SLAC-PUB-5182
January 29, 1990
M

Network Management for the Micom Digital PABX and the 3Com/Bridge Ethernet Terminal Servers at SLAC*

T. C. STREATER

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94309

(Submit *ted for Publications)*

*Work supported by the Department of Energy, contract DE-AC03-76SF00515.

ABSTRACT

The Stanford Linear Accelerator Center (SLAC) is located on some 480 acres of the Stanford University campus, near Palo Alto, California, and is operated by Stanford University for the Department of Energy. It is devoted to experimental and theoretical research in elementary particle physics, and to the development of new techniques in high-energy accelerators and elementary particle detectors. SLAC operates a Computer Center with an IBM† 3090 mainframe; the SAS System‡ is used for statistical analysis of accounting data. Elsewhere on site are some 100 DEC§ VAX and MicroVAX minicomputers. These machines are all on a Local -Area Network (LAN) using Ethernet' technology. Access to these systems by ASCII terminals is provided by Micom* Digital PABXes, 3Com/Bridge Ethernet Terminal Servers, and DEC-LAT servers. This paper describes how some network-management functions for the Micom and 3Com/Bridge equipment are provided at SLAC. These functions are implemented as a distributed software package running under VMS on a DEC MicroVAX II, with a user interface available under VM on the IBM 3090 and under VMS on the VAXes. This report describes the motivations behind this management software, discusses their implications for the implementation, and how the resulting package has managed to meet the challenges placed upon it.

† IBM is a trademark of International Business Machines, Inc.
‡ SAS is a registered trademark of SAS Institute Inc.
§ DEC, VAX, MicroVAX, and VMS are trademarks of Digital Equipment Corp.
¶ Xerox, Ethernet, and XNS are trademarks of Xerox Corp.
* Micom and Instanet are trademarks of Micom Communications Corp.

Table of Contents

1. Introduction

1.1. THE ASYNCHRONOUS TERMINAL NETWORK AT SLAC

Connectivity for asynchronous terminals at SLAC is mainly provided in two ways: via a number of interconnected Micom Instanet 6000/6600 Digital PABXes[1] and via a network of 3Com/Bridge Ethernet Terminal Servers[2]. For simplicity, the Micom PABXes will henceforth be referred to as the *Micom* and the 3Com/Bridge Servers as the *Bridge.* The term *Switch* will be used generically to refer to the -Bridge as a unit or to any of the Micom PABXes. Figure 1 shows how the main elements are connected together.

1.2. MANAGEMENT COMMUNICATION WITH A SWITCH

The Statistics Port The Micom offers the user a simple means of choosing a desired host computer or other service. From an initial disconnected state, the user wakes up the Micom by entering a carriage-return or a space character on his terminal. The Micom then prompts the user to enter the symbolic name of the desired service class and connects him to it if an idle port is available. When such a connection occurs, a *statistics* record is generated within the Micom. This record is output to the *statistics* **port** on the Micom, one of two ports** provided for monitoring and control functions on the Micom. When the session terminates, and also under various other conditions, other types of statistics records are generated.

The Command Port The second port, known as the *command* or *administration* port, permits the *configuration* of the Micom (to define which ports are connected to which service, for example) and its *interrogation* (to ascertain the state of ports, for example). These functions are performed using the predefined *command set,* a set of allowed commands implemented by the firmware within the Switch. The

** Some models of Micom PABX have more than two such ports. However, for the purposes of this discussion it is sufficient just to consider two.

term *command port* will be used to refer to this port. The Bridge also has two ports which provide essentially the same functions.

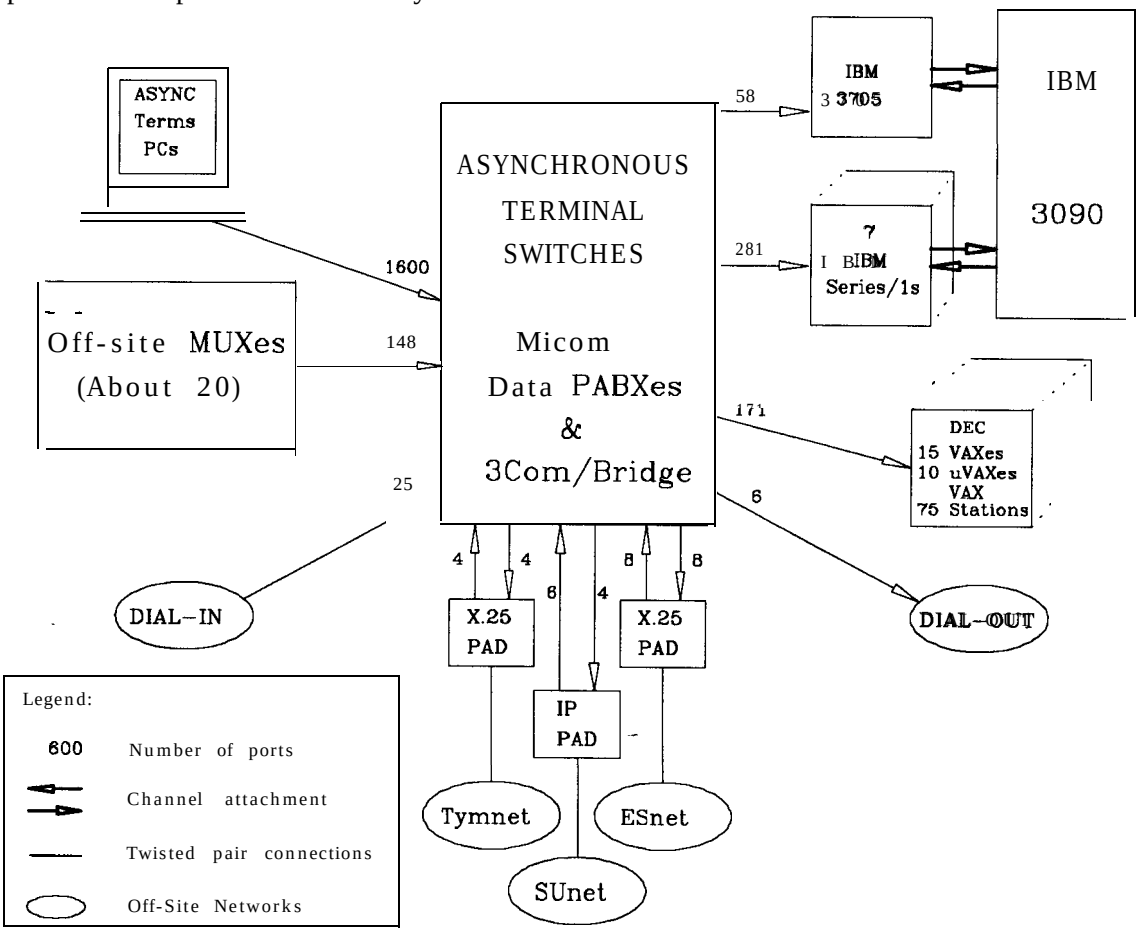

Figure 1: SLAC Asynchronous Connections

1.3. USE OF THE COMMAND AND STATISTICS PORTS

In most installations, these ports are treated in a fairly simple way. The statistics port might be left unconnected, or it might perhaps be connected to a printer, producing a hard copy of the statistics log. The command port is usually looped back to the Switch as a service class, thus allowing access to it from any terminal. Some sites just attach a terminal to the command port and leave it at that.

At SLAC, much more than that is done; at this point it is useful to examine the reasons behind doing more.

2. Motivations

The SLAC asynchronous environment is characterised by these features:

1. **Continuous Growth.** Over the past seven years, the number of interfaces on the Micom has grown from some 500 (on one PABX) to around 2500 (on two PABXes). In addition, the Bridge Terminal Server network has been added (some 200 interfaces). This growth has happened both on the terminal side and the computer port side of the Micom and the Bridge. Not all the interfaces are connected to equipment on-site; about 200 terminals and computer ports are connected via leased lines to over 20 other institutions across the country, while local offsite access is provided via dial-in and dial-out modems, and international access is provided via an X.25 PAD connected to Tymnet.

2. **Continuous Change.** Once interfaces are configured for a given use, that group of ports may grow, shrink, or be moved. User request or proaction on the part of the SLAC's Network Group can cause growth. Movement may occur due to consolidation. Shrinkage would most likely occur due to action on the part of Network Group personnel.

3. **Caution and Dedication to Service.** Although Network Group personnel sometimes take risks, in general there is a desire to provide a reliable and continuous service, with as few interruptions as possible due to outages and to inadequate resources.

   On the Bridge, the configuration information is held on a floppy disk. Thus, if any part of the Bridge needs to be rebooted or have its configuration reloaded, it is reloaded from the floppy. The Micom on the other hand has no floppy; its configuration information is held in CMOS memory with battery backup.

The battery backup works well; however, operator error can cause loss of the configuration information. Under these conditions all configuration information, which may amount to several hundred lines of input, has to be reentered via the command port. Obviously, it is desirable to accomplish this rapidly and reliably to prevent a prolonged outage. As a corollary the configuration must be maintained correctly and automatically, independent of the Micom.

Both the Micom and the Bridge provide queueing. Should no ports be available in a given service class, the user may choose to be queued for that resource and be automatically allocated a port when one becomes free. Thus the person who is queued is free to do other things and is not obliged to keep attempting to connect. Although this saves some of the time wasted by the necessity of queueing, Network Group personnel and users alike regard it merely as a bandaid. Far better to be able to predict when the queuing is likely to start occurring, and use that lead time to plan for the provision of more resources.

4. **Easy Command Port Access.** At SLAC, many people must interrogate and perhaps configure the Micom and the Bridge: technicians, installers, developers, and trouble-shooters. As noted above, at some installations the command port is looped back to the Micom, thus making it available as a service class from any terminal. While this is much more flexible than merely attaching a terminal to the command port, it is much less flexible than the ideal. One would like that the command port not merely be available from any terminal on the network, but also that access to it be available on a guaranteed basis to any number of Network Group personnel, apparently simultaneously.

5. **User Information.** Users, particularly at SLAC, always like to know what is going on. Moreover, when a problem arises, they would like information *quickly* and *automatically* about it and when it might be fixed.

6. **Management Information.** Network Usage Reports inspire user confi-

dence in the system as users feel that management knows what is going on in the network. Additionally, such reports assist management in planning.

