A Computer Network Monitoring System

DAVID E. MORGAN, MEMBER, IEEE, WALTER BANKS, DALE P. GOODSPEED, STUDENT MEMBER, IEEE, AND RICHARD KOLANKO, STUDENT MEMBER, IEEE

Abstract-In order to help satisfy'an apparent need for tools for monitoring the activities of a computer network (see Mamrak [1]), a system of special hardware and software, called a Computer Network Monitoring System (CNMS), is being implemented in the' University of Waterloo Computer Communications Networks Group (CCNG). The'paper discusses the motivation and derivation of the CNMS, then provides functional descriptions of most of the major hardware and software components, illustrates use of the CNMS, and lists experiments and applications. In a previous paper [2l, the motivation and architecture of the system were sketched.

The CNMS consists of: 1) a set of hybrid monitors, each of which is controlled by a locally or remotely located computer; 2) monitor control and data analysis software; 3) a network traffic generator; 4) measurement software in each computer monitored. Each com-' puter to be monitored is attached to a monitor. Telephone lines, possibly different from those of the network, connect the monitors to the controlling computer.

Index Terms-Accounting for computer usage, computer instrumentation, computer networks, computer performance measurement, computer reliability.

I. INTRODUCTION

A COMPUTER system, consisting of hardware and the software to control it, is often so complex that it is difficult to understand what is being done, how efficiently it is being done, and what problems exist. Moreover, computer systems are often connected to other computers as well as terminals to form even more complex computer networks.

Cole and others have defined a network of computers to consist of two or more computers linked together, while a computer network has been defined to be either a network of computers or a set of terminals connected to one or more computers [3]. Most networks of computers consist primarily of nodes, hosts, transmission links, and terminals. A node (in this context) usually refers to a computer used primarily to switch data. A computer whose primary role is not switching data in the network to which it is attached, is called a host. In some networks, a sharp distinction is made between nodes and hosts, while in others no distinction exists. Terminals are devices which serve as the interface between man and the computer. The transmission links, of course, join this collection of hardware together to form a network.

Determination of what a computer system or network is doing is essential to effective management of it. This involves monitoring (observing) its behavior as it executes a set of programs and responds to its environment. The behavior of a system or network acting on a set of programs and data is characterized by the sequence of values of certain parameters of the system and by the sequence of events that occur as the system executes. These are manifestations of the sequence of states traversed by the system as programs of the workload are executed.

Although a variety of hardware and software monitoring tools and techniques have been developed to aid in observing the behavior of computer systems (see, for example, [4]-[14J), little attention has been paid to developing tools for monitoring computer networks. Kleinrock and Cole [3J have successfully used elegant software techniques to monitor the performance of the Advanced Research Projects Agency (ARPA) network. Abrams *et al.* at the National Bureau of Standards of the United States, have developed tools for observing data flowing along a communications line between a computer and a terminal [15]. Fuller and others are instrumenting the C.mmp multi-mini processor system.

The purpose of this paper is to discuss the motivation, architecture, components, and use of a system of hardware and software designed to monitor the behavior of a computer network or system. Morgan, Banks, and others have previously sketched the purpose and architecture of the Computer Network Monitoring System (CNMS) [2J, [16]. This CNMS is being created in the Computer Communications Networks Group at the University of Waterloo, Waterloo, Ont., Canada.

The paper is organized in six sections. Section II motivates the need for a monitoring system rather than a set of unrelated, uncoordinated software and hardware tools. By considering the problems involved in monitoring a computer network, the section motivates characteristics and major components of an ideal computer network monitoring system. Section III presents the architecture and describes the components of the CNMS being created at the University of Waterloo. Use of this CNMS is explained and illustrated in Section IV. Section V lists some experiments being performed using the CNMS, and. mentions some potential applications of the system. Section VI summarizes this research, evaluates the CNMS in terms of nine characteristics presented in Section II, presents our conclusions so far, and outlines some future work.

Manuscript received July 15, 1974; revised March 11, 1975. This research was supported by the Department of Communications of Canada under Research Contract SP2-36100-3-0406; the Defense Research Board of Canada under Grant 9931-37; the National Research Council of Canada under Grant A8116, and by Mordata, Ltd. The research was performed in the Computer Communications Networks Group's laboratory at the University of Waterloo, Waterloo, Ont., Canada.

The authors are with the Computer Communications Networks Group and the Department of Applied Analysis and Computer Science, University of Waterloo, Waterloo, Ont., Canada.

II. NETWORK MONITORING SYSTEM MOTIVATION

There are four fundamental reasons for monitoring a computer system or network:

1) to observe its performance, thereby determining whether work is flowing satisfactorily through it;
2) to detect malfunctions;
3) to diagnose the causes of any problems observed; and
4) to measure the resources used so that appropriate charges can be made.

Usually the people who wish to monitor the activities of a computer system are neither hardware, software, nor statistics experts. Rather, they are either managers responsible for the system, software maintenance people who lack detailed knowledge of hardware, hardware maintenance personnel who lack detailed knowledge of software, or researchers (students or professionals) seeking statistics for their work. It is indeed rare that the person seeking information is a computer software, hardware, and statistics expert all in one. Thus, system monitoring tools should be easy to learn to use as well as easy to use, and detailed knowledge of hardware, software, and/or statistics should not be necessary to observe the system and get useful information about it. Furthermore, the tools and techniques should be incorporated in a single monitoring system to avoid having to learn to use several different tools and techniques.

Computer networks are often distributed geographically, so monitoring the behavior of a computer network involves distributing the monitoring activities across the network. In order to correlate observations from scattered sites, these monitoring activities must be centrally controlled and coordinated, and the results must be centrally analyzed. Thus, monitoring a computer network involves communications as well as monitoring. Network monitoring tools and techniques must be designed with this in mind.

Although software has been used with some success to monitor the performance of computer systems and networks, experience indicates that monitoring software without hardware often perturbs the system or network unsatisfactorily. Hardware monitors without software aid are too inflexible for most network applications. Certain parameters, events and their attributes can best be determined by software, while others can best be determined by hardware. Thus, some combination of hardware and software monitoring tools appears better than either hardware of software tools alone. Moreover, if a computer network is to be monitored, either the computers in the network could include special hardware to aid them in self-monitoring while producing minimal interference with the network's functions, or a set of software-controlled hardware monitors (often called "hybrid monitors"), one attached to each computer to be monitored, could be employed to complement necessary monitoring software in each computer. Each of these software-controlled monitors should be capable of having its activities controlled and its monitored results sent through telecommunications links. Control of these monitors and analysis of the data could be performed by a computer system at a Network Monitoring Center (NMC), such as the ARPA Network's NMC, which is used to coordinate software measurement activities at each IMP and to collect and analyze the data.

Because transmission of large volumes of data is expensive and usually not necessary, a network monitoring system needs facilities for reducing data before transmission to the NMC. Cole [3] showed that good approximations to many kinds of distribution functions could be obtained from log histograms. Much of the measurement data obtained by the ARPA network measurement software is transmitted in the form of log histograms to reduce the amount of data transmitted. Extending Cole's reasoning, only the first four moments of a distribution are required to model its behavior for most purposes. Rather than produce these moments at the NMC from data transmitted from remotely located monitors, hardware could be provided in each remotely located monitor to produce these moments, then transmit only the moments to the NMC, thereby reducing the data that must be transmitted. The main disadvantage is the cost of such data reduction equipment.

