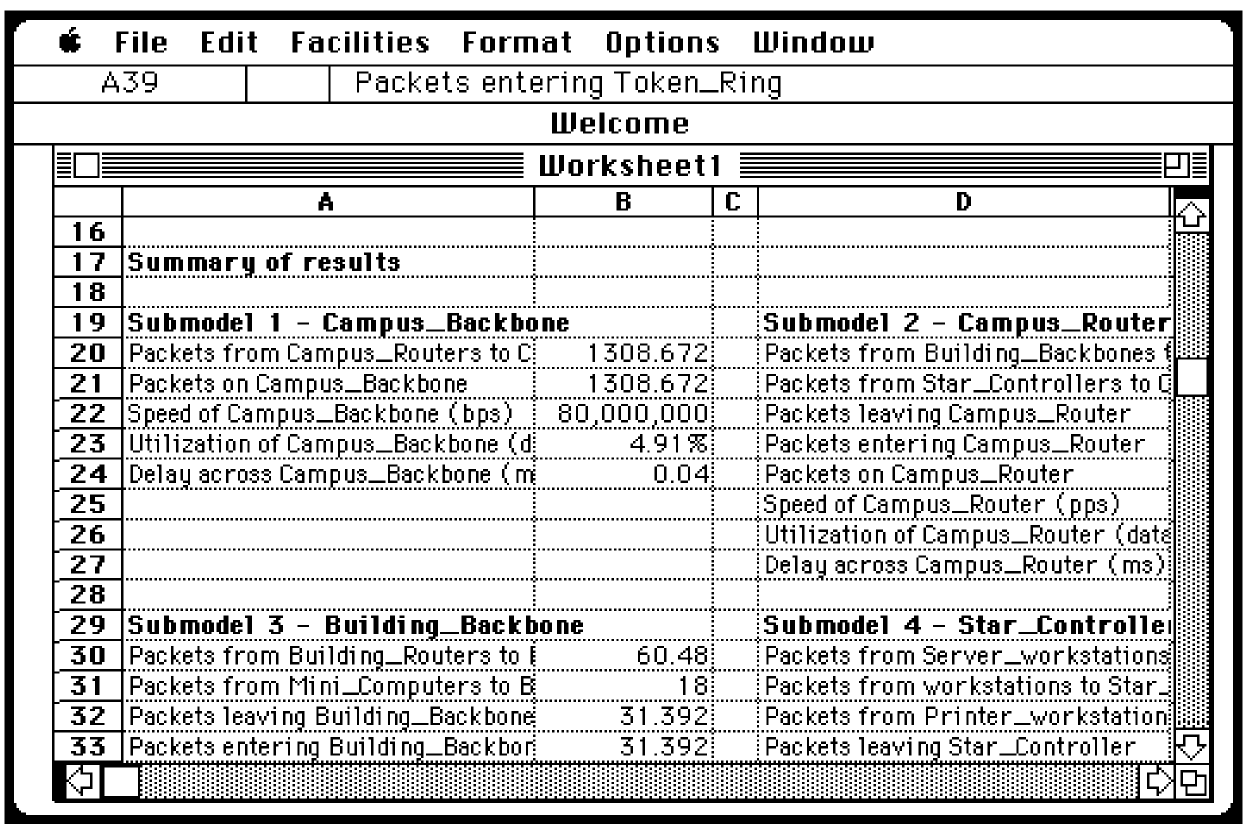