The initial specification for the design of a monitoring system was based on these characteristics. The initial system consisted of cooperative software, some running on a PDP-11/60 RSX-11/M system, and some on an IBM VM/CMS system. The resulting experience permitted an expanded system to be designed and implemented for a DEC MicroVAX II VMS system. This is the current hardware platform, and since this report is not intended to be a historical perspective -no more will be said about previous versions. Rather, that experience and the general characteristics listed above will be turned around and made into a set of specifications which describe the existing software and hardware.

3. Specifications

3.1. HARDWARE

The hardware platform for the system almost defined itself. As noted above, each vendor's Switch product has some serial ports for management. More precisely, each *individual* Micom PABX has several ports, of which three are used in this application. Two are the statistics and command ports as described in section 1.2. The third is another command port for access to a set of *redundant* control logic for that type of Switch. This redundant logic provides a hot-switchover capability. The Bridge has two serial ports, and as there are two production Micom PABXes, as well as a test PABX and another smaller older one, this leads to a total requirement of 12 serial ports. It was thus clear from very early on in the development of the management software that the hardware platform and its operating system needs to be capable of properly supporting many serial ports. A VAX became the natural choice because:

1. The ubiquity of VAXes at SLAC offered plenty of support and allowed another VAX to be used as a backup system.

2. Although the main purpose of any machine used for management functions would be just that, the Network Group needed a VAX to start providing VAX network support for the rest of SLAC.

3. The VAX has good development support, good I/O support, and in particular good generic support for many serial I/O ports. The operating system, VMS, is also excellent in many other respects.

4. A MicroVAX II has adequate speed to do the job.

A MicroVAX II was duly purchased in late 1985 and used as the hardware platform for the management system.

The hardware connections in the MicroVAX system are shown in Figure 2. Not shown are connections to two smaller Switches, or the connections to the redundant command ports of the Micom Switches.

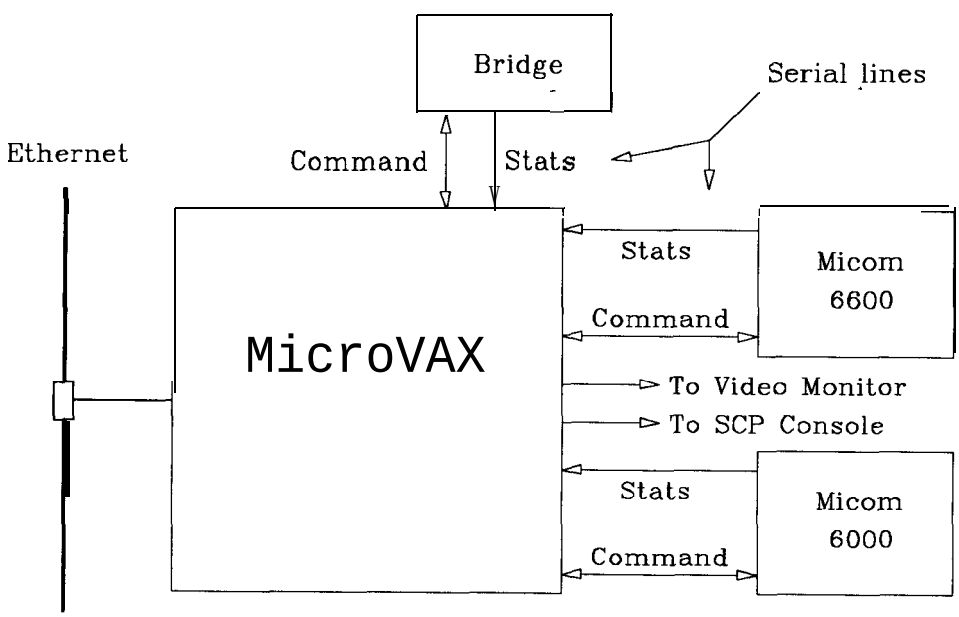

Figure 2: Hardware Configuration, showing MicroVAX connections to the principal Switches.

3.2. MULTIPLEXING ACCESS TO THE COMMAND PORT

Problem Description As mentioned previously, many people at SLAC have a legitimate need to interrogate or configure one or more of the Switches in the asynchronous network. Making the command port available as a class of service on the Switch is not acceptable in SLAC's environment. If the port were connected in this way, only one person could use it at a time. If that person did not disconnect when he-was finished, others would be prevented from using it. Further, it could be hard to discover who was using it and so make it available. This could be important if it became necessary to access the port from home, say, in an urgent situation.

The Resulting Requirement To solve this problem requires that each command port be available with apparent simultaneity from any terminal on the network. The word *apparent* is used since evidently true simultaneity is not possible; there is only one command port on each Switch. Thus if several people need to send commands to the same Switch, the management software needs to queue these requests and handle them one by one. This requirement has various implications which will be explored later.

3.3. SESSION RECORDS

Problem Description A main requirement of the management software is to provide a record of every session that occurs in the each Switch. With such information, one can build up usage statistics for each Switch, track over- or underutilisation of resources, and track the path by which an intruder might have gained access to the network.

The Resulting Requirement The provision of session records has some serious ramifications on the implementation of the management package. These will also be looked at later, as there are other specification items which need to be mentioned first.

3.4. CONFIGURATION MANAGEMENT

Problem Description As noted before, the Micom has only battery-backed CMOS RAM to hold its configuration, whereas the Bridge has a floppy. In practice, such memory-based configurations have been lost, fortunately infrequently. Also should such a configuration be lost, there might well be no record of what the configuration is supposed to be.

The Resulting Requirement The management software must do two things for the Micom.

1. It must automatically track changes made to the configuration and record them.
2. It must store the configuration permanently, and rapidly download it when necessary.

The consequences of this requirement are fortunately very similar to those noted above and will be discussed at the proper time.

3.5. MULTIPLE SWITCHES

Problem Description Early in the development of the management software, it was apparent that the environment at SLAC would encompass many Switches; at present there are six, of various types.

The Resulting Requirement The management software has to be capable of handling many Switches.

3.6. EXTENDED COMMANDS

Problem Description This requirement only affects the Micom, and concerns the command set available to anyone who manages a Switch. The main drawback of the Micom 6000/6600 Switch command set is that there are few ways to logically group

channels. The main provision is for entities called *Destination Groups* (DGs)* which are groups of *called* channels (i.e. channels which for the most part will be connected to computer ports). All called channels† are by definition in a DG. Up to four DGs may be grouped in a *Resource Class* (RC), which is a named item, and whose name is that specified by the user when requesting a resource. However, although channels may be so grouped, almost no commands exist for manipulating them. In particular, it is not possible to put such a group of channels in-service or out-of-service.‡ The network manager has to know the identities of the channels involved, for all commands involving groups of channels.

In addition to this, DGs or RCs are not the only logical groupings of channels that need to exist. As mentioned above, there are some 200 channels which are connected by multiplexors and leased lines to remote terminals or computer ports. Each of these multiplexors constitutes a group of channels which it would be very convenient to be able to treat as a single unit.

When configuring the Micom, various resources within it may be allocated for a particular purpose. For example, a Message Set of up to 128 distinct messages may be used for a Welcome *Message,* for error messages, and so on. Another resource within the Micom is the Protocol *Group* (PG), of which there are also 128. A PG contains all the items such as speed and parity which pertain to the *physical* characteristics of one or more channels. Now, these resources may be allocated at -will, but there is no easy means of knowing quickly which are in use, which are free, and which are in use for a temporary test.

The requirement described here is in fact only part of what one would like to have: easy access to detailed information about the actual state of each Switch.

The Resulting Requirement The multiplexed command port, as defined in sec-

---

* For a full description of the architecture of this type of Micom Switch, the reader is referred to [3].

† The term *channel* is used interchangeably with the term *interface.*

‡ In fairness to Micom, it should be pointed out that some extra commands have recently been added to the native command set. The result is still inadequate, however.

tion 3.2, must appear to provide more commands than are defined in the native command set by Micom. In particular, the following extra functions are needed:

1. **Extended Group Commands.** Several commands refer to a *group* of channels. The management software should extend the concept of a group to anything which can logically be so considered. Thus, all channels in a particular DG or having a certain PG, for example, form a group.

2. **Interface Groups.** The management software should implement the concept of an Interface *Group* (IG). An IG may consist of any desired channels without restriction. This is a most important concept, as it permits the management of groups of calling channels or any other logical group of channels defined by Network Group personnel, by allowing them to be referred to as a single entity.

3. **Generalised Extra Commands.** The management software should be structured to enable extra commands giving information about the resources of each Switch to be easily added. In particular, it should be possible easily to find out which Switch management resources are free and which are in use. In addition, there will be a means of discovering what a given resource is being used for and who has configured it to have its present characteristics.

3.7. UNIFORMITY OF USER INTERFACE

Problem Description Access from any terminal on the Network, mentioned in section 3.2, means in practice at SLAC *any terminal on the standard computer hosts accessed by users, namely the IBM VM system and the VAXes.* The user interface should appear the same from any of these terminals.

The Resulting Requirement The main hardware platform for the management software is a DEC MicroVAX II. Thus access from terminals on other computers requires network connectivity from these other systems to the MicroVAX. In addition, there must be a program on the VM and VMS systems which can use this

connectivity to communicate with the management software, which must for its part be ready for such communication at all times. The VM/VMS program must exist in only one incarnation, thus providing uniformity of user interface.

3.8. SECURITY

Problem Description Security in this context means that while access to Switch information be widely available, only authorised people should be permitted to modify the characteristics of a Switch. In addition, it should not be possible to issue dangerous commands (such as a reset) while a Switch is in normal service.

The Resulting Requirement The management software should implement levels of security and protection as required.

3.9. SYSTEM BACKUP

Problem Description The MicroVAX hardware is assumed to be fairly reliable, nevertheless there are bound to be times when it has to be down to effect software or hardware maintenance. A way has to be found to minimise the effect of these interruptions.

The Resulting Requirement The management system should be able to run on another VMS machine in just the same way as on its normal host, the MicroVAX, without requiring changes elsewhere in the system. In particular, the VM/VMS program should require no alterations and should automatically communicate successfully with the new host.

3.10. USER INFORMATION

Problem Description To assist in keeping the user community informed, the management software should provide information actively rather than merely *passively* responding to user requests. Indeed, in the case of an outage it may not be possible for the user to request information in the usual way.

