Reflections on VM/Pass-Through: A facility for interactive networking

by N. Mendelsohn
M. H. Linehan
W. J. Anzick

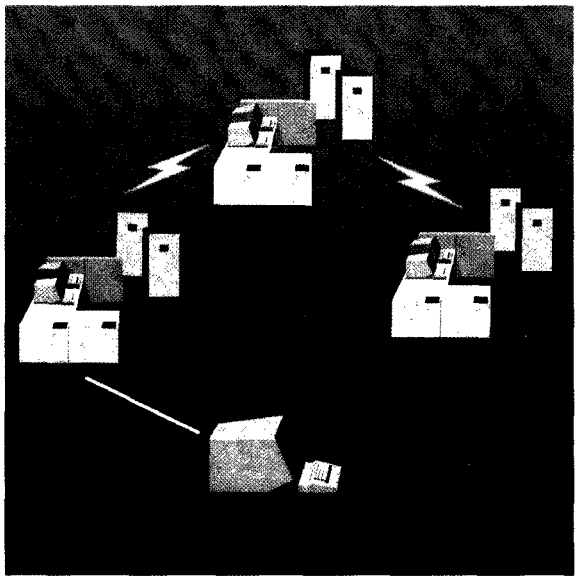
VM/Pass-Through, an interactive networking facility, has gained widespread acceptance within IBM and with IBM customers. Pass-Through allows a single terminal access to many different computers, including those at distant locations. In building Pass-Through, and in observing its growing use, we have had an opportunity to study the practical implications of this facility and of our approach to its design.

This paper is divided into two parts. The first, an introduction to Pass-Through networking, describes features of the system, supported configurations, and use of Pass-Through within the IBM Corporation. A brief history of Pass-Through's development is also provided. In the second part of the paper, Pass-Through is used to motivate a technical discussion of interactive network technology and virtual machine subsystems. Topics covered include appropriate use of the virtual machine environment, choice of routing strategy, and performance considerations. Although the introductory portions of the paper presume no prior knowledge of computer network or operating system technology, the subsequent technical discussions do depend on a basic understanding of these areas.

Virtual Machine (VM) Pass-Through[1] is a simple Virtual Machine/System Product (VM/SP)-based networking facility that provides remote terminal connections for time sharing and data query applications. Although large VM/Pass-Through networks have been used to connect hundreds of computers, typical installations comprise only two or three processors. Because of this, Pass-Through has been designed with particular attention to the needs of small installations, especially those that may not have prior experience with computer networks.

This paper, a retrospective on our seven years of experience with Pass-Through, has two main goals. The first is to provide a simple introduction to Pass-Through networking. We describe Pass-Through's features, examine typical installations, and contrast Pass-Through with other more sophisticated networking facilities. The history of Pass-Through's development and growth is also discussed. In the second portion of the paper, we use Pass-Through to motivate a discussion of technical issues related to the implementation of interactive networking subsystems. We explore the choice of an appropriate subsystem environment, support of virtual terminals, routing strategies, and performance considerations. We also compare the implementa-

Figure 1 Interactive networking using Pass-Through



tion of Pass-Through with that of more sophisticated networks and explain various compromises made during the development of Pass-Through. Readers whose primary interest is in these technical issues may wish to skip directly to the section headed "Architecture and implementation of Pass-Through."

Background

Pass-Through is one of the many facilities developed to meet a growing demand for distributed computation and computer networking.[2] Here we briefly summarize some related commercial and academic developments.

Several companies, including IBM, have developed network architectures to organize the distribution of their customers' computing.[3,4] Carefully planned and often complex in their implementation, these systems are designed to provide comprehensive frameworks for the interconnection of computer equipment. The architectures apply to hardware as well as software, and are often optimized for the needs of large, complex installations. Interface standards provided by the architectures ensure that diverse equipment and software applications can cooperate successfully in a single network environment.

Ongoing university and industrial research projects complement these commercial offerings.[5-7] During the 1960s, a major effort sponsored by the United States Department of Defense led to the development of ARPANET,[8-12] the most famous of all research networks. A high-performance system for interconnecting computers throughout the world, ARPANET provides a vital link among research

Pass-Through provides interactive networking facilities for users of the VM/SP operating system.

institutions and is also a primary test bed for the development of military telecommunications systems. ARPANET is designed to provide subsecond response time for a wide variety of demanding network traffic, to adapt in sophisticated ways to changes in load, and to survive major disruptions in the face of natural catastrophe or war. ARPANET is a classic example of sophisticated, reliable, and efficient computer networking.