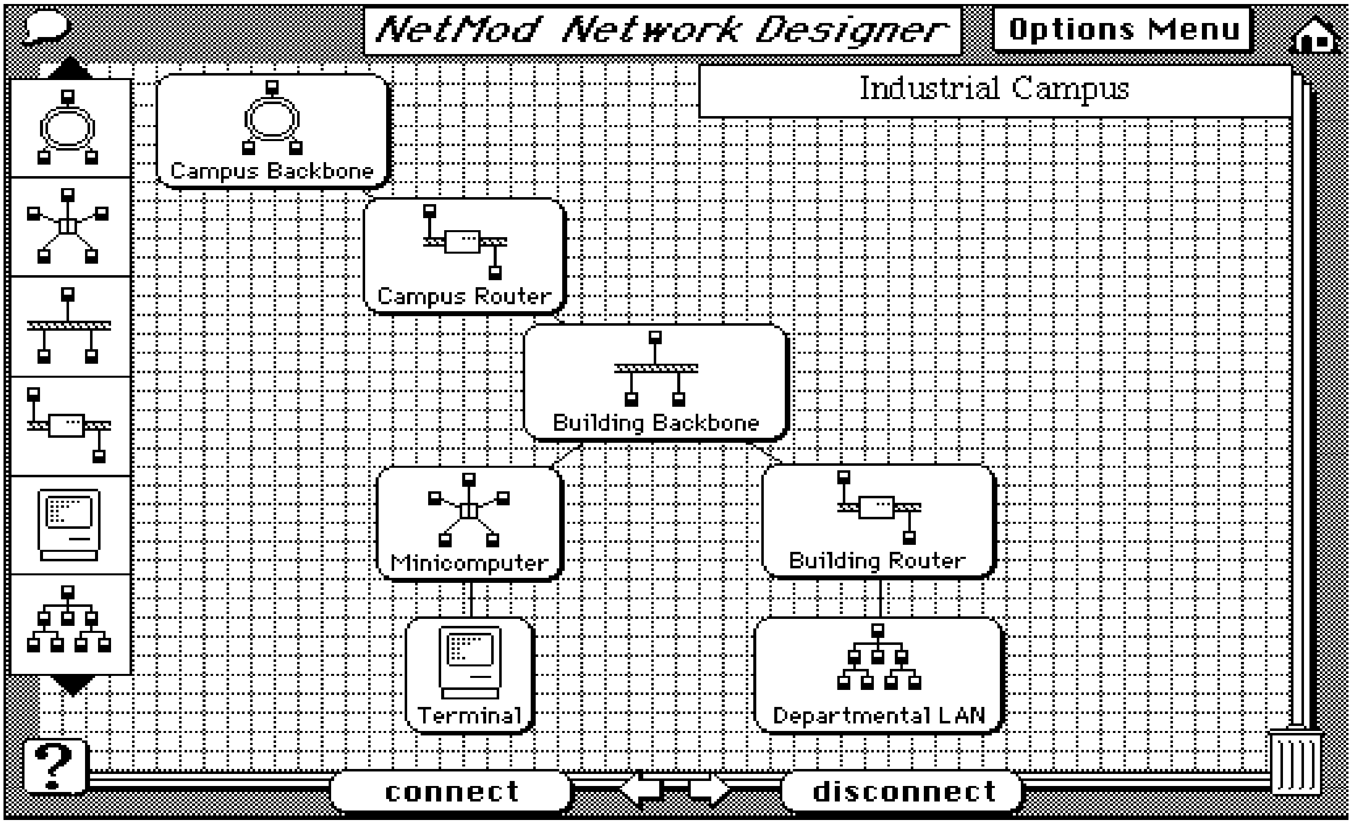