The Resulting Requirement SLAC has a Video Monitor system, which presents information about the status of the central computers and also public events at SLAC as a series of graphic frames on ordinary color televisions, each driven by a VIC 20 microcomputer. The VICs are fed information from a central IBM PC and the televisions are scattered at strategic locations around SLAC. For the *networking* frame, the PC obtains information it needs from the management software on the MicroVAX. This frame contains information about the major service classes on the various Switches and how full they are, among other things.

3.11. CONNECTION TRACING

To assist in finding how intruders have accessed the system, as well as providing a service to users, it would be useful to be able to determine easily the path that a connection takes through the various Switches, both in real-time, and later, off-line. The management software should implement a trace facility to assist the real-time part of this process.

3.12. HELP

Online help is an important facility. The management software should provide online help, which should look the same from either VM or VMS.

4. General Implementation

4.1. INTRODUCTION - BASIC STRUCTURE

One of the main characteristics of the management software is the ability to handle many events asynchronously. Thus some form of multitasking is a pre-requisite; the question is how to best provide this. The VMS operating system on the MicroVAX provides multitasking and communication between cooperating processes. However, while the management system was being designed, an important software component became available for VMS. The *Network Communications*

Executive (NCX) had been previously purchased by SLAC from Interlan as part of Interlan's implementation of the XNS Ethernet Protocol Suite[4]. NCX consists of a small library of routines which provide simple non-preemptive multitasking within a single program. As NCX had been used very successfully in a previous project, it became natural to wonder whether it would be a good vehicle for the management software.

Briefly, NCX provides scheduling of various processes, forking of new processes, mailboxes, and semaphores. For full details of these and its other features, see[5].

Thus the debate was about which way to provide multitasking for the management software: using several cooperating VMS processes, or using one VMS process which would be internally multitasked using NCX. There are various pros and cons for each choice.

1. **One program as opposed to many.** With NCX the management software could all be contained within one program, simplifying its management. On the other hand, with many cooperating processes one could add and remove individual processes without interfering with the rest of the management software, thus resulting in a more modular system.

2. **Ease of Implementation.** As mentioned previously, NCX is non-preemptive. This means that if one execution thread (i.e. NCX process) is running, it will continue until it makes an NCX call which results in a task-switch. So an appropriately-written sequence of code is *guaranteed* to run without interference. This is important when it comes to accessing common data structures. For example, while a code sequence runs down a chain, it must have uninterrupted access to the chain; otherwise, a pointer may be modified by another module causing the code sequence to fail catastrophically. Using NCX guarantees this will not happen if the above rule is followed. Using cooperating VMS processes requires the implementer to have a mechanism for securing uninterrupted access to a data structure. In addition, for several processes to have such access requires that each data structure be

placed in a shared segment and be mapped to when needed. While none of this is impossible, it obscures the implementation.

3. **Implementing a Listener.** One of the management software specifications called for access to it across the Ethernet, from VM and from any VAX (see section 3.2). The XNS Protocol Suite, implemented on VM and most SLAC VAXes, is a readily available tool for communication from other hosts to the management software. To service many parallel communication requests, XNS provides a way of implementing the concept of a *Listener*. Initially, *connection* requests are made to the Listener, which will allocate a separate communications channel, or *socket* in XNS terms, for each concurrent usage. The initial request is made to a *well-known socket*. SLACnet[6] was designed for VMS as a set of cooperating VMS processes, and when its Listener[171] was implemented various problems arose which were hard to solve on an ad-hoc basis and which in fact were not amenable to a good solution. For example, when the SLACnet Listener forks a process, it is not notified whether the spawn succeeded, and cannot easily pass-data to the spawned process. In addition the start-up time for the spawned process is relatively long. To avoid these problems, we see already an incentive to use NCX for a module containing the Listener and the processes which communicate with those users or Network personnel who need access to information held by the management software.

These factors led to the choice of a single program based on NCX. Thus the system starts to take shape. It consists of a program running in the MicroVAX with access to all the Switches in the network via the serial ports on each Switch. An embedded XNS Listener accepts requests for connections from across the Ethernet and forks *Slave* Processes within it to handle remote requests. At the remote end a program takes users' requests, transmits them to the management software, and displays the responses to the user. This latter program runs under VM and VMS, and for ease of maintenance should to the extent possible be the same on both systems.

The management system running on the MicroVAX is called the *Switch Control Program* (SCP). The user interface program that can run on VM or VMS is called, for historical reasons, MMX.

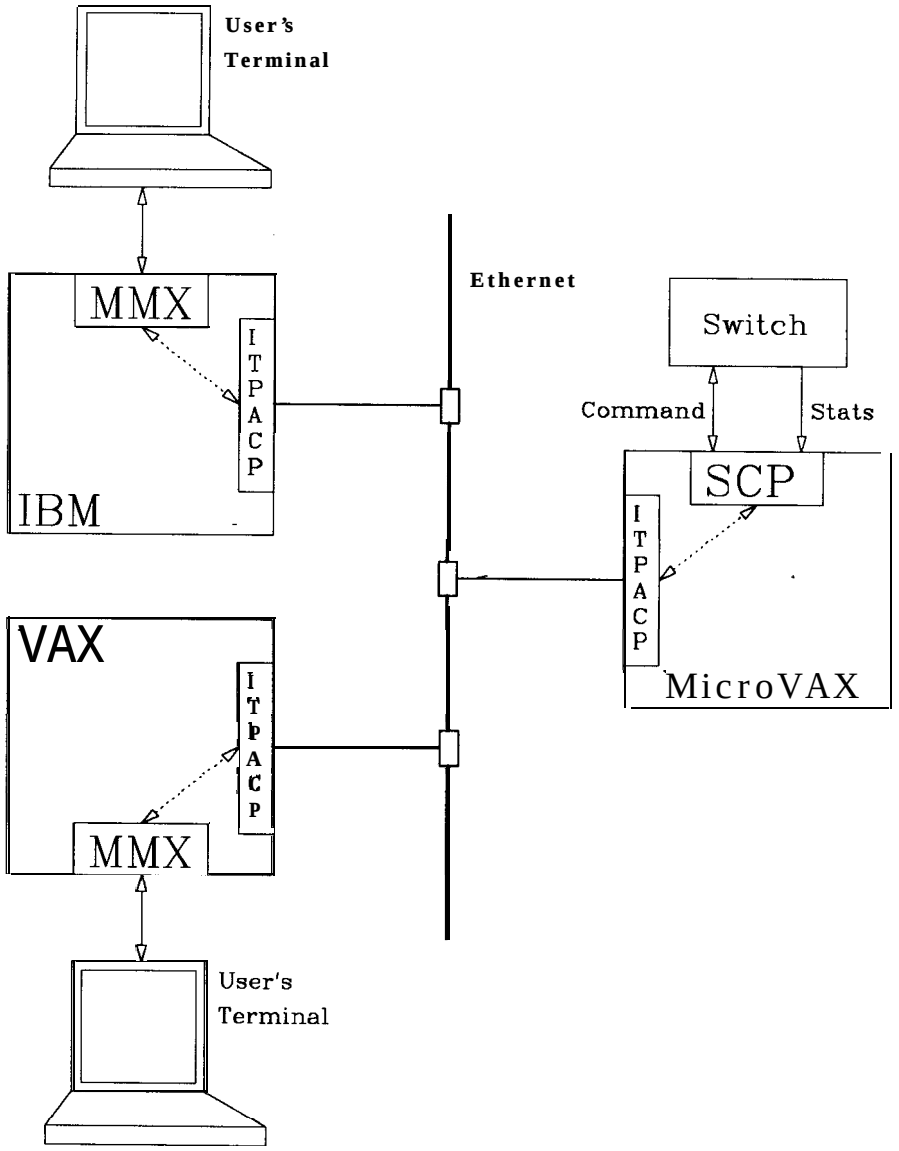

Figure 3: Overall Software Configuration.

Figure 3 shows the relationship between SCP and MMX. The user communi-

cates with a Switch by invoking MMX at his terminal; this communicates with SCP over the Ethernet, using the services of another software module, ITPACP.

4.2. **GENERAL CHARACTERISTICS OF** SCP **AND** MMX

Having reached this point, other characteristics start to take shape. Indeed, it is now time to discuss the consequences, noted earlier, of the various requirements that SCP should satisfy.

Multiple Command Port Access (See section 3.2) This requirement states that various remote MMX users should see what looks to them like their own command port. The easiest way to accomplish this is to have a process within SCP which owns the command port (the *Command* Process) and which handles all communication with that particular Switch. NCX implements mailboxes, and so it is simple enough to arrange that the Command Process wait on a mailbox for work and return its replies to a mailbox specified by the requesting process. The Command Process need have no knowledge of a particular Switch; therefore one re-entrant copy of the code for this function exists within SCP; for each command port there is one NCX process using this code.

For those Switches which have them, and for which it is desired, there is also a separate Command Process for each command port which is connected to a set of redundant logic (see section 3.1).

Session Records and Configuration Management for the Micom (See sections 3.3 and 3.4) It is convenient to group these together, since they tend to lead rapidly to several consequences.

To manage the Micom configuration, SCP must have a model of the configuration. This model is in memory, not on disk, because of real-time constraints; configuration changes may come in bursts and need rapid treatment. Having the configuration in memory allows the creation of session records. When a user connects through a Micom Switch, the Micom generates a *connect record* indicating

the start time and the requested resource. A *disconnect record* is generated when the user disconnects from the resource. These *statistics records* are gathered, for each Switch, by a *Statistics Process* within SCP. The session record is made up by *correlating* the connect and disconnect records, which means that SCP has to keep information at least about the state of each interface within the Switch. It is a short step from there to keeping complete model information about each Micom, in memory. The statistics records as they are imbibed by the statistics process are used to modify the model. In principle the model could be kept on disk in some sort of quick-access database, but that would again complicate the implementation. It would also lead to heavy disk traffic, as statistics records can come at peak rates of 10 to 15 per second.

Of course, merely keeping the configuration in memory as described is not enough. If SCP were to exit or abort, that would be the end of the configuration information. So the entire configuration is written out once a day to a *configuration file*; for safety several such files are kept, the oldest one being deleted once a new, one has been written. To track short-term configuration changes made explicitly by requests from Network Group personnel, changes are saved for a five-minute period and then appended to an *update file*. When the new configuration file is written out the update file is deleted. To perform this periodic work, each Switch's *Housekeeper Process* uses the NCX timer facilities to execute an infinite loop in which it sleeps for five minutes, takes action upon awakening, sleeps again, *ad infinitum*.