The activities and parameters of a system or network can be monitored continuously, periodically on a sampling basis, or only when events of interest occur. An event usually consists of a logical and/or sequential combination of other events. Fundamentally, an event is the occurrence of a specific pattern or sequence of patterns in particular portion(s) of the system, network, environment, or workload. Tools for monitoring events should include facilities:

1) to detect specified event(s);
2) to register the (real or relative) time of occurrence of the event;
3) to time the duration of the event (i.e., the set of states comprising the event, or bounded by two particular events) and/or its consequences;
4) to obtain and record selected attributes of the system, workload, and/or environment when the event occurred;
5) to count the number of occurrences of the event; and
6) to initiate some action as a consequence of the event, e.g., diagnosing the cause of a problem defined by the occurrence of the event, checking for any damage, or initiating repair and/or recovery activities.

A well-known problem in physics and astronomy is to determine the order in which nearly simultaneous events occur in widely separated systems. The same problem occurs when monitoring a network of computers. One way to minimize such problems is to synchronize all the clocks as accurately as possible with a single, very accurate, reliable source of time-of-day readings such as that provided by the National Bureau of Standards of the United States or the National Research Council of Canada.

In order to determine the effects that changes have on the performance of the network, some way of controlling the workload applied to the network is desirable. Thus, a monitoring system would be more useful if it included facilities to apply loads with specified characteristics to the object system or network.

From the above discussion, it follows that an ideal CNMS should include the hardware and/or software tools. necessary:

1) to observe, measure, record, and evaluate the behavior of each of the components of a computer network (including its workload and environment), such tools being:
   a) a set of computer monitoring systems, each having hardware and/or software capable of detecting particular events, measuring their attributes, recording and reducing the data, and transmitting the data for analysis elsewhere;
   b) terminal and/or telecommunications link monitors, each having hardware and/or software to observe activities and information flow;
2) to define, control, and coordinate monitoring activities throughout the network;
3) to provide a single, accurate, reliable source of time (e.g., time of day readings and precise intervals) for the entire CNMS; and
4) to provide network traffic with known characteristics, should the CNMS user need this capability for monitoring purposes.

To the best of our knowledge, no such computer network monitoring system has yet been developed. However, a number of hardware monitors and software monitors, plus a few hybrid monitors (e.g., [17]-[19J) have been developed for computer systems, and software techniques and tools havebeen used by Kleinrock, Cole, and others to monitor computer networks [3].

Experience in monitoring computer systems, and study of pertinent literature indicate that an ideal computer network monitoring system (CNMS) should possess the following characteristics:

1) be easy to use, yet flexible and expandable;
2) be as system independent as practical;
3) be dependable and easily diagnosed;
4) allow gathering of measurement data at a distance from the monitor control and analysis functions, with minimal human intervention required;
5) span the network;
6) interfere minimally with the performance and m-tegrity of the measured systems;
7) interfere minimally with computer-computer and terminal-computer communications;
8) have no ill effects on the security or integrity of any of the systems;
9) offer a choice of resolution, so that the unit of measure fits what is measured; and
10) be low in cost while not compromising the other goals.

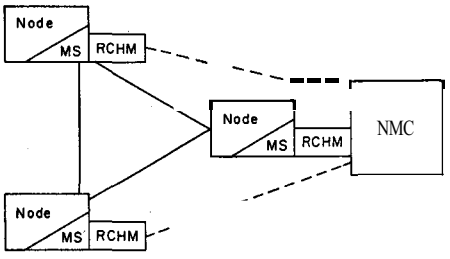


Fig. 1. Measurement software (MS), remote computer controlled hardwar.e monitor (RCHM), regional network measurement center (RNMC), and network measurement center (NMC).

It is, of course, impossible to achieve the ideal CNMS. Nevertheless, this paper describes an attempt to produce a system that hopefully will accomplish many of these goals to a reasonable degree. In Section VI, the CNMS described in Section III is evaluated in terms of these characteristics of an ideal CNMS.

III. DESCRIPTION OF A CNMS

A. System Architecture

Using the characteristics listed in Section II as goals, a CNMS has been designed and is being developed. A prototype system has been implemented and is being tested by using it to monitor two types of minicomputers and two small laboratory networks. This CNMS consists of the following major components, which correspond with the list of tools needed that is given in Section II:

1) a set of hybrid monitors (called remote controlled hybrid monitors, and abbreviated RCHM), each being controlled by a computer which can be miles away (i.e., at the NMC), and containing components to enable it to monitor a computer system and a set of communications links to terminals or other computers;
2) software to define, control, and coordinate the activities of a set of hardware monitors, and to obtain and analyze data from them;
3) monitoring software in each observed computer, and tools to enable the activities of the monitoring software to be controlled and coordinated from the NMC;
4) facilities in each hardware monitor to gain access to a single standard clock for time-of-day readings as well as precise intervals; and
5) a network traffic (load) generator capable of simulating the activities of several users (human or nonhuman) interacting with the network.

Fig. 1 shows how the components of this CNMS can be configured to monitor a network. Note that a telephone line, which may be physically or logically separate from the data links of the network, can connect each monitor to the computer controlling it, or the monitor can be attached directly to the controlling computer. These telephone connections need not remain establishedthroughout a monitoring session. Computer control is required only to set up the experiments, to read accumulated data

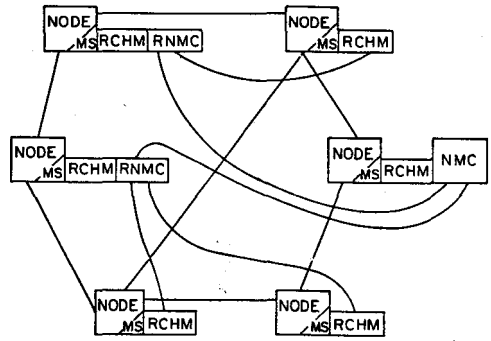


Fig. 2. Network hardware monitors attached to a computer network.

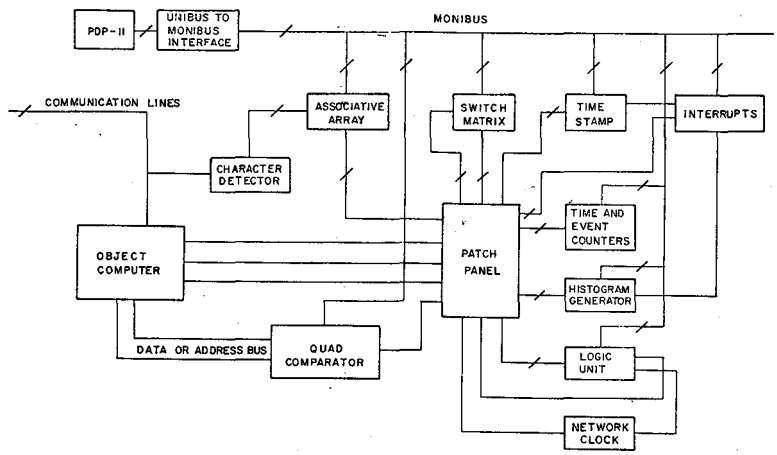


Fig. 3. Generalized monitor.

periodically during some experiments, and to terminate the monitoring session. Each remotely controlled monitor has a microprocessor to handle real-time details, and a minidisk to hold accumulated data for the NMC.

If the NMC cannot control all the RCHM's in the network, regional network monitoring centers (RNMC's) are introduced to form the hierarchy illustrated in Fig. 2. Note that RCHM's can be controlled indirectly (i.e., via telecommunications links) or directly by either a RNMC ora NMC.

As Fig. 3 illustrates, the RCHM is composed of a number of specialized modules interconnected by a bus, called the MONIBUS. The modules included in a monitor depend on the activities to be monitored. Each module is assigned a set of MONIBUS addresses which are used by the controlling computer to send control information and to receive monitored results. The modules included so far are listed here and described in Section III-B.