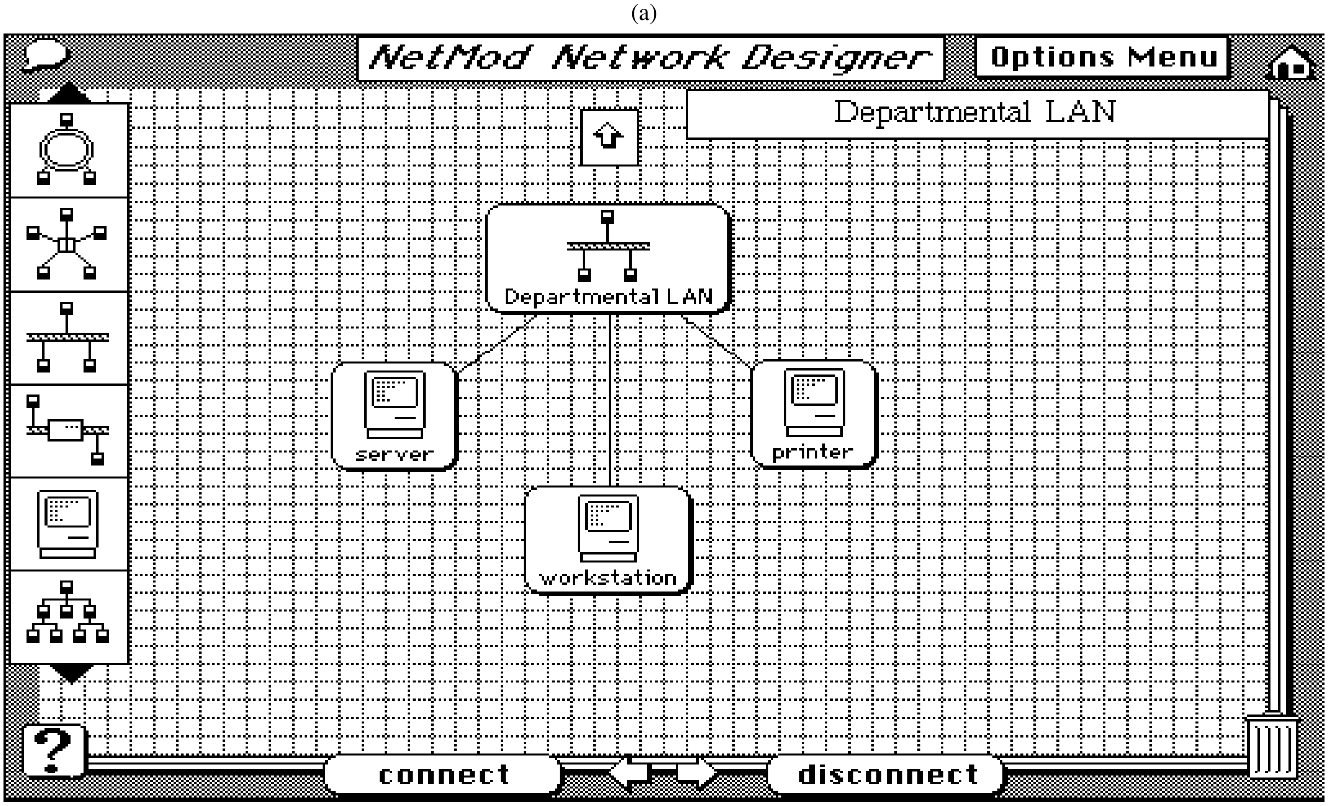