In comparison with these sophisticated commercial and academic networks, Pass-Through is a modest special-purpose facility. Pass-Through provides only interactive networking services and is optimized for use in small installations. By specializing, Pass-Through avoids much of the complexity inherent in general-purpose networks; Pass-Through demonstrates that simple approaches can be surprisingly effective for many purposes.

What Pass-Through does

Pass-Through provides interactive networking facilities for users of the VM/SP operating system. With Pass-Through, VM/SP systems may be connected with other systems to create time-sharing networks. Frequently used for remotely logging on, Pass-Through is also valuable in query applications and for remote computer operation.

Pass-Through enables a single VM/SP terminal to access many different computers running a wide variety of software systems. Figure 1 shows a VM/SP display terminal with access to two remote com-

puters. Input entered by a user at the terminal is transferred by Pass-Through across the telecommunications links that connect the systems. This input is processed by the remote system, and response data are transmitted back through the network to the original display unit.

Two simple methods, illustrated in Figures 2 and 3, are available for the initiation of a Pass-Through connection to a remote system. Once a connection has been established, Pass-Through becomes transparent, reflecting every entry to the remote system as though the terminal were connected directly. Although a single entry may be relayed through several computers to reach its destination, Pass-Through handles the necessary indirect routing of data automatically. Upon completion of remote work, Pass-Through restores the original local envi-