1) Event detectors:

a) Masked-word range comparators, used to detect an event defined in terms of data or address ranges;
b) combinational logic units, used to detect an event defined ıı terms of Boolean functions of other events;
c) sequential logic units, used to detect an event defined as a sequence of other events; and
d) character detectors for bit-serial lines.

2) Time measuring modules:

a) Time stamp units, used to record time of occurrence, identity, and other selected attributes of an event;
b) event timers;
c) interval timers, for sampled measurements; and
d) network clock, which is synchronized with a standard reference clock.

3) Counters, data reducers, and data recorders:

a) Histogram generators;
b) moment generator, used to yield the first four moments of a given distribution;
c) event counters;
d) flip-flop banks;
e) content-addressable memory (CAM); and
f) random-access memory (RAM).

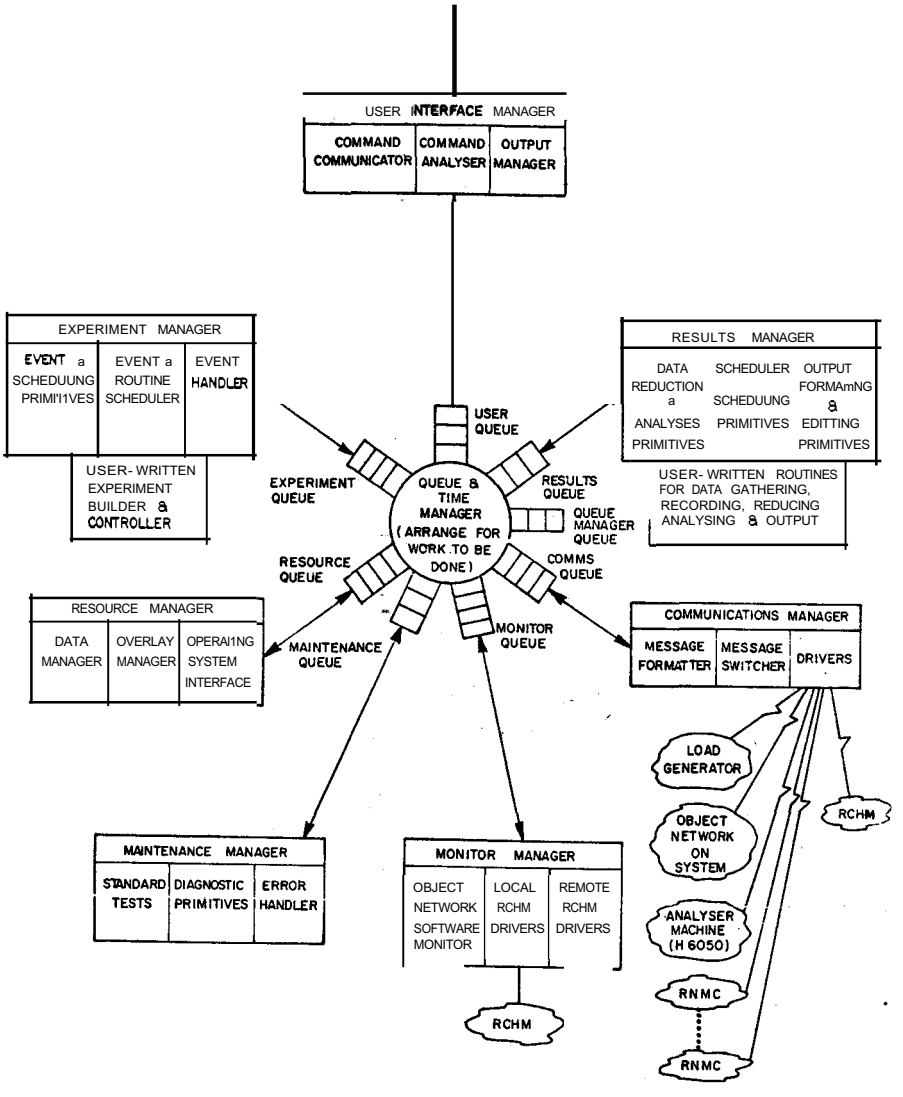


Fig. 4.

4) *Communications and control equipment:*

a) Programmable switch matrices;
b) interrupt generators;
c) RCHM c.ontroller and communications link interface; and
d) MONIBUS-to-UNIBUS (PDP-11) interface.

5) *Signal conditioning circuitry and patch panels*

A set of high-impedance probes connect points of interest in the object computer to the monitor. The probes terminate on a patch panel containing signal conditioning circuitry.

The highly modular monitor architecture makes it quite easy to add new special-purpose data gathering or data reducing components as needed. This modular hardware architecture and the desire to allow the monitoring system to evolve dictate a similar architecture for the software. Figs. 4 and 5 depict the architecture of the software at this stage of its evolution.

The heart of this software is a small, real-time, message-switched operating system, containing a set of RCHM module drivers, interrupt handling routines, primitives to aid authors of experiment control and data analysis routines in writing and scheduling execution of their routines, a small supervisor to allocate resources (processor, memory, and communications) plus the communications control routines. A liInited set of standard routines for data analysis and output formatting, an embryonic version of a translator for a monitoring language,and software to allow the user to interact with the monitoring system, complete the present version of the system software. These software components are described in Section III-B.

As experience is gained using the system, a monitoring language is being defined. The current version of the language is an extension of Fortran; however, the possibility of basing the language on a BCPL-like language (e.g., B [20J) is being actively pursued.

A load generator has been implemented to provide a user with the ability to specify what traffic should be in the network while monitoring, should known traffic be desired [21J. The current version simulates the typing

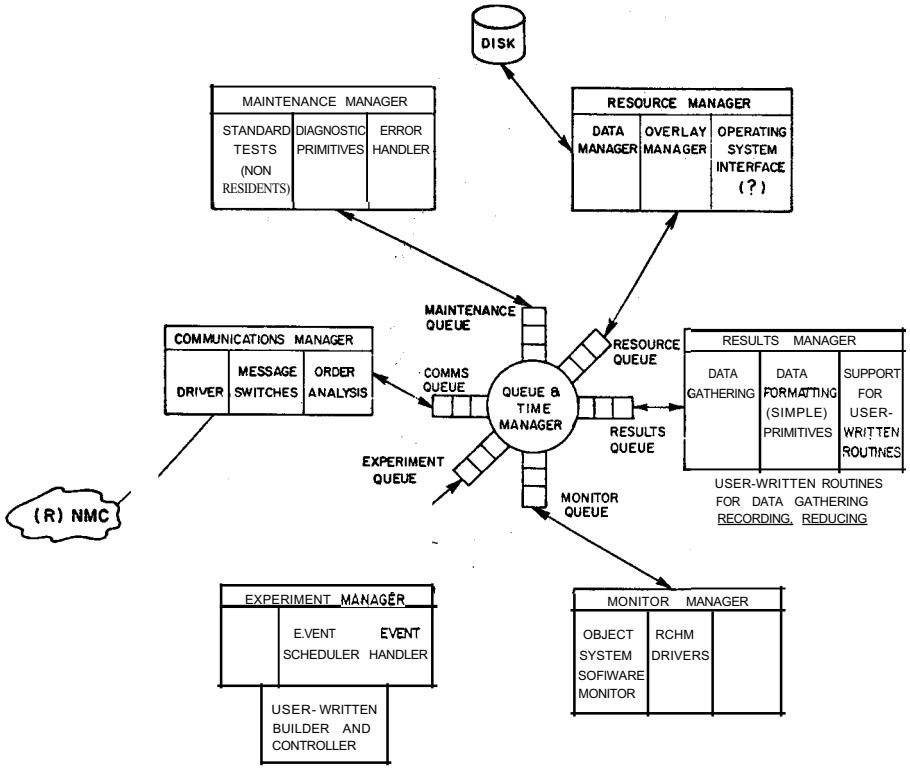


Fig. 5. RCHM controller.

action of up to sixteen users at terminals with speeds up to 300 baud. The load generator transmits prepared scripts from disk to the appropriate line(s), and can simulate thinking and typing time distributions. A version of the load generator to produce' higher speed traffic, simulating host activities, is being created. Eventually, the load generators will also be controllable from the NMC using the monitoring language.