Fig. 3. - Excel display of part of network model



(a)



(b)

Fig. 4. NetMod logical network diagram

Fig. 5. Spreadsheet with initial topology

Fig. 6. Department LAN changed to Ethernet

Fig. 7. New saturation point

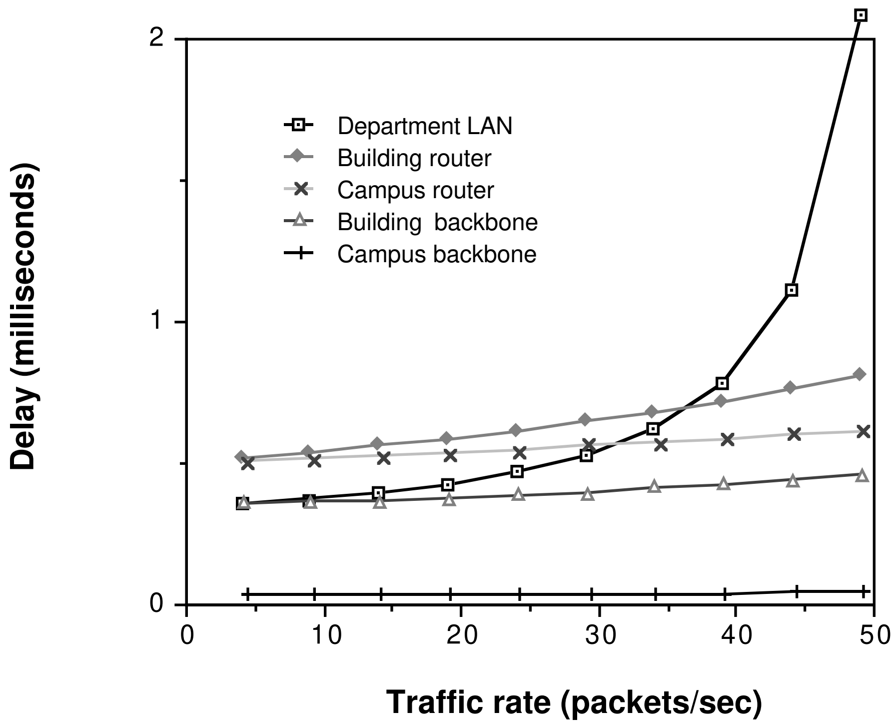


Fig. 8. Effect on delay of global packet rate

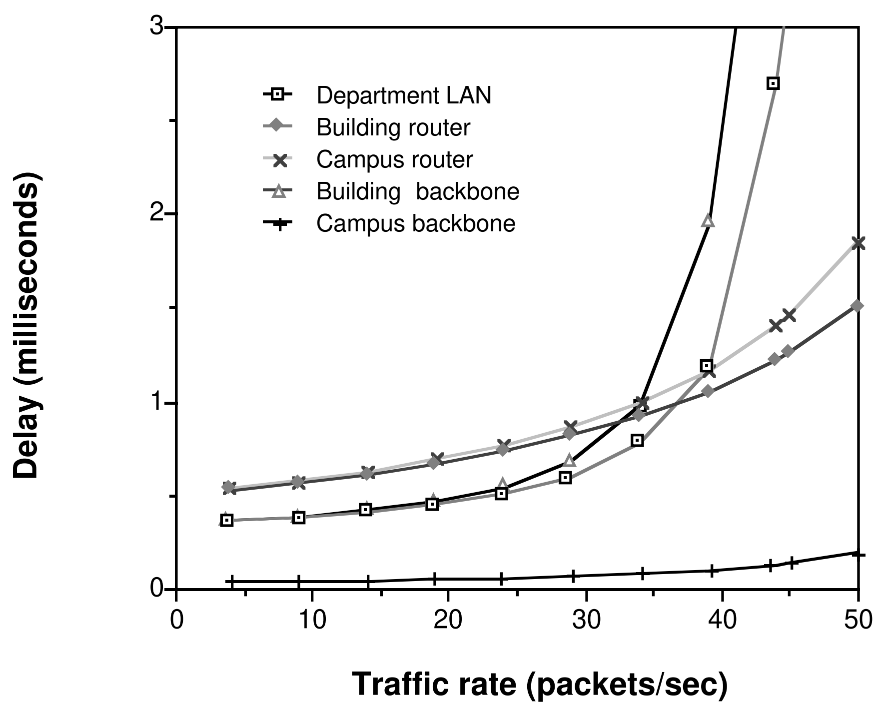


Fig. 9. Effect of changing routing probability to 60%

Table I

Response Times for Simulation and Analytical Models for Bimodal Packet Size Distribution

System Component	Simulation Global Model (ms)	Analytical Submodel (ms)
Apollo		
Nongateway	0.40	0.40
Gateway	0.40	0.40
LANIF		
Uplink, Apollo	28.85	26.41
Uplink, WS	4.14	4.21
Downlink, WS	4.30	3.28
PTE bus		
LANIF-to-Apollo	0.20	0.14
LANIF-to-WS	0.21	0.14
From fiber ring	0.17	0.16
Fiber ring	0.05	0.04