Two questions may be posed at this point: how is the configuration initialised, and how is it modified when a change is made to a Switch. The initialisation of the configuration itself has two aspects. When SCP is started, it reads the configuration and update files for each Switch into memory. In this way it has an up-to-date idea of the state of each Switch. The other part of the initialisation deals with the case when a new Switch is added, and SCP has no knowledge of its state.. How this is handled with will be deferred for the moment.

Changes to the configuration are handled by parsing them as they are entered and then changing the model if the updating of the Switch was successful. This leads us to the next facet of SCP. For the Micom Switches there has to be a *Syntax Analyser*, which is not a process but a set of callable routines. If we examine the command set for the Micom 6000/6600 Switch, we see that the *commands* and the *responses* have a similar syntax. This is very convenient; if we choose that the configuration file on disk also use the same format, then one analyser can treat in-put-from all three sources, scan it, and use the results to modify the model.

Session Records and Configuration Management for the Bridge The Bridge differs from the Micom in two important ways. Firstly, the Bridge, having a floppy, can store its own configuration and so need not have it managed by SCP. Secondly, the command port on the Bridge need not be used for configuring the Bridge; any port on the Bridge can get the privilege necessary to do so. This means that an entirely different approach may be taken: SCP merely tracks what it needs of the Bridge configuration and doesn't write it to disk. Commands for the Bridge require no analysis; they are passed straight to it. Analysis of the subset of Bridge-commands needed by the Spy Process (see below) is, however, done. This allows SCP to keep enough of a model so that it can generate session information. It also means that extra summary and status information is available to users and Network personnel.

Audit Trail Kept with each piece of each model is information about who most recently modified it and when, and a comment string that may be used for any desired purpose, for example to describe what this item is used for.

The Spy Process A deferred topic may now be dealt with: for the Micom, how may the initial configuration of a new Switch be obtained, and for the Bridge how can SCP track the configuration? Well, in both cases it can simply ask. Thus, for each Switch there is a *Spy Process*. For the Micom this sends out all possible status queries and uses the results to update the model. For the Bridge, a subset of all the possible status queries is made to allow SCP to build a model dynamically from

scratch. In the Micom the structure of the model is predefined, with a dynamic part being the number of shelves and channels in each. By contrast there is no predefined model for the Bridge; it depends entirely on how many boxes there are, how many ports there are on each, and how many rotaries and classes have been defined.

The Spy processes run continuously, rather than just once to pick up the initial configuration and then stopping. For the Micom, this allows SCP to pick up any discrepancies in the configuration. These are not written to the update file however. To do this would require the Spy to compare existing information with that returned from the Switch, which would be a major extra overhead. It is simply assumed that errors of this type are rare enough that it is sufficient to pick them up when the configuration file is written out once a day. For both the Micom and the Bridge, having the Spy run continuously means that changes in the connection status of interfaces can be detected should any statistics records be lost, due, for example, to overrun within the Switch.

It is possible to configure SCP to delay between each individual query made by each Spy process. For the production Micom Switches, this delay is set to about one second. For the Bridge, this is deliberately set to zero to make the Spy perform each pass as rapidly as possible. This is important for the Bridge, as it helps ensure that the information gathered during the pass be self-consistent. However, *between* passes, the Spy for the Bridge sleeps for one minute. This allows it to automatically track changes in the number of Bridge boxes, rotaries, and classes without any outside intervention. Having these delays means that although the Spy processes run continuously, the CPU load on the MicroVAX is not excessive.

More on the Command Process Some of the characteristics of the Command Process were seen earlier in this section; now is the time to see some extra ones.

Some (not all) of the Micom and Bridge commands are multi-level rather than single-level. Single-level commands are given to the Switch and once the response is obtained the command is complete. By contrast, a multi-level command results

in another prompt as its response to the command. Further information is then input; several prompts may be involved, depending on the command.

Multi-level commands pose a problem; if the Command Process has only the first part of the command at its disposal, then a request from someone else may be used incorrectly as a response. The commands will, in other words, get mixed up. One way to avoid this is to lock out other requests. The Command Process could either indicate that the command port is busy or queue those requests until the-first was complete. This solution is bad for several reasons. Firstly it locks out other requests for what may be a long time, particularly if the first MMX user is interrupted in the middle of the command and then forgets to complete it. Secondly it goes against one of the main design criteria for the management software, of giving each user good access to the command port. Even if the first user responds to the second-level prompts in a timely manner, it could still be a slow business for some commands. Moreover, locking out other users by either method would interfere with the Spy Process.

A better solution is to give the Command-Process all the responses necessary to complete the command at the same time. These need to be packaged by the requesting process and sent to the Command Process as one mail item. It can then deal with more complex commands of this type in the minimum time. This solution works because the number of commands requiring this sort of treatment is small; most commands in the Micom are of the single-level type.

As noted earlier, no analysis at all is done for Bridge commands. Thus the Spy Process could fail badly if a user were to issue a multi-level command for the Bridge via MMX (setting a password, for example). Fortunately it is never necessary to do this.

Initial Startup of SCP When SCP is started, it must be informed about what to do. In particular, it has to know which Switches exist and which serial ports to use for which purposes. This is handled by giving SCP an overall startup configuration file. To make SCP as flexible as possible, this file can contain commands to suppress

specific SCP actions for a given Switch. Thus for example for a test Switch one might suppress the Housekeeper Process, the configuration files for a small test Switch being unwanted. The startup file also contains a list of privileged MMX users and any other information which defines the overall configuration of SCP.

Characteristics of MMX The remote program which the user runs to interface with SCP via the network is called MMX. MMX has several important characteristics. One is that is it completely insensitive to the type of Switch with which the user is communicating. This is important since there are many copies of MMX on different systems. Thus, if a new Switch type is being handled by SCP, or extra commands are added, it is not necessary to replace many now-obsolete copies of MMX. The only time it is necessary to replace MMX is to implement an improvement to the communication protocols used between MMX and SCP. This evidently has to be a rare occurrence.

Another characteristic of MMX is that the versions for VM and VMS be the same. This is nearly true; the VM version has a very useful interface to the text editor Xedit. Some of the commands in the Micom require message editing, but the native Micom editing commands are fairly primitive. It is possible to do the same under VMS but harder to implement, and even when done the interface would inevitably look different.

In section 3.9 a special requirement was described: that if SCP were moved to run on a backup VAX, nothing need be done to MMX other than have the users restart their copies. Because MMX is also an NCX-based program, it is very easy to effect this by having it try to connect to SCP on the two possible hosts. It accepts the response from the host on which SCP actually runs and ignores the error response from the other one. This of course presupposes that SCP is in fact only running on one host, which is always the case. NCX makes this work transparently for the user because it permits MMX to do these two things in parallel.

Connection Tracing With the existence of the online Cables Database on VM

and with SCP having exact knowledge of the state of interfaces on each Switch at any given moment, it is possible to implement a connection tracing capability. This permits one VM user to ascertain the location of another. This is possible even if the other user has crossed say from a Micom Switch to the Bridge network, and then come back to the Micom and been interconnected to another Switch. This works as follows: SCP has a TRACE command, to which may be given an interface number. If that interface forms part of a connection, SCP will return t&he enquirer a list of all the interfaces forming the connection, from the *calling* interface which originated the connection to the *called* interface terminating it. Since SCP is one single program, such information is available *quickly* and reliably to a remote user.

Connections which can pass through more than one Switch is handled by defining *Trunks* within SCP. A Trunk is a set of interfaces in one Switch that are physically connected to a set of interfaces in another Switch.

Each group of interfaces in a Trunk has to be made into an Interface Group (IG) first. Then by using the *TRUNK* command, they may be joined to make up" a Trunk. SCP performs some checks before permitting this: no interface may be in more than one Trunk, and each of the two IGs involved has to have the same number of interfaces. Also, once an IG is a Trunk IG, SCP will prevent attempts to modify it, to preserve consistency; the Trunk has to be deleted first and then recreated once the IG has been modified.

The pairing of the interfaces is defined by the order within the two IGs; it is up to the person defining the Trunk to ensure that this corresponds to the physical pairing.

On-line Help The requirement given earlier was for help to be available on-line. This is centralised into the MicroVAX, so that there be only the one copy of the help files, and that they do not have to be installed on all machines with MMX. The help is hierarchical, and gives details of all available Micom commands. No help is given for the Bridge commands as it is assumed that MMX will not be used

for configuring the Bridge. Help is also given for all non switch-specific commands which have been implemented within SCP.

Security As the management software has been designed to allow easy access to the various Switches, accidental or deliberate execution of commands which may have an adverse effect on a Switch must be prevented. This is done in two ways. Firstly by implementing different privilege levels within SCP as follows:

1. **User Level.** This is the default level if no privilege has been assigned to the user. The user may issue all commands to display Switch information, but none that modifies a Switch characteristic. No help is available at this level, as the user level is mostly intended to permit other programs to call MMX to obtain information for their own purposes. It is not foreseen that the general user will call MMX directly.

2. **General Level.** This is the lowest to be specifically assigned. It allows access to just one command which modifies the Switch configuration. This command allows the modification of just one of the 128 messages available within the Micom Switch. With this many different messages, there are enough to make one available to each VAX system manager at SLAC, for them to configure as desired to give information about outages, for example?

3. **Supervisor Level.** This is the normal level assigned for technicians and other networking personnel. It permits the modification of any part of the Switch configuration.

4. **System Level.** This level allows the execution of commands not needed at the supervisor level. These include stopping SCP, for example, and forcing SCP to write a configuration file for a particular Switch instead of waiting until the file would normally be written, during the night.

Additional security is provided for the Micom by forbidding some commands except under certain conditions. Normally, commands which would drastically

⋆ This feature is not in fact implemented, although the privilege level exists.

affect operations are not permitted. To work on a Switch, for example during an outage, an MMX command may be issued which will put that Switch out-of-service, and which will enable these commands. In addition, the Spy Process will be stopped, and configuration changes will no longer be recorded. This is known as putting the Switch into *maintenance mode.* Security is thus increased by requiring that a special command be issued before potentially dangerous commands may be issued, and by tieing this specifically to Switch maintenance. Another command reverses all the changes given above, returning the Switch to normal service.

-User Information The requirement described in section 3.10 was that the Video Monitor system be able to obtain from SCP the data it needs to build a frame giving networking information to users. This is handled within SCP by a single process (the *Monitor Process).* This uses a serial line to communicate with the Video Monitor system; it waits and is prompted for information by the Video Monitor system.