Monitoring software in each node (or host) is obviously quite system dependent; however, sets of standard monitoring software primitives are being designed to observe parameters characteristic of many systems. We are striving to minimize the amount of such software required, as well as the amount of work required to write, install, and debug it. Standard means of communicatilig between this software and the RCHM are being designed.

B. Component Descriptions

1) RCHM Hardware Components: Corresponding to the three main types of computer system events, there are three types of event detectors in·the RCHM:

"value" events ↔ masked-work range comparator
sequential events ↔ sequential logic unit
combinational events ↔ combinational logic unit.

The masked-word range comparator determines whether a bit string, logically "ANDed" with the specified mask, then regarded as a binary value, falls within two program-specified limits. There are five output lines to indicate whether the given string is above, below, or within the range, or whether it is equal to the upper or lower limit. Currently each comparator tests strings of at most sixteen bits, but the comparators have been designed so that four can be easily combined to test a 64-bit string. Four comparators are usually interconnected in a different way to form what we call a quadri-comparator which has four ranges and 16 (of the 17 possible) outputs. The outputs are: below the lowest value, equal to one of the range boundaries, within a range, between two ranges, or greater than or equal to the highest value.

The sequential logic unit determines whether a sequence of events represented on its input lines is following a specified pattern. The pattern is defined by a regular expression which was specified by the experimenter, manipulated by the controlling computer, and stored in the sequential logic unit. The current design features eight inputs, eight outputs, and a maximum of 32 states; however, the unit could, like the other types of units, be of arbitrary size, determined by cost and need. Races are avoided by buffering the inputs and by using a synchronous design.

The combinational logic unit determines whether the eight events represented on its input lines satisfy the Boolean expression specified by the experimenter's program. The functional representation is translated to a Karnaugh map and stored as such in the unit.

The character detector receives characters serially by bit from the telecommunications link to which it is attached, thereby detecting telecommunications link events such as the start of a message. The detector tests to see if each character (5,7, or 8 bit codes) matches one of the sixteen patterns specified by the controlling program. If so, the output line corresponding to the pattern matched is set to "high." This unit employs a simple associative

memory to achieve the desired speed. The character detector and the sequential logic unit are used together to handle communications protocols, e.g., to detect the header of a message or packet.

In a monitoring system there is a need to accurately determine real time, to measure durations, and to obtain pulses of specified widths at specified regular intervals. Facilities for each are provided in the RCHM.

The network clock serves as the source of accurate real time as well as very accurately spaced pulses. The current clock supplies pulses every 100 ns and selected multiples.

When a specified event occurs, the time stamp unit records an identifier for the event, time of occurenceof. the event, and sixteen indicators of the state of the object system when the event occurred. Currently, the output for an event contains 48 bits of information, the minimum time between recordable events is 200 ns, and the clock resolution is 100 ns.

The timer and event counter counts the number of occurrences of the specified event, and measures the total amount of time the event occurred during the period of observation. Currently, each register has 32 bits, the timing resolution is 100 ns, and the maximum count rate is 10 MHz. The controlling program selects one of four clock rates. Under program control, each timer may be used as either a timer or an event counter. One can arrange for the controlling computer to be notified when a counter overflows by connecting the counter's overflow output to the input of an interrupt generator (see below).

The interval timer produces a "high" output every $X + Y$ μs, and the output lasts for Y μs. X and Yare set under program control. The interval timer is used to indicate when to sample the system, should sampling rather than event-driven monitoring be desirable for a particular set of experiments.

In order to keep track of the fact that a set of events of interest has occurred or to aid in measuring the time between events, a set of flip-flops is provided on the patch panels together with numerous standard logic gates.

To record monitored data until the controlling computer has an opportunity to access it, event memory is provided in the form of some semiconductor memory and a minidisk.

In most measurement experiments, the object is to obtain the distribution function for the quantity being measured. In order to facilitate obtaining this distribution function, histogram generators and a moment generator are included in the design of the RCHM. A histogram generator consists of a mask, a bank of comparators (or a CAM), and a corresponding bank of counters. When the input data falls within a range, the corresponding counter is incremented by one. The mask and ranges are set under program control.

The moment generator, working with the output of the histogram generator, produces the first four moments of the distribution.

The interrupt generator signals the nearest controlling computer (whether RCHM controller or NMC or RNMC) whenever one of its input lines indicates that an event has occurred which requires computer intervention. Such events include overflow of counters or timers, data overrun in the time stamp unit, or events selected by the experimenter.

The program-controlled switch matrices connect a software specified input to one or more specified outputs: Using the switch matrices, several experiments can be set up by making the necessary patch panel connections in advance; then, under program control, setting up the switch matrices to perform one experiment at a time. To change from one experiment to another, one merely calls a routine to disconnect certain links, then calls another routine to make the required connections through the switch matrices. Using these matrices, probes are linked to event detecting units, which are connected to data gathering units, and these are connected to data recording and data reducing units. Two sizes of switch matrices have been used thus far: 8 X 8 and 16 X 4.

These programmable switch matrices make the CNMS feasible. Originally, it was hoped to use switch matrices exclusively, eliminating patch panels, but the size, speed and cost of the switch matrices required dictated the compromise of using patch panels plus switch matrices to make the necessary connections. When compared with the exclusive use of patch panels to achieve interconnection, the compromise reduces the time required to change from one experiment to the next and reduces the amount of human intervention required, but not to the level we had hoped. A less expensive switch matrix on a medium scale integrated (MSI) chip is planned to further reduce the use of a patch panel.

2) NMC, RNMC, and RCHM Software: As Figs. 4 and 5 illustrate, the system software for the Inicroprocessor that controls the RCHM and the system software for the NMC and RNMC have basically the same structure. However, the RCHM software is simpler and much smaller than that for the (R)NMC. The principal components of the software correspond with the types of functions to be performed. The message-switched operating system structure is well suited to the problem of controlling parallel tasks in multiple machines.

The user interface manager receives, analyzes, and interprets commands typed by the user as the user sets up an experiment, interacts with it, and obtains and analyzes the results. The command interpreter calls upon the other components of the system to serve the user.

The experiment manager schedules and supports the execution of experiment control programs written by users in the monitoring language.

The monitor manager drives (or arranges for the RCHM controller to drive) the components of the RCHM to perform the monitoring activities requested by the user.

The resource manager allocates main and auxiliary memory, and records and retrieves monitored data, experiment control programs and system programs.

The communications manager in the NMC handles communications with the RNMC's it controls, with the load generator, with the object network monitoring software, with the data analysis system, and with any

RCHM's it controls directly. The communications manager in the RNMC provides communications with its controlling NMC and possibly the object network monitoring software. The communications manager in the RCHM controller is only concerned with its controlling NMC orRNMC.

The results manager schedules and supports user-written or system-supplied routines to record, reduce, and analyze monitored data. Experiments requiring a great deal of data analysis send the data to a larger system that is more suitable for such analysis.[1] At the University of Waterloo, the Honeywell 6050 is used for this purpose.' When the analysis is complete, the results are formatted by routines of the results manager selected by the user.