Figure 2 A typical Pass-Through session

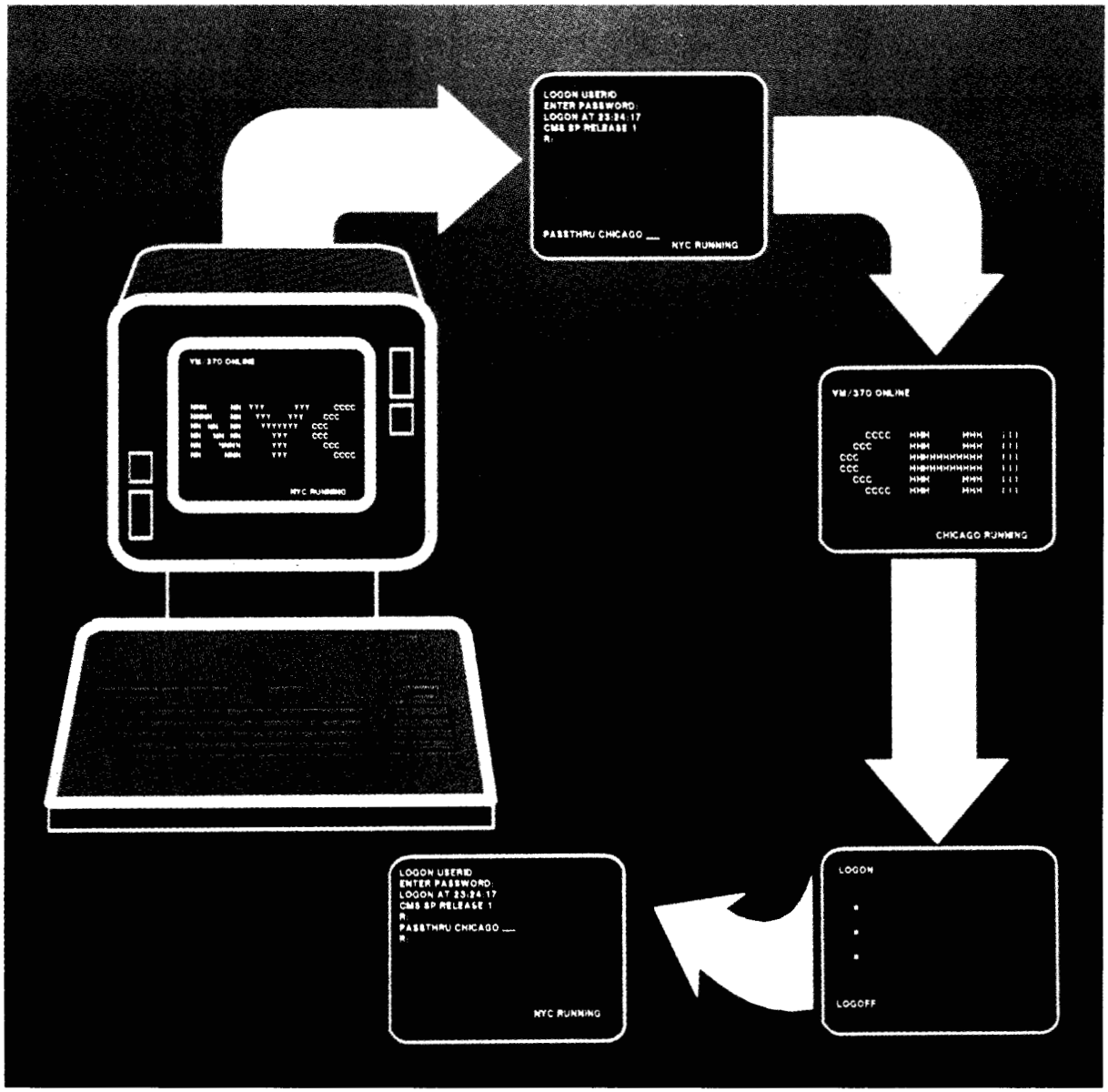
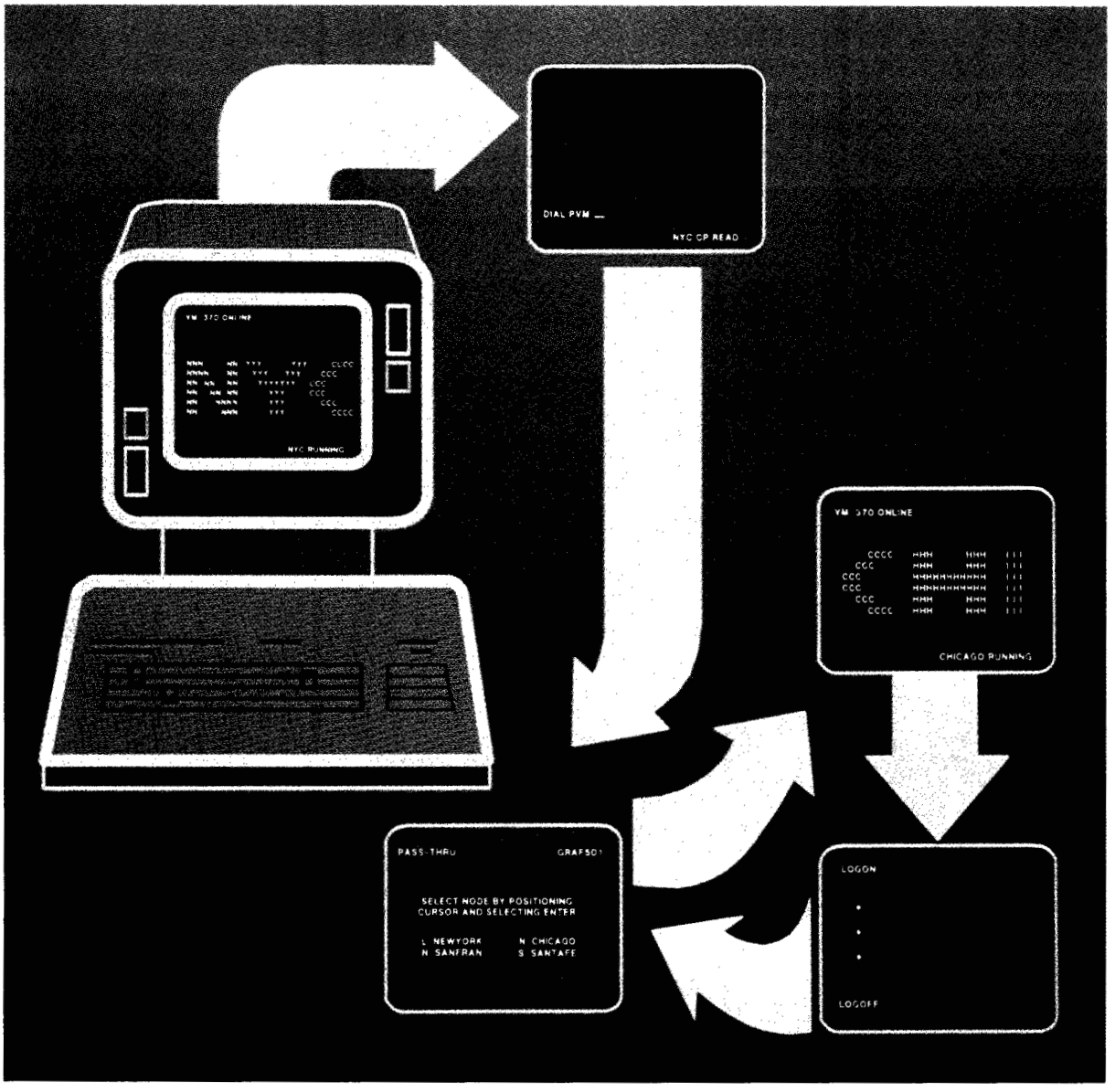


Figure 3 Initiating a Pass-Through session without logging onto a local system



ronment. A wide variety of computer users, including those without technical background, have found Pass-Through to be a convenient and effective means of accessing remote systems. With only a few minutes of informal instruction, most new users become self-sufficient.