4.3. **ROBUSTNESS OF** SCP

As SCP is intended to run continuously, it is important that it be very robust. It should be proof not only against common or garden bugs, but also against problems in the elements with which it interfaces. Some examples of this follow.

SCP has to communicate with Switches via serial lines, and as it can send a stream of characters to a Switch with little or no spacing between them, it is clearly possible to overrun the input on the Switch. This is avoided by using the echoed character from the Switch as a timing mechanism. A character is sent to the Switch, and the echoed character is checked to ensure that it is the same as the one sent out. Only then is the next character sent. This ensures that the command has been correctly received; also, should the echo comparison fail, it is possible to abort this attempt to communicate and try again.

Giving commands to MMX will result in SCP returning output. Should MMX abort during this process or be aborted by the user, SCP must not be adversely

affected. It should perform whatever cleanup is needed; in particular the Slave Process handling these requests should detect this situation and exit gracefully.

The XNS services used by SCP to communicate with remote users are implemented in the ITPACP software module. SCP can detect the removal of this module from underneath it and recover; it deletes any Slave Processes, and its Listener attempts periodically thereafter to reconnect to ITPACP.

4.4. DOWNLOADING

Downloading is implemented for the 6000/6600 type of Micom Switch only. As mentioned before, the Bridge holds configuration information on its own floppy and so the possible loss of the configuration is not a consideration. Although the older type of Micom Switch, the 600, is supported in SCP, downloading is not. The Micom 600 at SLAC was upgraded from to a 6600 during the early stages of SCP development and so no further effort was put into supporting the 600.

Due to the way SCP is structured, adding downloading is not hard. The necessary routines to communicate with the Switch exist already. They are combined under a single *DOWNLOAD* command to make SCP generate the commands needed to configure the Switch. When a download is done, it is assumed that the configuration has been reset to the factory default; then only the changes from this default need be sent. Note that during a download, SCP continues with all its other functions. In particular, another Switch may be downloaded at the same time.

Downloading is also available for the redundant logic set if it exists and its command port is known to SCP.

As mentioned in section 4.2, some commands are refused unless the Switch has previously been put into maintenance mode. The *DOWNLOAD* command for the active logic set is included in this group; it may not be downloaded under normal running conditions. The redundant logic set may be downloaded at any time.

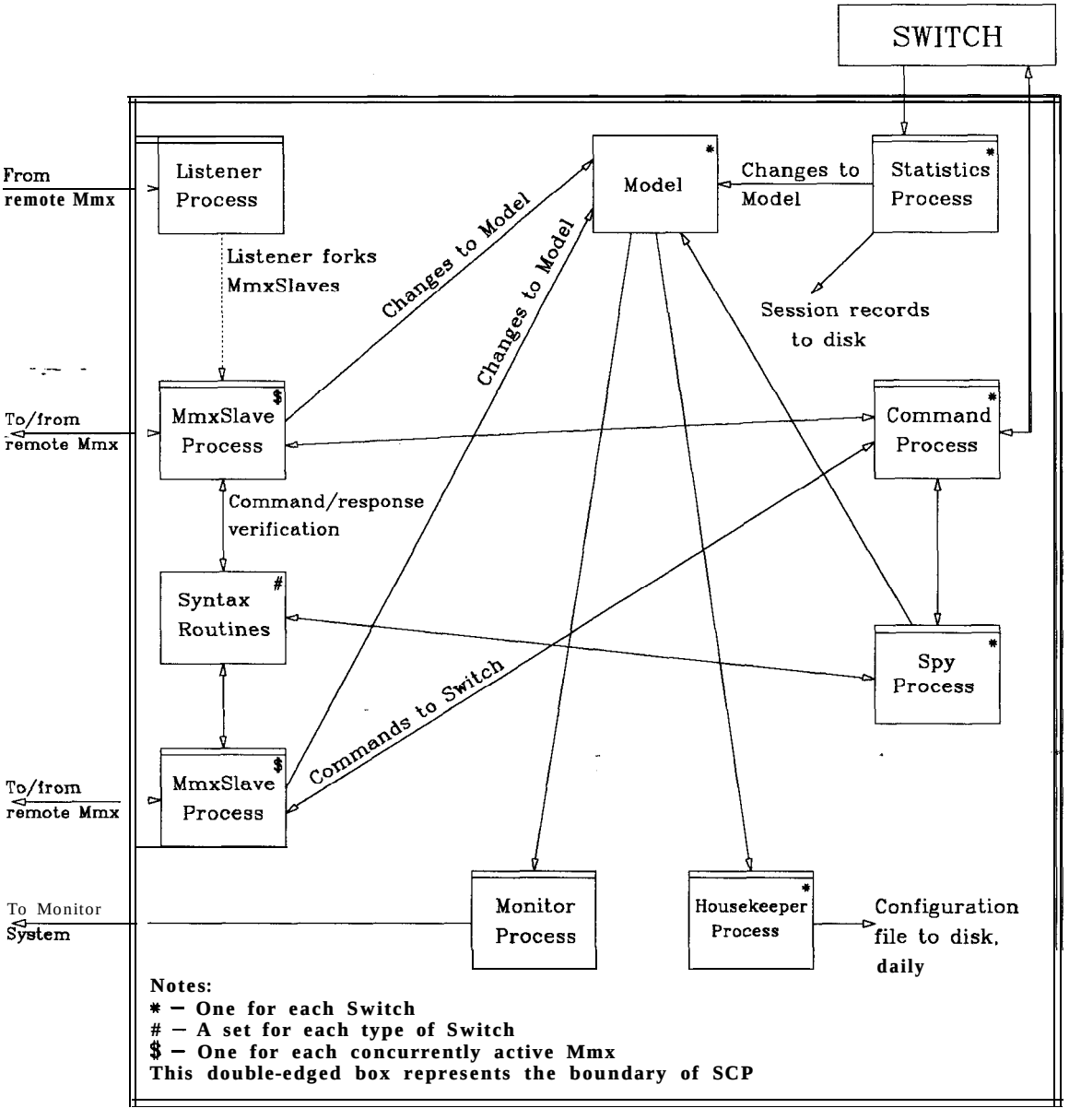

Figure 4: How SCP Processes are related.

4.5. SUMMARY OF SCP CHARACTERISTICS

In the previous section we went through SCP's characteristics by an evolutionary process. This is a good place to summarise them; the reader should also refer to Figure 4 to get a better idea of how the various SCP processes interact.

SCP is a single program based on NCX and runs under VAX/VMS. It interfaces

with all the Switches via a set of serial ports. It contains a *Listener Process* which accepts requests from remote users for access to the management software. These remote users run MMX to get such access, and as each copy of MMX starts, the Listener forks a separate Slave *Process* to handle that user's requests. Many such Slave Processes may be active simultaneously. SCP is thus a *server*, handling many MMX ***clients.*** As each user request arrives, the Slave Process will check that it is a valid command, using the correct set of syntax analysis routines. If the-command involves interrogating a Switch, the Slave Process will generate one or more appropriate commands from the user's original request, and then queue them to the Command *Process* for that Switch. The Command Process, which controls the serial line connected to the command port on the Switch, sends these commands one by one to the Switch and returns the responses to the Slave Process, which modifies the Model where appropriate and passes information back to the remote user.

There is a *Statistics Process* which reads statistics information from another serial line on the Switch.

Also for each Switch a *Spy Process* performs initial reading of the Switch configuration and then continues checking that the model is correct. Finally for each Switch a *Housekeeper Process* performs functions such as writing information out to update and principal configuration files.

SCP is configured via a principal *startup file* that defines the Switches it will manage. It then reads a *configuration* and an *update* file for each Switch before starting its normal management actions.

SCP, then, is able to provide multiple simultaneous access to the various command ports, by means of the Command Processes. It does configuration management through Slave Processes which track configuration changes made by users, and through Spy Processes which continuously poll each Switch for changes. It gathers session information using the Statistics Processes.

5. Experience Gained in Interfacing to a Switch

Through the experience of implementing and running SCP it is possible to see how the external interface presented by the Switch might be improved, and how this would have simplified the implementation of SCP.

5.1. Interface Design Problems

Information Conflicts. Tracking the Switch configuration of a Micom or the Bridge involves two sources of information. One is the statistics port, which produces unsolicited information in real time. The other is the command port used for interrogation of the Switch. The information received by SCP from these two sources can be in conflict. For example, a disconnect record arrives from a Switch via the statistics port indicating that an interface has just disconnected and gone idle. However, the Spy Process, using the command port, has just enquired about.., the status of that interface. If the timing is exact, the Spy Process can believe that the interface is still connected *after* the Statistics Process has just been informed to the contrary. One might expect this to be a rare occurrence, but experience at SLAC shows it is not; it happens several times a day.

This problem cannot be resolved by SCP. In part it is made worse by timing conflicts within the Switch, but the problem mainly occurs because two information sources are used. This will be examined further in the next section.

Menu mode on the Micom. More recent versions of the firmware on the Micom implement a different interface at the command port, known as Menu Mode. This is intended to be easier to use than the command line interface. Unfortunately, while it may be easier for a human, it is much harder for a computer, because Menu Mode is much more modal, and the response to a given input depends on the position in the menu tree. Because Menu Mode is the default, a restart of

the Switch takes it back to that mode.* This can only be recovered manually, by issuing a command from MMX to exit from Menu Mode. A better solution would be for the firmware to read a configuration jumper on the main logic card and to choose an operating mode depending on it.

5.2. BETTER COMMUNICATION WITH A SWITCH

The Micom documentation on the statistics port implies that it could be connected to an external processor for the handling of statistics output. So little is said about this, however, that it would seem that not much thought has gone into how this might be done. As mentioned previously, by far the most common treatment of the statistics port is to connect it to a printer. For this, evidently the output needs to be in human readable form, which it is. This is unnecessary when a computer is being used to interface to the Switch, and is in fact inefficient.

A problem mentioned earlier is that information is gathered from two sources by SCP, namely the command port and the statistics port. This is necessary, as a noisy interface could cause the statistics port to become overloaded trying to produce error messages, and this could cause the Switch itself to lose statistics records if its internal buffers overflow. Each serial line on the MicroVAX is allocated a type-ahead buffer of roughly 32k bytes, so it is fairly unlikely that SCP would be unable to handle the input data stream. Any likelihood of data loss requires that the Spy Process run continuously to ensure that lost state changes of interfaces due to the loss of disconnect records, for example, are recovered.