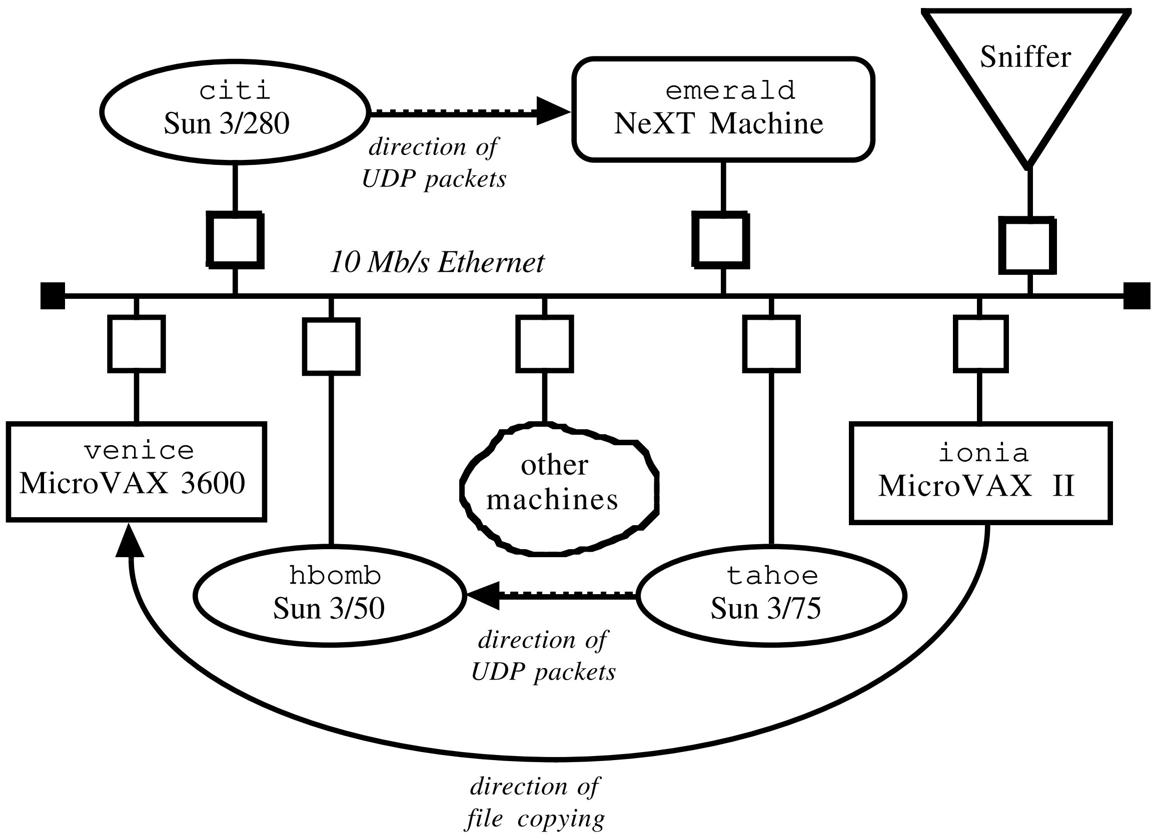


Fig. 10. Experimental setup

new file transfer

	A	B	C	D	E
1	**File Transfer Problem**				**5/5/89**
2	Models transfer of large file over ethernet				
3	Includes background traffic				
4					
5	**Default workstation parameters**			**Topology**	
6	Packet rate (pps per ws)	60		sending_workstations per Local_	1
7	Packet size (bits)	11,680		receiving_workstations per Loca	1
8				background_workstations per Lo	1
9					
10	**Results**				
11	Per-packet delay (ms)	1.06671584		Background traffic	0
12	Total number of packets	68493.1507		Total traffic on Local Ethernet	711,840
13	Ethernet delay (packets*delay)	73.0627286			
14	Software delay (packets/rate)	1141.55251			
15	Total delay in seconds	1141.55251			
16					
17					
18	**Submodel 1 - Local_Ethernet**			**Submodel 2 - sending_workstation**	
19	Packets from sending_workstati	60		Packets leaving sending_workst	60
20	Packets from receiving_workst	10		sending_workstation mean packe	11680
21	Packets from background_works	0			
22	Packets to Local_Ethernet	70		**Submodel 3 - receiving_workstation**	
23	Packets on Local_Ethernet	70		Packets leaving receiving_works	10
24	Local_Ethernet mean packet size	10169.1429		receiving_workstation mean pac	1104
25	Speed of Local_Ethernet (bps)	10000000			
26	Utilization of Local_Ethernet (da	6.64%		**Submodel 4 - background_workstation**	
27	Utilization (data & collisions)	6.64%		Packets leaving background_wor	0
28	Effective throughput of Local_Et	65.30		background_workstation mean p	8376
29	Delay across Local_Ethernet (ms	1.07			