The maintenance manager provides diagnostic routines and standard test packages for hardware and software of the CNMS. Using these routines, a knowledgeable user can interact directly with the components of the RCHM's and can perform reasonably complex experiments without having to write and compile experiment control programs. The maintenance manager also contains all routines necessary to handle CNMS hardware or software errors.

The queue manager provides the primary means of communicating between these software components. A service to be performed is requested by building a queue entry and asking the queue manager to put it in the appropriate place in the proper queue. A service which is to be done at a particular time is placed in the queue manager's queue until it is time to move it to the appropriate action queue.

A good first generation of most of these software components is being used to perform moderately complex experiments. A more sophisticated and complete version of the software is now being written.

IV. USING THE CNMS

A. General Procedure

In order to monitor a network using the CNMS, several functions must be performed.

Step 1: Determine what is to be monitored and how to monitor it.

Step 2: Determine what hardware probes (if any) are to be used, install them, and test them using an oscilloscope and the diagnostic software of the CNMS.

Step 3: Determine what software monitoring tools (if any) are to be used in the object network, install them, and test them using the diagnostic software of the CNMS.

Step 4: Decide whether known, controlled traffic is desired for the experiment and, if so, provide the load generator with the necessary scripts, distributions and line descriptors.

Step 5: Define the software necessary to define and control the experiment and analyze the resulting data.

Step 6: Set up the patch panel as required and debug the resulting combination of hardware and software using the diagnostic software of the CNMS.

Step 7: Using the command language, initiate the experiment from a terminal attached to the NMC.

Step 8: Interact with the experiment.

Step 9: Obtain and interpret the results.

Today's computer architecture unfortunately demands that Step 2 be performed by a computer hardware expert. Step 3 must be performed by a computer software expert, but we are trying to develop software monitoring primitives that will simplify this step. Step 4 requires knowledge of the use of a load generator and a text editing system. The text editor is needed to create the scripts of transactions required by the load generator. To do Step 5 requires knowing how to write experiment control programs using the monitoring language and how to use the support routines provided to help control experiments and analyze results. Step 6 hopefully only requires knowledge of CNMS and the system being monitored, but could require expert help if problems exist with the hardware or with the object network software. Steps 7-9 only require knowledge of CNMS and the characteristics of the object network.

Thus, monitoring a computer network is still a rather demanding task. However, once Steps 1-6 have been performed for a set of experiments, the remaining steps can be performed without detailed knowledge of how the monitoring is being accomplished.

B. An Example

The following example was chosen from a set of experiments that have been performed to measure, evaluate, and improve the CNMS and its components [22]. The example illustrates use of the RCHM and indicates a useful way of representing data.

As discussed in Section III, an important component of the CNMS is the load generator. In order to study its behavior, we assembled a two-node computer network. Each node executes the load' generator to produce traffic for the other node through a variety of data links. The rate at which the load generator applies transaction traffic to each of its output lines iscontroUed by the user's .choice of distribution function (e.g., exponential, uniform, hyperexponential) and by his choice-of parameters for the distribution. A variety of line speeds can be produced in our networks laboratory for the data links joining the two nodes of this captive network. Thus, we can subject the load generator to a wide variety of tests while observing its behavior.

We have found two histograms to be particularly useful for understanding and modeling the behavior of computer systems or networks: system state versus time in each state, and system state transition versus number of such transitions. Both types of histograms are being produced

[1] Alternately, the NMC could be used at some sacrifice in efficiency and power.

```
*** EXPERIMENT #4 RESULTS ***

   THE FOLLOWING TABLE INDICATES THE TIME SPENT
IN EACH OF THE 8 POSSIBLE STATES INVOLVING
2 CPUS AND A COMMUNICATIONS LINK BETWEEN THEM.


ALL TIMES IN MICROSECONDS
TOTAL REAL TIME:     0.977234670000000D U7
TOTAL COMPUTE TIME:     0.484869800000000D U7
STATE VECTOR IS
   <LINE BUSY,CPU #2 BUSY,CPU #1 BUSY>
   CPU #1 = MATH PDP11/20
   CPU #2 = ENGINEERING PDP11/20

   STATE    STATE    TIME                    TIME/              TIME/
            VECTOR   IN STATE                REAL TIME          COMPUTE TIME

   0        000      0.4576359500D 07        0.4683D 02         0.S4380 02 %
   1        001      0.7356001000D 06        0.7527D 01         0.1517002 %
   2        U10      0.1052109300D 07        0.10770 02         0.21700 02 %
   3        011      0.2303453600D 07        0.2357D 02         0.4751D 02 %
   4        100      0.3160886 0000 06       0.3255D 01         0.65600 01 %
   5        101      0.58153200000 05        0.5951D 00         0.11990 01 %
   6        110      0.3175255000D 06        0.3249D 01         0.6549D 01 %
   7        111      0.3822386000D 06        0.3911D 01         O.7883D 01 %

                    TOTAL TIME              TIME/              TIME/
                                            REAL TIME          COMPUTE TIME
CPU #1 BUSY:        0.3479451500D 07        0.3561D 02         0.7176002 %
CPU #2 BUSY:        0.4055327000D 07        0.41500 02         0.8364D 02 %
LINE BUSY:          0.1076011900D 07        0.11010 02         0.221S0 02 %
```

Fig. 6. Measurement of file transfer from machine 1 to machine 2.

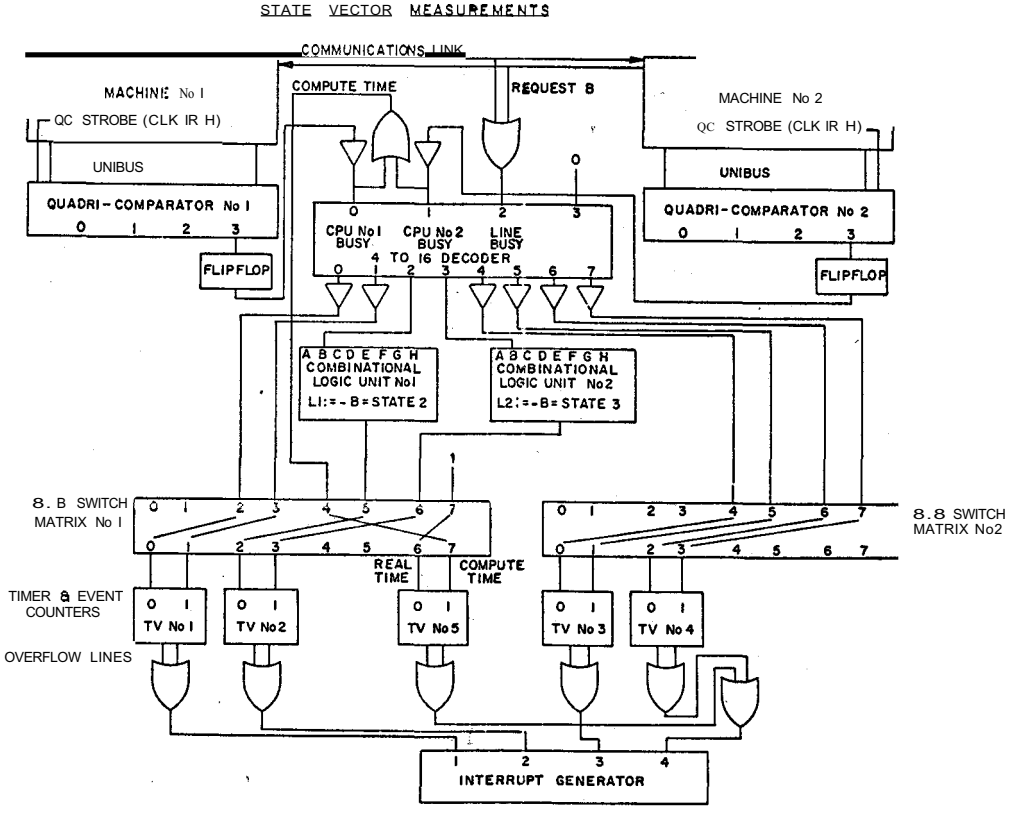


Fig. 7. State vector measurements.

as part of the measurement and evaluation of this network of load generators, but only the first will be presented here. (A paper describing the measurement and evaluation of the load generator network is being prepared.)