It should be possible to combine the two serial ports into one,† although having two does permit distinguishing easily between *solicited* and *unsolicited* information coming from the Switch. This is hard to do with only one port because no protocol is available for communication with the Switch which would allow SCP to

---

* Later firmware versions, at least for Warm restarts, remember the mode the Switch was in before the restart. However, this does not cover Cold restarts, which therefore remain a problem.

† An older type of Micom Switch, the 600, had the option of permitting the statistics output to be sent to the command port.

distinguish between these two types of information. It is not the purpose of this note to present a design of such a protocol, however the broad outlines are easy enough to guess at.

A suitable protocol would be binary, reducing the amount of wasted bandwidth. For example, a disconnect record of some 40 bytes could be reduced to 7 or 8 bytes and still contain the same amount of information. Also, unsolicited information could be sent out by the Switch at any time, probably packed into blocks, and *marked as* being unsolicited, so that the Command Process would know to route this information to the Statistics Process. When the Command Process sent a request from a remote user to the Switch, the replies received would again be binary, and the reply block could contain unsolicited information too. Each part of the reply block would need to be marked so that the Command Process could determine which parts were unsolicited and which parts should be returned to the user.

The final necessary feature of such a protocol concerns something which is currently a problem with the Micom Switches--but not the Bridge. The MicroVAX has no means of reliably knowing when output from the Switch has terminated and it is ready to accept another command. What is needed here is a *configurable* end-of-message character. Strictly, this is not available on the Bridge but has been faked by setting the prompt appropriately. With the current Micom firmware one cannot do this and so a timeout has to be employed, which slows communication quite substantially!

It should be possible to reduce the amount of unsolicited information produced by changing the way some minor alarms are presented. At present a minor alarm statistics record is generated for each minor incident for each interface. These records, as noted above, can flood the statistics port; for example a faulty statistical multiplexor might generate a lot of noise on many interfaces. Some control over

‡ The older Micom Switch, the 600, had no prompt at the command level and did have a configurable end-of-line string. A change is planned in a future release of the firmware of the current Micom Switches to cover the problem described here.

the way these alarms are presented is needed; for example, settable thresholds for minor alarms which would generate a major alarm when exceeded. Otherwise, since SCP only reduces these minor alarm records to a count indicating the rate at which these errors are occurring, it would be more efficient if the Switch generated the counts in the first place. It would be necessary to have SCP poll for this type of information on a fairly regular basis, and major alarm information would continue to be produced in the same way as at present.

---Another problem arises as a result of the lack of any communication protocol between SCP and the Micom. SCP has to analyse the text returned by the Switch, and this is designed for human use. The exact form of the text can change subtly between Micom firmware revisions, thus requiring changes to SCP when new Micom firmware is installed in a Switch. Some effort is required to track these changes; with a proper protocol a lot of this effort could be avoided.

5.3. CONFIGURATION OF. THE EXTERNAL INTERFACE TO THE SWITCH

The previous section mentioned the use of configuration jumpers to fix once and for all the behaviour of the Switch in various areas. It is important for operational reasons that this be possible. For example, if an unexpected cold start occurs the Micom is taken back to menu mode. So before the download can commence, the normal (command) mode has to be entered, which as described above is not easy to arrange automatically and may require manual intervention. It is clearly preferable to avoid this situation, and using jumpers is a simple way to achieve this.

5.4. USING MANY SERIAL PORTS

Digital PABXes of the type made by Micom provide one great advantage to connected users: they have a direct connection to the destination device; there is the minimum possible latency created by the connecting device (the PABX in this instance). By contrast, packet-switched mechanisms introduce problems when the

connection is used for interactive purposes. The smooth handling of flow control is one such! The use of separate serial lines for each statistics and command port does mean that the rest of the network is not perturbed by Network Management traffic; these lines can be as busy as desired without affecting the network.

Against this, a packet-switched network provides a transparent means of passing other than just user traffic over it. Much greater bandwidth is available for this than with a serial line. In particular, one may pass management traffic, although its volume should not be so much as to distort the general traffic patterns. This management traffic is then available to be collected centrally, and adding another device gives easy access to its management information. The management traffic from a Switch, however, requires a dedicated serial port, which then has to be connected to the central collection point. At SLAC, this has not been too much of a problem as the main Micom Switches and the Bridge are physically close together, so running serial lines to the MicroVAX is not difficult. It would be a problem if, for example, SCP were required to manage some number of large Switches separated by large distances. Then making the command port reliably available at a distance, for example to perform a download, could be a problem. Clearly the command port could not be remotely connected via the Switch, as it would not be available when most needed.

5.5. Lack of Local Storage on Micom Switches

We saw earlier that one of our main motivations was to provide configuration management for our Micom Switches. This single requirement has complicated the implementation of SCP more than any other, as it means that SCP has to understand the Micom command set and all the configuration details, and the Spy process has to check all the Switch configuration information. By contrast, SCP knows nothing about the Bridge command set and configuration; the Spy does the minimum necessary to allow it to provide session record information and SCP need

§ For an interesting comparison of these two types of network at SLAC, see [8].

not manage the Bridge configuration. In fact, since any terminal connected to the network via the Bridge may be used to configure it, it is never strictly necessary to use MMX to configure the Bridge, although it could be done. Because of this, SCP has not been implemented to be cognisant of those Bridge commands which are multi-level in the sense described in section 4.2, subsection "More on the Command Process". The point is that it would not be much work to implement them, as very few Bridge commands are of this type. Equally, not many of the Micom commands are of this type, and the SCP implementation could have been much simpler had management of the configuration not been a requirement.

One thing that is highlighted by the previous section is how the lack of disk storage on the Micom Switch has complicated SCP. It is unfortunate from this perspective that no provision was made for connecting at least an external floppy drive to the Switch. The purpose of such a floppy would be solely as a backup should the CMOS configuration be lost, and although this normally only happens due to incorrect handling of the main logic cards of the Switch, when it does happen it is a serious problem. Even using the partially optimised *DOWNLOAD* command, downloading takes some 40 minutes to complete, involving several hundred commands.

5.6. THE IMPLEMENTATION COSTS OF SCP

The sort of network management described in this report is based on a number of well understood requirements which characterise the environment at SLAC. There is clearly a mismatch between these requirements and what the various manufacturers have provided. SCP was created to fill the gap. The cost of developing SCP, however, is not small. The necessary hardware cost some $30k, the software effort took perhaps 2 FTE-years, involving writing some 30k lines of C. Clearly such a system is only cost-effective when many Switches must be managed, or when many changes are continuously being made. One small Switch with 250 interfaces requires only limited network management. The problem, though, is that one needs to know before hand whether a small system is likely to grow larger.

If so, it is very important *to start* as one means *to continue,* with an expandable management system from the beginning.

6. The Benefits of SCP

As has been mentioned, SCP was created to meet a specific number of needs. At this point, it is instructive to determine how well those have been met.

6.1. Robustness of SCP and its Design

The question of the robustness of SCP and its design has several facets. First the overall structure of a Command Process, a Statistics Process, a Listener, and so on has endured very well. This part of the design was worked out early in the evolution of SCP and has not needed substantial modification. Some changes have been made, for example to allow a Statistics Process to exist without a Command Process, but these involved only minor changes. The main structure has not needed any modification to date.

Second is execution robustness. Due to careful structuring and attention to detail, it has been possible to make changes to SCP without introducing instability, and the program itself is not prone to crashes or other unexplained behaviour. On several occasions SCP has run continuously for a month or more, only being interrupted when a new version was introduced or because the MicroVAX needed to be shut down.

6.2. Multiplexing Access to the Command Port

This has been very successful. The command ports of the various Switches are accessible from the VM and VMS systems, better than if one connected directly to the command port. The only drawback is the extra latency involved, at least when accessing Micom 6000/6600 type Switches. As mentioned above, this is mostly due to having to use a timeout under VMS to detect the end of transmission from the

Switch. Each user of a command port does indeed have the impression of having sole access to it.

6.3. SESSION RECORDS

SCP writes session records to a file on the local disk, and these files are automatically shipped to the main SLAC computer, an IBM VM/CMS system, for archiving and analysis. The analysis is performed using SAS[9] and a combination of-graphs and other information is always available on-line. One may easily determine, for example, the number of sessions made to a given service class (RC), the number of sessions made by interfaces in a given Interface Group (IG), and the total number of session minutes in a particular month for a given RC or IG. Doing such analysis outside SCP gives flexibility in that new types of analysis may be performed without needing to interfere with SCP.

The ensemble of the statistics records constitutes an important historical record. It may also be used to perform resource recovery. By examining the session records for a period of time from the present back to, say, 90 days into the past, one may ascertain which terminal lines have not been used during that period. The corresponding SLAC Cables Database permits automatic accurate tracing of the interface to the end user's terminal location. Users may then be polled to determine whether they still need an underused resource, or whether it may be reclaimed.

6.4. CONFIGURATION MANAGEMENT

This too is very successful. On the MicroVAX, SCP stores the seven previous configurations for each Switch. In addition, the current configuration is shipped nightly to the VM system, where the archiving system adds to its security. Downloading a 2000-interface 6600 takes some 40 minutes and has fortunately not been necessary in the last two years.

6.5. MULTIPLE SWITCHES

SCP handles multiple Switches very well. If the new Switch to be added is of a type already known to SCP, then a single line in the main configuration file for SCP is all that has to be added. SCP does have to be restarted, but adding a new Switch is rare enough that this is not a problem.

6.6. EXTENDED COMMANDS

The new commands that SCP provides for Micom Switches are extremely useful. Broadly speaking there are two types. Firstly there are commands to display information that SCP keeps for a particular Switch, the model information. This is presented in a more readable way than the raw information that comes from the Switch itself, and contains more information than the Switch can present. Then there are extended ways of interrogating the Switch itself, which usually involve generalising the syntax of an already existing command. These commands make management of the Switch that much easier.

Refer to Appendix A for some examples of extended commands.

6.7. UNIFORMITY OF USER INTERFACE

This requirement, that MMX look the same from VM and VMS, has largely been met. MMX can run on any VM or VMS system at SLAC which is running the XNS protocols. Under VMS, there are, however, some slight restrictions, the most serious of which, concerned with message editing, has already been mentioned. Apart from that, all MMX users see the same external interface.