Fig. 11. File transfer with no background traffic

new file transfer

	A	B	C	D	E
1	**File Transfer Problem**				**5/5/89**
2	Models transfer of large file over ethernet				
3	Includes background traffic				
4					
5	**Default workstation parameters**			**Topology**	
6	Packet rate (pps per ws)	60		sending_workstations per Local_	1
7	Packet size (bits)	11,680		receiving_workstations per Loca	1
8				background_workstations per Lo	1
9					
10	**Results**				
11	Per-packet delay (ms)	17.1082081		Background traffic	8,660,784
12	Total number of packets	68493.1507		Total traffic on Local Ethernet	9,372,624
13	Ethernet delay (packets*delay)	1171.79507			
14	Software delay (packets/rate)	1141.55251			
15	Total delay in seconds	1171.79507			
16					
17					
18	**Submodel 1 - Local_Ethernet**			**Submodel 2 - sending_workstation**	
19	Packets from sending_workstati	60		Packets leaving sending_workst	60
20	Packets from receiving_workst	10		sending_workstation mean packe	11680
21	Packets from background_works	1034			
22	Packets to Local_Ethernet	1104		**Submodel 3 - receiving_workstation**	
23	Packets on Local_Ethernet	1104		Packets leaving receiving_works	10
24	Local_Ethernet mean packet size	8489.69565		receiving_workstation mean pac	1104
25	Speed of Local_Ethernet (bps)	10000000			
26	Utilization of Local_Ethernet (da	48.10%		**Submodel 4 - background_workstation**	
27	Utilization (data & collisions)	48.10%		Packets leaving background_wor	1034
28	Effective throughput of Local_Et	566.52		background_workstation mean pa	8376
29	Delay across Local_Ethernet (ms	17.11			

Fig. 12. Background traffic first affects transfer time

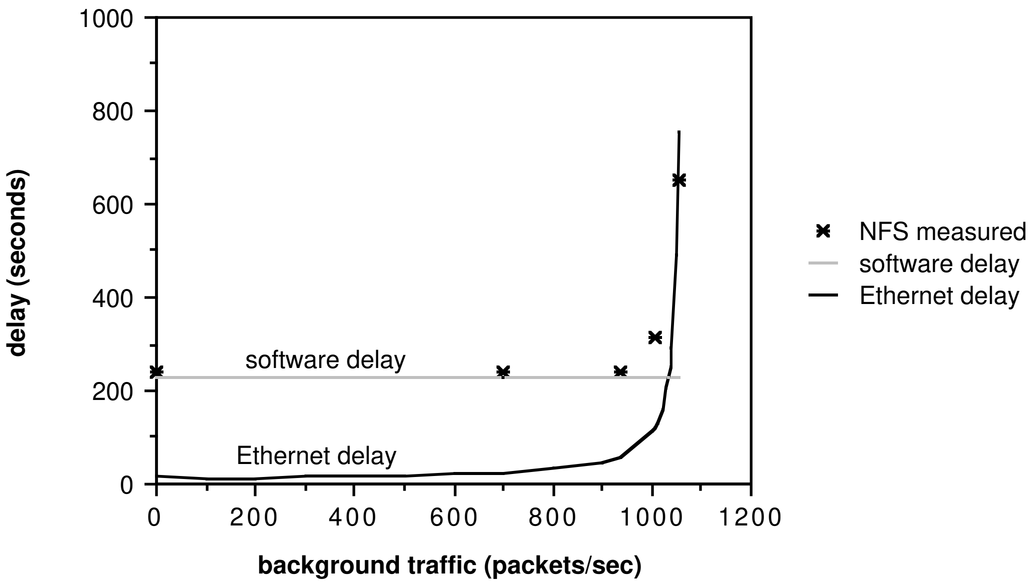


Fig. 13. Comparison of predicted to measured delay