The histogram shown in Fig. 6 was produced by connecting the components of the RCHM to the load generator network as shown in Fig. 7, and by writing and executing the experiment control program shown in Fig. 8. The subroutines called by the experiment control program are primarily RCHM drivers. The variable "RCHM" indicates which RCHM is being used. (When this experiment was being performed, only one RCHM was assembled and working, but now there are two.) The variable "UNIT" indicates which of the several com-

```
C
C                    FILE NAME:   OEXP4.EXP
C
C  PURPOSE OF EXPERIMENT #4
C
C
C         -TO DETERMINE THE TIME SPENT IN EACH OF
C          8 POSSIBLE STATES AS A RESULT OF USING THE
C        • STATE VECTOR [LINE BUSY. CPU #2 BUSY, CPU #1 BUSY]
C          WHERE THE LINE IS A COMMUtUCATIONS LINK BETWEEN
C          ]HE TWO CPU'S.
C
C         -TO FILL IN THE FOLLOWING TA8LE:
C          STATE       TIME IN          TIME/            TIME/
C                        STATE         REAL TIME       COMPUTE TIME
C
C          WHERE REAL TIME= >TOTAL EXPERIMENT TIME. AND
C          COMPUTE TIME => CPU # 1 OR CPU #2 BUSY.
C
C  THE SYSTEM IS DEFINED TO BE BUSY, OR DOING USEFUL
C  COMPUTING. IF ONE OR BOTH OF THE MACHINES IS
C  EXECUTING OUTSIDE OF THE PROGRAM WAIT LOOP.
C
C
C
          SUBROUTINE DEXP4-

          EXTERNAL TVOVFL
          IMPLICIT INTEGER(A-Z)
          INTEGER TVOVFO(5), TVOVFL(5)
          COMMON /CONFIG/NODE, UNIT
          COMMON /TVCOM/TVOVFO(5). TVOVFL(5)
C
C  OUADRA-COMPARATORS MUST BE SET UP, USING
C  THE TEST PROr.RAM.   FOR COMPUTE TIME, SET RANGE #3 TO
C  THE ADDRESSES OF THE WAIT LOOP.
C
          WRITE(6.12)
12        FORMAT(' USE TEST PROGRAM TO SPECIFY QC RANGE #3 = WAIT LOOP',/ '
          1 '   ON BOTH MACHINES' ./' ')
C
C  COMPUTE TIME => CPU'I BUSY   OR   CPU #2 BUSY
C         => (OUTSIDE CPO #1 WAIT LOOP)   OR
C           (OUTSIDE CPU '2 WAIT LOOP)
C
C
C
C
C SET UP LOGIC UNITS
C
C  LOGIC UNIT #1 = -B = STATE 2 • 1SW815
C
          L1 :=-B

     LOGIC UNIT #2 = -B = STATE 3 = 1SW816

          L2: =-B
C
C  SET UP 8X8 SWITCH MATRICES
C
C  5TATES
               50=2
               51=3
               52=5
               53=6
               54=4
               55=5
               56=6
               57=7

     TIMER & EVENT COUNTERS

               TVIBO=O
               TV1BI=1
               TV2BO=2
               TV2BI=3
               TV3BO=0
               TV3B1=1
               TV4BO=2
               TV4B1=3
               TV5BO=6
               TV5B1 =7
               RTIME=7
               CTIME=4

               UNIT'1
               CALL 5W8DIS
               CAll SW8CON(SO, TV1BO, 5', T'!IBI, S2, TV2BO, S3, TV2B1,
               1 RTIME. TV5BO. CTIME, TV5BI)

               UNIT=2
               CALL 5WBDI5
               CALL SW8CON(S4, TV3BO, S5, RV3BI, S6, TV4BO, S7. TV4BI)

     5ET UP TIMER & EVENT COUNTERS

               DO 35 UNIT=I, 5
35             CALL TV5ET(I.1.1,3.1,1)
C
C
C  5ET UP INTERRUPT GENERATOR AND CLEAR OVERFLOW COUNT5.
C
               DO 40 UNIT=I,4
               TVOVFO (UNIT) =0
               TVOVFL(UNIT)=O
40             CALL IGSET(TVOVFL, UNIT)
C
C  TV #5 OVERFLOW LINES ARE 'OR'ED WITH THOSE OF TV #4, SINCE
C  ONLY HAVE FOUR INTERRUPT LINES.   A SPECIAL CHECK IS MADE IN
C  'TVOVFL'.
               TVOVFO (5) =0
               TVOVFL(5)=0

               RETURN
               END
```

Fig. 8.

ponents of the same type In an RCHM is being addressed.

V. APPLICATIONS OF THE CNMS

Besides its intended use of monitoring a computer network, the system has been used successfully to monitor the activities of a computer system. Several experiments have been and are being performed, including the following.

1) Using the monitor to locate frequently used code and system bottlenecks in DOS-11 and Fortran as programs are compiled and executed;
2) determining the loading on the PDP-11 UNIBUS;
3) determining the frequency of execution of each of eight classes of instructions for different kinds of programs in order to compare our results with those obtained by Klar and Schreiber at the University of Erlangen [23J;
4) validating a mathematical model of two transaction-oriented data base management systems interacting with each other and sharing data; and
5) locating inefficiencies in a computer network simulator implemented to work in parallel on three interconnected PDP-11 computers.

Reports describing the results of these experiments are being prepared. Many other experiments are planned, e.g., measuring the swapping activities of various PDP-11 operating systems, observing Honeywell's GCOS executing on a 6050, watching VM/370, monitoring the performance of an experimental loop network joining our laboratory with laboratories in Toronto and Ottawa, as well as monitoring our campus network.

Eventually we think that some form of a CNMS will be a vital component in an automated maintenance system for computer networks, helping to detect and diagnose malfunctions and bottlenecks in hardware and software semiautomatically. Such a system is being designed, and implementation of a prototype is planned during 1975-1976. Furthermore, we anticipate that a form of CNMS will eventually be an important component in an adaptively controlled computer network. We are working toward these goals as well as toward "simply" monitoring the performance of computer networks and systems.

Components of the CNMS are being employed in a novel experiment to test the feasibility of providing hardware assistance to an information retrieval system. The results of this experiment should be available in about 18 months.