6.8. BACKUP FOR THE MICROVAX

As SCP is intended to run continuously, provision has been made to run it on a backup VAX 11/780 when the MicroVAX is not available. This normally involves stopping it on the MicroVAX, at which time any pending configuration changes are written to the update files; the session files are closed automatically when SCP stops. Then it is restarted on the 11/780, and as it needs the latest configuration and update files, these are copied over prior to the restart. The various serial lines are switched from the MicroVAX to the 11/780 using a small patch panel. The switchover can be effected manually in less than five minutes.

The other requirement of this backup scheme is that it be transparent to MMX users. That this has been achieved has already been noted, in section 4.2.

6.9. USER INFORMATION

Unsolicited information is made available to users via the Video Monitor system at SLAC (see section 3.10). The networking frame contains information about the load on various Bridge and Micom classes. This is, however, a minor use of the information that SCP passes (via its Monitor Process) to the Video Monitor system. This information tells about the instantaneous load on each service class and each Interface Group known to SCP. Since the Video Monitor requests this information about every 20 minutes, and the information is also passed to VM by the Video Monitor system, the offline analysis package can use it to generate tables and graphs showing, for example, the maximum daily usage for any given RC or IG or the maximum number of queued users.

6.10. CONNECTION TRACING

The *TRACE* command is a recent addition to SCP, so little experience exists of using it other than on the VM system, which has a command to request the location of a user. In this environment it works very well, providing a rapid response

when this command needs to trace the user back through the ensemble of Switches known to SCP.

Refer to Appendix B for an example of tracing.

6.11. HELP

Help is provided by SCP, using VMS library routines to interface to the VMS help facility. The advantage of this is that the help will appear identical to users on VM or VMS, and there is only one set of help libraries.

7. The Future

7.1. INTEGRATING SCP INTO A LARGER NETWORK MANAGEMENT SCHEME

As networks are getting larger, and more and more diverse elements must be integrated in them, management becomes-harder. SLAC's effort to integrate just part of its network into a common scheme has been large. It has meant the implementation of a fairly large software package, with the result that the equipment of only a few vendors (3Com/Bridge and Micom) has been integrated into a management scheme. The SLAC network, overall, contains many other types of element: modems, statistical multiplexors, bridges, gateways, as well as other Ethernet-based terminal servers and also computers.

To meet the challenges of managing such diversity, there is an emerging set of IS0 standards concerned with network management. These standards are probably about two years away from completion, and so some companies have started to implement management schemes which will be in the marketplace before then, as well as planning for schemes which will be fully ISO-compliant. Generally, the interim schemes are as ISO-compliant as is possible given the incomplete state of the standards. The idea is that products will be migrated towards the standards as they appear.

Two such companies are Digital Equipment Corporation, (DEC), and AT&T. For the longer term, DEC is working on its Enterprise Management Architecture (EMA), which is intended to allow diverse vendors' products to be managed across a whole enterprise as a single integrated network. AT&T's long term strategy is based on its Unified Network Management Architecture (UNMA). Both companies have or are planning interim products which will address the short term.

Both EMA and UNMA are hierarchical and will provide an overall framework for network management. They will concern themselves with information management and presentation at a more global level, but will have hooks at a lower level to enable gathering information from specific network devices or other types of device. The idea is that at that level EMA and UNMA will expect to interface to network devices in a standard way; there will be a uniform way to *control* the network devices and to *interrogate* them.

Of course, different types of device actually communicate to the world in different ways. To provide the uniformity demanded by EMA and UNMA at this level, an *agent* must be provided for each device type, which understands its particularities. For Micom and 3Com/Bridge equipment, SCP is an excellent way of providing a more or less ready-made agent. It would be fairly simple to add another process per Switch to it, which could provide the agent interface needed by an overall Network Management system.

It was noted earlier that communication between a Switch and SCP could be improved by providing a protocol for them to use, still communicating via a serial line. A better solution might be to provide integral support for the Simple Network Management Protocol (SNMP) [10] for use in networked environments. This protocol is supported by many vendors, and in fact DEC intends that EMA will be able to use it.

The Computer Center at SLAC is aiming towards running with a fully "lights-out" environment [11], with unattended operation being the norm during off-hours. An integrated Network Management system, capable of monitoring not only the

network but also the environment, will form an important part of this effort.

7.2. THE SLAC PHONE SWITCH

As part of an increasing effort at SLAC to provide more efficient user control of and access to SLAC facilities, it is planned to provide an overall management system for the internal phone network at SLAC. For full details the reader is referred to[12]; what concerns us here is how one might interface to the Northern Telecom SL/1 phone PABX at SLAC. It turns out to have the same pair of ports as the Micom and the Bridge provide. In fact, Call Detail Records (i.e. phone session records) are already being gathered by SCP, and shipped to the VM system just as are the other Switch session records, and are used for accounting. It is planned to provide MMX access to the command port on the SL/1, although the phone PABX interface provided for this PABX is poorly suited for computer interfacing: it is very modal, consisting almost entirely of multi-level commands, and has no end-of-message character. It is evidently intended that only a human operator be able to input configuration changes to the SL/1. Some work has already been done to get around these problems, however; the barriers they present are slight compared with the benefits which will accrue from making such remote access available.

APPENDIX A : Extended Micom Commands

Introduction SCP implements several different types of extended command. The purpose overall is to provide more functionality than the native command set, by giving access to specific information either more quickly or in a more useful format, and by extending the ways in which commands may be used. This appendix gives some examples.

Extending Existing Commands The way in which existing commands are extended is by increasing the ways in which groups of channels may be specified for those commands which operate on such groups. This covers the SET, SHO, INS, and OUT commands. Normally, for such commands, only a simple channel range may be given to these commands. SCP extends this by allowing anything which may represent a group of channels to be used. It will if necessary generate several commands in the native set to send to the Switch. The example which follows shows firstly a native command, and then an example where instead of a simple range being given, the name or a Resource Class (RC) is supplied. SCP generates the required native commands and displays the responses for the user.

```
mmx
MicA?
sho ch 1/9:12

  01/009 PG=05 AG=53   DG=NONE CONN 1200 VM       CALLING   04/108
  01/010 PG=05 AG=53   DG=NONE IDLE
  01/011 PG=05 AG=53   DG=NONE IDLE
  01/012 PG=05 AG=53   DG=NONE IDLE

MicA?
sho ch rc "xa"

  02/017  PG=55 AG=NONE DG=073  IDLE
  02/018  PG=55 AG=NONE DG=073  IDLE
  02/019  PG=55 AG=NONE DG=073  IDLE
  02/020  PG=55 AG=NONE DG=073  IDLE
  02/053  PG=55 AG=NONE DG=073  IDLE
  02/054  PG=55 AG=NONE DG=073  IDLE
  02/055  PG=55 AG=NONE DG=073  IDLE
  02/056  PG=55 AG=NONE DG=073  IDLE
  03/014  PG=55 AG=NONE DG=073  IDLE
```

```
  03/015  PG=55 AG=NONE DG=073  IDLE
  03/016  PG=55 AG=NONE DG=073  IDLE
  03/036  PG=55 AG=NONE DG=073  IDLE
  03/037  PG=55 AG=NONE DG=073  IDLE
  03/038  PG=55 AG=NONE DG=073  IDLE
  03/039  PG=55 AG=NONE DG=073  IDLE
  03/040  PG=55 AG=NONE DG=073  IDLE
  03/073  PG=55 AG=NONE DG=073  IDLE
  03/074  PG=55 AG=NONE DG=073  IDLE
  03/075  PG=55 AG=NONE DG=073  IDLE
  03/076  PG=55 AG=NONE DG=073  IDLE
  03/077  PG=55 AG=NONE DG=073  IDLE
  03/078  PG=55 AG=NONE DG=073  IDLE
  03/079  PG=55 AG=NONE DG=073  IDLE
03/105    PG=55 AG=NONE DG=073  IDLE
  03/106  PG=55 AG=NONE DG=073  IDLE
  03/107  PG=55 AG=NONE DG=073  IDLE
  03/108  PG=55 AG=NONE DG=073  IDLE
  03/109  PG=55 AG=NONE DG=073  IDLE
  03/110  PG=55 AG=NONE DG=073  IDLE
  03/111  PG=55 AG=NONE DG=073  IDLE
  03/112  PG=55 AG=NONE DG=073  IDLE

MicA?
qq
Ready; T=0.07/0.18 14:59:01
```

New ways of Grouping Channels An important concept introduced by SCP is that of the Interface Group (IG). This may consist of any group of channels as desired by the user.* IGs have proved to be very useful for managing the Micom Switch.

```
mmx
MicA?
sho ch ig "dynapac"

  02/065  PG=64 AG=NONE DG=027 IDLE
  02/066  PG=64 AG=NONE DG=027 IDLE
  02/067  PG=64 AG=NONE DG=028 CONN 9600 TYHNET   CALLED BY 08/112
  02/068  PG=64 AG=NONE DG=028 IDLE
  05/037  PG=03 AG=54   DG=NONE  IDLE
  05/038  PG=03 AG=54   DG=NONE  IDLE
  05/039  PG=03 AG=54   DG=NONE  IDLE
  05/040  PG=03 AG=54   DG=NONE  IDLE
```

* We do not deal here with how IGs are set up.

```
MicA?
qq
Ready; T=0.04/0.07 15:50:25
```

New ways of displaying Information SCP allows the user to display the information it keeps internally. The DISPLAY command may be used for this purpose. Note that some of the information in the following examples is the same as could be obtained by using the SHO command; other items such as the IG and the session start and end times could not be.

```
mmx
MicA?
display ch ig "dynapac"

Intface      State AC DG PG InterfaceGroup ConnTo Speed    Class StrtDay/Time
-------      ----- -- -- -- -------------- ------ -----    ----- ------------
02/065         Idle  0 27 64        DYNAPAC
02/066         Idle  0 27 64        DYNAPAC
02/067  <Conected    0 28 64        DYNAPAC 08/112 9600   TYMNET 363 15:09:04
02/068         Idle  0 28 64        DYNAPAC
05/037         Idle 54  0  3        TYMNET>
05/038         Idle 54  0  3        TYMNET>
05/039         Idle 54  0  3        TYMNET>
05/040         Idle 54  0  3        TYMNET>

MicA?
display ch ig "dynapac" session

Interface          State  ConnTo  Speed DTyp StDay StartTime EndDay EndTime
---------          -----  ------  ----- ---- ----- --------- ------ -------
02/065               Idle                User   361 04:00:26    361 04:01:08
02/066               Idle                User   363 04:00:34    363 04:01:16
02/067       <Connected  08/112   9600  Host   363 15:09:04    363 13:44:48
02/068               Idle                Host   363 12:59:57    363 13:00:14
05/037               Idle                Host   363 13:03:21    363 13:07:58
05/038               Idle                Host   363 12:02:59    363 12:12:18
05/039               Idle
05/040               Idle

MicA?
qq
Ready; T=0.13/0.30 15:32:36
```