TABtE I
EVALUATION OF CNMS (SEPTEMBER 1974)

Weight	Criterion	Raw Score Now	Raw Score 1976	Weighted Score Now	Weighted Score 1976	Comments
	Ease of Use					
	Software					
.09	To write exp control programs	+ 1	+ 3	.09	.27	
.07	To use system software	+ 1	+ 3	.07	.21	
	Hardware					
.08	To set up experiments	- 2	- 1	-.16	-.08	
.03	To write software for	+ 3	+ 3	.09	.09	
.03	To use, once set up	+ 3	+ 3	.09	.09	
	Flexibility and expandability					
.06	Of software	+ 3	+ 3	.18	.18	
.06	Of hardware	+ 2	+ 2	.12	.12	
	Object system independence					
.02	Of software	+ 1	+ 2	.02	.04	
.02	Of hardware	- 1	+ 3	-.02	.06	
.04	Int-erference with object system	+ 3	+ 3	.12	.12	
	Dependability					
.02	Of software	+ 3	+ 3	.06	.06	
.03	Of hardware	+ 1	+ 2	.03	.06	
	Diagnosability					
.02	Of software	0	+ 1	0	.02	
.03	Of hardware	+ 2	+ 2	.06	.06	
.04	Interference with Corr.municationsl	+ 3	+ 3	.12	.12	
.02	Object system security and integrity	+ 1	+ 1	.02	.02	
.01	Choice of resolution	+ 1	+ 1	.01	.01	
.1	Ability to remotely monitor	0	+ 3	O·	.3	
.03	Ability to span the network	0	+ 2	0	.06	
.2	Cost	- 3	- 2	-.6	-.4	
1.0	TOTAL			0.3	1.41	

Finally, it appears that the CNMS, with minor software and hardware modifications, can be used to monitor a set of electronic switching offices for telephones [24J.

VI. SUMMARY, CONCLUSIONS, AND FURTHER WORK

In Section II it was concluded that an integrated monitoring system is preferable to a set of unrelated, uncoordinated monitoring tools, because few people who need to monitor a system or network are hardware, software, and statistics experts whose primary interest in life is to monitor the system or network. In Section III the CNMS being created at the University of Waterloo was described. Section IV presented a simple example to illustrate use of the system, and Section V mentioned several experiments that are being performed using the CNMS.

S. Mamrak, in her forthcoming article entitled "Performance Evaluation in Computer Networks: A Survey" [1J, states: "Actual system measurements, analyzed using statistical techniques and used to improve queueing and simulation models, have been relatively neglected. (This neglect may be due in part to the unavailability of tools for making desired observations of dynamic systems and of statistically significant test environments.)" It is hoped that the CNMS described in this paper will be a good first step toward satisfying this need.

As promised in Section I, we have evaluated our CNMS based on our experience in using it. Table I is our evaluation of the system as it stands at this writing and our prediction of an evaluation as the system should be at the beginning of 1976. The evaluation is based on the nine characteristics of an ideal CNMS listed in Section II. The scores range from -5 to +5, with -5 meaning "Terrible, couldn't be worse," +5 meaning "Excellent, couldn't be better," and 0 being the borderline between being acceptable and unacceptable.

As the scoring indicates, the main problems with the CNMS are cost and the continuing need for a patch panel. We anticipate that both problems can be solved in time,

especially considering the rate at which the cost of logic is dropping and recent developments in solid state switching for data communications.

. The prototype RCHM's use TTL logic, which limits our resolution to 10 MHz; however, some of the newer components .contain Schottky logic in order to monitor the microprogrammed PDP-U/45.

A few conclusions can be drawn from our experience thus far.

1) The hardware monitor (RCHM) works and is not prohibitively expensive to build (i.e., $10,000 to $100,000, depending on which modules are included and in what quantity).

2) The modular components plus the bus architecture make it easy to insert or remove components as desired. Above the cost of a basic monitor, the cost of the monitor increases as the complexity of the experiments to be performed increases.

3) It is not difficult to write control programs for the monitor, even without the monitoring language, because each component is addressed as a set of memory locations on the controlling PDP-II. The primitives of the current, embryonic monitoring language are simply Fortran subroutine calls. The routines themselves are written in either Fortran or Assembler for the PDP-II.

4) The diagnostic hardware and software that we included in the CNMS continually proves its value. Many monitoring tools do not include such error detection and diagnostic tools. We have found that it is definitely worth our time to quickly run through a set of routine hardware and software test programs before running an experiment.

5) A system hardware expert would not be required to install the probes if computer manufacturers provided an accessible panel of probe points on their products.

The security and privacy questions that arise from considering the widespread use of CNMS's are thought provoking, to say the least, and could be the subject of a long discourse. A simple way to prevent unauthorized snooping is to keep the phone numbers of the RCHM's of the CNMS a well-guarded secret.

Creating a network monitoring system is an ambitious project. Although much has been accomplished since the project began in mid 1971, a great deal of work remains to be done, some of which is listed here.

1) Develop a theory of computer-monitoring. Possible topics include extending the work of Morgan and Sutton [35J in formally defining events in terms of system states, providing a formal basis for deciding when the performance of a system is acceptable, and creating a theoretical basis on which to build monitoring systems.

2) Find solutions to the problems of determining exactly when an event occurred and deciding the order in which two nearly simultaneous events occurred in separate computers of the network.

3) Develop an easily used, extensible language for defining and controlling monitoring experiments as well as analyzing the data. Our Fortran-based language is only a poor first step toward this goal.

4) Determine what parameters characterize the performance and workload of a computer system or network. Some form of so called Kiviat graph might be useful to represent the performance of the network in terms of these parameters [36].

5) Create a self-monitoring computer system, and then a self-monitoring computer network. A self-monitoring computer system is one that includes special hardware (e.g., microprogrammed special instructions) to aid the system in observing its own activities. Similarly, a self-monitoring computer network would contain special hardware and firmware to help the network observe its own activities.

6) Create an adaptive computer system that monitors its workload and its performance while continually adjusting its resource multiplexing parameters accordingly. A mathematical model of such a system has been analyzed by Badel *et al. [25].*

And much, much more.

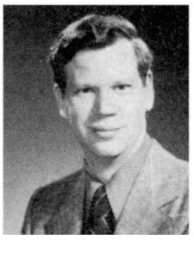
ACKNOWLEDGMENT

The hardware skills of J. Runge and K. Jedermann, and the software talents of F. Mellor, K. Pammett, J. Palpraman, D. Sutton, W. Colvin, and R. Shen have contributed immensely to the project.

REFERENCES

[1] S. Mamrak, "Performance evaluation in computer networks," *Comput. Surveys,* to be published.
[2] D. Morgan, W. Banks, W. Colvin, and D. Sutton, "A performance measurement system for computer networks," in *Proc. 1974 Int. Fed. Inform. Processing Congr.,* pp. 29-33.
[3] G. D. Cole, "Computer network measurements-techniques and experiments,"Univ. California, Los Angeles, NTIS, Rep. AD-739-344, Oct. 1971.
[4] C. T. Apple, "The program monitor-a device for program performance measurement," in *Proc.* Ass. *Comput. Mach. 20th Nat. Conf.,* Aug. 1965, pp. 66-75.
[5] R. A. Aschenbrenner *et al.,* "Neurotron monitor system," in *1971 FaU Joint Comput. Con/., AFIPS Con/. Proc.,* vol. 39. Montvale, N. J.: AFIPS Press, 1971, pp. 31-37.
[6] A. J. Bonner, "Using system monitor output to improve performance," *IBM Syst. J.,* vol. 8, no. 4, pp. 290-298, 1969.
[7] D. T. Bordsen, "Univac 1108 hardware instrumentation system," in Ass. *Comput. Mach. SIGOPS Workshop System Performance Evaluation,* Harvard Univ., Cambridge, Mass., Apr. 1971, pp. 1-29.
[8] G. Estrin *et al.,* "SNUPER computer," in 1967 *Spring Joint Comput. Conf., AFIPS Con/. Proc.,* vol. 30. Washington, D. C.: Thompson, 1967, pp. 645--656.
[9] L. E. Hardt and G. J. Kipovitch, "Choosing a system stethoscope," *Comput. Decisions,* pp. 20-23, Nov. 1971.
[lOj T. Y. Johnston, "Hardware vs. software monitors," in *Proc. SHARE, XXXIV,* vol. 1, Mar. 1970, pp. 523-547.
[11] K. W. Kolence, "Software physics and computer performance measurements," in *Proc.* 1972 *Ass. Comput. Mach. Annu. Conf.,* pp. 1024-1040.
[12] H. Lucas, "Performance evaluation and monitoring," *Comput. Surveys,* vol. 3, pp. 79-91, Sept. 1971.
[13] R. W. Murphy, "The system logic and usage recorder, in 1969 *FaU Joint Comput. Conf., AFIPS Conf. Proc.,* vol. 35. Montvale, N. J.: AFIPS Press, 1969, pp. 219-229.
[14] C. D.Warner, "Hardware techniques," *Ass. Comput. Mach. SIGCOSM Newsletter,* no. 5, pp. 5-11, Aug. 1970.
[15] M. D. Abrams, "Consumer-oriented measurements of computer network performance," in *Proc. Nat. Telecommunications Conf.,* IEEE Communications Soc., San Diego, Calif., Dec. 1974.
[16] W. Banks and D. Morgan, "A computer controlled hardware monitor: hardware aspects," in *Proc. Int. Meeting Minicomputers and Data Communications,* Liege, Belgium, Jan. 1975.
[17] J. Hughes and D. Cronshaw, "On using a hardware monitor as an intelligent peripheral," Xerox Corp., EI Segundo, Calif., Oct. 1973.

[18] Y. Kalef and M. Melman, "Performance analysis of the Golem B computer system," Dep. Applied Math., The Weizmann Inst. Sci. Rehovot, Israel, Tech. Rep.
[19] P. R. Sebastian, "Hybrid events monitoring instrument," in *Proc. 1974 SIGMETRICS,* Oct. 1974.
[20] B. W. Kernighan, "A tutorial introduction to the language B," Bell Laboratories, Murray Hill, N. J., Comput. Sci. Tech. Rep., Jan. 1972.
[21] F. Mellor, "A general purpose load generator," Univ. Waterloo, Waterroo, Ont., Canada, Apr. 1974.
[22] D. Morgan, D. Goodspeed, and R. Kolanko, "Demonstration of a programmable hardware monitor," Univ. Waterloo, Waterloo, Ont., Canada, Comput. Commun. Network Group External Rep., Oct. 1974.
[23] R. Klar and H. Schreiber, Inst. Computer Sci., Univ. Erlangen, Erlangen, Germany, unpublished Rep.
[24] R. E. Machol, "Acquiring data for network planning and control," *Bell Laboratories Rec.,* vol. 52, pp. 279-285, Oct. 1974.
[25] M. Badel, E. Gelenbe, J. Lenfant, J. Leroudier, and D. Potier, "Adaptive optimization of the performance of a virtual memory computer," in *Proc. 1974 SIGMETRICS,* Oct. 1974.
[26] *Comress: Dynaprobe 7900: System Specifications,* Comress Rep. CR4-0031.
[27] J. M. Grochow, "Real time graphic display of time-sharing system operating characteristics," in 1969 *Fall Joint Comput. Conf., AFIPS Conf. Proc.,* vol. 35. Montvale, N. J.: AFIPS Press, 1969, pp. 379–386.
[28] J. H. Salzer and J. W. Gintell, "The instrumentation of multics," in *Second Symp. Operating System Principles,* Oct. 1969, pp. 167-174.
[29] K. W. Kolence, "System improvement by system measurement," *Ass. Comput. Mach. SIGBDP Newsletter,* pp. 6–11, Winter, 1969.
[30] E. F. Miller, "An experiment in hardware monitoring," General Res. Corp., Santa Barbara, Calif., Rep. RM-1517, July 1971.
[31] R. G. Canning, "Savings from performance monitoring," *EDP Analyzer,* vol. 10, pp. 1-15, Sept. 1972.
[32] R. L. Patrick, "Measuring performance," *Datamation,* vol. 10, pp. 24-27, July 1964.
[33] S. H. Fuller, R. J. Swan, and W. A. Wulf, "The instrumentation of C.mmp, a multi-(mini) processor," in *Proc. IEEE Int. Cornput. Soc. Conf.,* Feb. 1973, pp. 173-176.
[34] E. L. Burke, "A computer architecture for system performance monitoring," in *First Annu. SIGME Symp. Measurement and Evaluation,* Feb. 1973, pp. 161-169.
[35] D. A. Sutton and D. E. Morgan, "The monitoring of computer systems and networks: A summary and a proposal," Univ. Waterloo, Waterloo, Ont., Canada, Comput. Commun. Network Group External Rep., Apr. 1974.
[36] J. D. Noe and N. W. Runstein, "Develop your computer performance pattern," in *Proc. 1974 SIGMETRICS,* Oct. 1974.



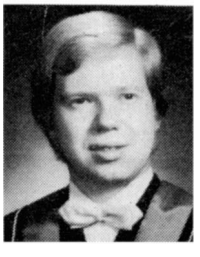
David E. Morgan (M'74) was born in Terre Haute, Ind., on March 22, 1942. He received the B.Sc. degree in mathematics from Rose Polytechnic Institute, Terre Haute, Ind., the M.Sc. degree in mathematics from the University of Michigan, Ann Arbor, and the Ph.D. degree from the University of Waterloo, Waterloo, Ont., Canada, in 1964, 1965, and 1971, respectively. He earned the Graduate Study Program Certificate of Bell Telephone Laboratories in 1966.

As a member of the Technical Staff of Bell Laboratories from 1964 to 1970, he helped develop IBM's time sharing system for the 360/67, studied possible landing sites on the Moon for the Apollo astronauts, and performed other systems programming and systems engineering functions. He has served as consultant for several governmental and industrial organizations during the past five years, and is President of Mordata Ltd., a small computer consulting firm. He is presently Associate Director of the Computer Communications Networks Group and Assistant Professor of Computer Science at the University of Waterloo. He is the author of several papers relating to computer networks, performance measurement, and operating system reliability.

Dr. Morgan is a member of the Association for Computing Machinery and the Canadian Information Processing Society.

Walter Banks graduated from the Eastern Ontario Institute of Technology, Ottawa, Ont., Canada, in 1969.

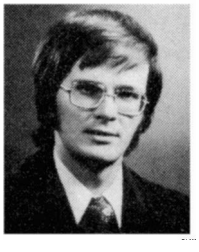
From 1969 to 1970 he was employed at Canadian Westinghouse, where he was engaged in the devel(jpment of real time process control systems and in the development of special purpose and peripherals for their Prodac line of process control computers. Since 1970 he has been employed by the University of Waterloo, Waterloo, Ont., Canada, where he is currently the Laboratory Director of the Computer Communications Networks Group. His interests are in special purpose processors and microcontrollers.



Dale P. Goodspeed (S'74) was born in Sudbury, Ont., Canada, on September 26, 1950. He received the B.Math. and M.Math. degrees from the University of Waterloo, Waterloo, Ont., Canada, in 1973 and 1974, respectively.

He is currently a Ph.D. candidate in computer science at the University of Waterloo, and holds a National Research Council Scholarship. His interests include performance measurement, computer networks, and minicomputer systems.

Mr. Goodspeed is a student member of the Association for Computing Machinery.



Richard Kolanko (S'73) was born in Sydney, N.S., Canada, on April 26, 1948. He received the B.Sc. degree from St. Francis Xavier University, N.S., Canada, and the M. Math. degree from the University of Waterloo, Waterloo, Ont., Canada, in 1970 and 1972, respectively.

He is currently completing the Ph.D. requirements at the University of Waterloo.

Mr. Kolanko is a member of the Association for Computing Machinery.