Miscellaneous New Commands Some information held within the Switch or even within SCP may be hard to get at. For example, if one wished to know which PGs are free and which are in use and what they are being used for, it would be hard to find out without some help from SCP. This comes in the form of the DISP FREE command, and also in the audit trail information. This is shown by the following example.

```
mmx
MicA?
disp free pg

The following PCs are free for use:
   2  14  15  17  18  20  23  24  25  26  27  28  29  30  33  34
  35  36  42  50  52  59  65  66  67  68  69  70  71  72  73  74
  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106
 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
 123 124 125 126 127 128

MicA?
disp pg 41:50 comment

 PG Updater      Date     Time  Comment...
 --   -------    ----     ----  ----------
 41 STREATER 2-Aug-89 18:55:58  Data only at up to 19.2k bps
 42 STREATER 17-Oct-88 15:06:20  For security testing
 43 STREATER 6-Oct-88 16:40:32  For ports at up to 19.2k
 44 STREATER 3-Jul-89 14:12:19  Standard PC for terminals up to 19.2k
 45 STREATER IO-Apr-89 23:31:17  Used by Nevis inbound lines.
 46 STREATER IO-Apr-89 23:46:17  Used by RC NEVIS ports.
 47  MLSSYS  5-Dec-88  19:52:46  Reset Characteristics after SYTEK test.
 48  MLSSYS  6-Dec-88  15:57:49  SYTEK outgoing class
 49   REB   31-Aug-88  14:18:44   AUTOSPD 24-96k for FNAL
 50 STREATER 8-Dec-89 10:13:57

MicA?
qq
Ready; T=0.06/0.11 16:16:08
```

APPENDIX B : Using the Trace Facility

This appendix gives an example of the use of the TRACE command by the *whereis exec* on the VM/CMS system at SLAC.

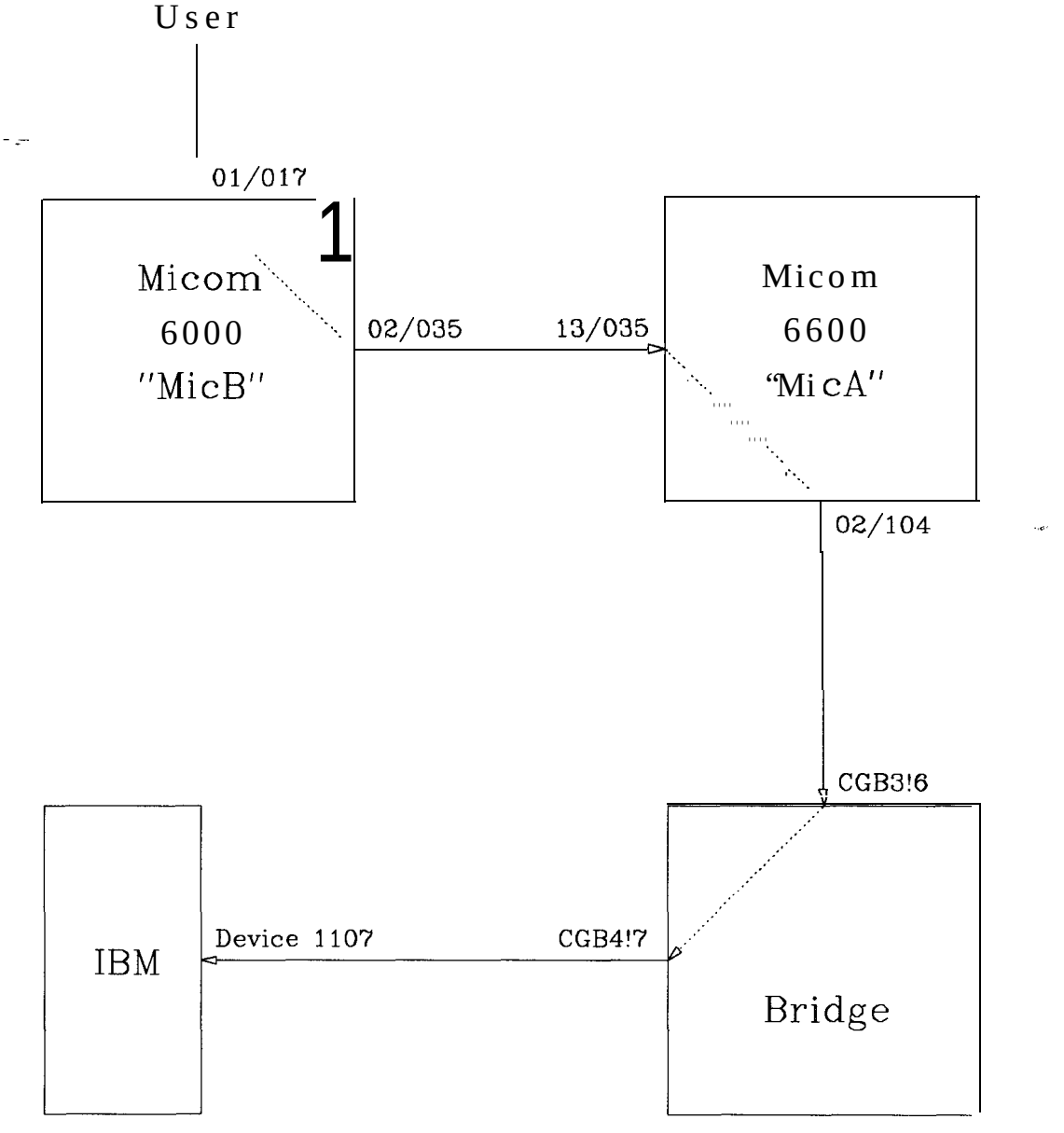

Figure 5: An example of Connection Tracing

Figure 5 shows the author's actual connection path as this text is being written. It is thus a real rather than a contrived example.

From the VM system at SLAC, a user may issue the *whereis* command. This is a REXX exec which, by giving a TRACE command to MMX and using various data base commands, can determine the physical location of a user. In the example here, the command *whereis streater* resulted in the following output:

```
whereis  streater
   DEVICE ADDRESS: 1107
  -PORT-NAME:       SID-07
   TERMINAL(l):     CGB4-07
   PORT-TYPE:       S1
   DEV-ADDR:        1107
   GRAF 1107 PROC 02 LOGON AS STREATER
   MICB:01/017 MICB:02/035 MICA:13/035 MICA:02/104 BRIDGE:CGB3!6 BRIDGE:CGB4!7
   PORT-NAME:       SID-07
   PORT-TYPE:       S1
   DEV-ADDR:        1107
   GRAF 1107 PROC 02 LOGON AS STREATER
   Tracing MICB:01/017...
   PORT-NAME:       MICB01/017
   PORT-TYPE:       M15312
   COMMENTS:        19.2KBPS OCTAL CARD
   CABLE-NO:        D455
   TO:              CGB: COMPUTER CENTER ROOM 302 THIS IS A DB25S CONNECTOR.
   PAIRS:           " IA BDF33 R1C1X5Y12 CGB-302 PIN-3 3449 "
   CKT:             3449
   Ready; T=0.35/0.54 14:40:20
```

The *whereis exec* starts by determining the device address (1107 in this case), and from the Cables data base relates that to a particular port on the Bridge system (CGB4-07 or CGB4!7). It then issues an MMX TRACE command, from which it finds that the other end of this connection is channel 01/017 on Switch MicB. It is then able to refer to the Cables data base again to find the physical location of the user, in this case room 302 in the Computer Center.

ACKNOWLEDGEMENTS

Any interesting project has contributions from many people, and this one is no exception. Many of my colleagues helped in various different ways, and I am grateful to them all. In particular, I should like to mention Teresa Downey, for help with interfacing to the Video Monitor; Les Cottrell, for many useful editorial suggestions, and for the general support which made the project and this paper possible; John Halperin for many suggestions for improvements to SCP; Mike Sullenberger, for helping me navigate the intricacies of VMS; the ever-willing Fred Hooker and his friendly technicians for all the hardware connections; Ginger Brower for answering all my dumb questions about TEX; Janet Dixon for her gentle but persistent pressure to write; and Ilse Vinson, whose excellent editorial assistance helped curb my natural verbosity and turn an enthusiastic collection of words into a readable paper. Thank you all.

REFERENCES

1. Instanet 6000 Data PABX System Description, Stock Number 800-1293-2a.

2. Product Line Overview, 3Com/Bridge.

3. Instanet6000 Data PABX Operator's Guide – Revision E Software, Stock Number 800-1560-1a.

4. Xerox System Integration Standard: Internet Transport Protocols (XSIS 028112), Xerox Corporation, Stamford, Connecticut, December 1981.

5. Network Communications Executive (NCX) Programmer's Guide, Hans Frese and Tim Streater, SLAC-Report-305, October 1986.

6. SLACnet - Implementation and Experiences, R. L. A. Cottrell, T. Downey, H. Frese, C. Granieri, M. Huffer, L. Moss, T. Streater, 0. Saxton, D. Wiser, presented at the SHARE 66 Conference, Anaheim, California, March 16-21, 1986.

7. SLAC Courier and Remote Login Facility, Michael E. Huffer, SLAC-Report-314, April 1987.

8. A Tale of Two Networks, R. L. A. Cottrell, Data Communications, October 1986

9. SAS User's Guide: Basics, Version 5 Edition, SAS Institute Inc., Cary, NC, 1985.

10. A Simple Network Management Protocol, J. Case, M. Fedor, M. Schoffstall, J. Davin, RFC 1067, August 1988.

11. Experience with Unattended Computer Operation, C. R. Dickens and T. L. Cram, ExecuSession, SHARE 70, Anaheim, CA, March 1, 1988.

12. Expanding the Scope of Telecom Management, Janet Dixon and Ilse Vinson, Business Communications Review, September 1989.