Figure 2 illustrates a typical Pass-Through session. Work commences on the New York City system (NYC) with the LOGON procedure. Following this, the user enters PASSTHRU CHICAGO, and is then connected to the Chicago system. Having completed the session, which includes LOGON, processing, and LOGOFF, control returns to the New York session, which is still active.

Figure 3 is an example of using Pass-Through without logging onto the local system (NYC). The

user enters DIAL PVM_, and the system responds with a menu of available systems. After selecting the Chicago system, the user is presented with the CHI logon screen. At the conclusion of the session, control returns to the Pass-Through selection screen.

Many people may use Pass-Through simultaneously; from any convenient VM/SP terminal, each user may access any computer in the network. Pass-Through coordinates the resulting data traffic, routing each transmission to the appropriate destination.

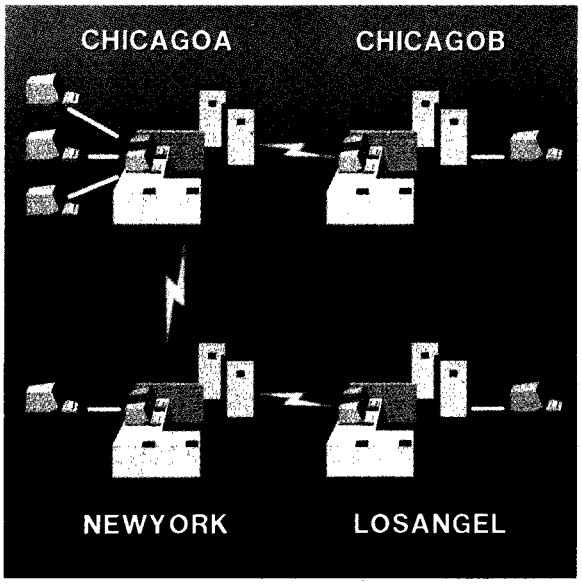
Our experience has been that Pass-Through can usually be installed and tested in a few hours.

A key feature of Pass-Through is its transparency to existing programs. Any program written for use on a local terminal works equally well when accessed remotely via Pass-Through. Consequently, the benefits of Pass-Through can be realized without any programming effort.

Pass-Through supports only terminals in the IBM 3270 family;[13] other VM/SP terminals may not be used to access remote systems. Detailed information on supported terminal configurations may be found in the VM/Pass-Through facility guide and reference manual.[1]

Pass-Through does not directly support the file transfer, electronic correspondence, and batch job submission services commonly provided by more sophisticated networks, but other products available from IBM, notably the Remote Spooling Communications Subsystem (RSCS)[14,15] do provide such services and are frequently used in conjunction with Pass-Through. The IBM Systems Network Architecture (SNA) provides a growth path for customers who wish to combine these functions within a single network.[3]

Figure 4 Four VM/SP systems connected using Pass-Through



In comparison with many other network facilities, Pass-Through is easy to install and operate. No teleprocessing expertise is needed during installation of the Pass-Through software, and the procedures are simple. Our experience has been that Pass-Through can usually be installed and tested in a few hours. Most Pass-Through networks are run without any operator intervention, but when manual control is desired, the entire network may be operated from any of the VM/SP systems. Pass-Through provides a simple facility for relaying network control commands to any machine in the network. Using this technique, an operator located at any site may issue commands to change the network configuration, to send messages to network users, or to query traffic on the network. The simplicity of the installation, operation, and maintenance procedures for Pass-Through seems to contribute significantly to its acceptance. Many prospective users find it easier to try Pass-Through than to undertake a detailed study of its potential.

Typical configurations

Figure 4 shows a typical network of four VM/SP systems connected using Pass-Through. Terminals

are attached to all four computers, and any terminal may access any of the four systems. In order to provide universal computer access without requiring a separate telecommunications link between each pair of systems, the CHICAGOA and NEWYORK systems forward data from CHICAGOB and LOSANGEL. Also illustrated in this figure is the use of Pass-Through as a local terminal switcher. If two or more processors are located in the same computer center (e.g., CHICAGOA and CHICAGOB), the input/output channels of the systems can be directly connected to provide very fast Pass-Through data transfer.[16] In such an environment, Pass-Through may eliminate the considerable expense of cabling each display terminal to several computers.

Within individual IBM sites, Pass-Through is frequently used to grant access to several computers from a single terminal.

Pass-Through itself runs only under VM/SP, but facilities are also provided for limited interconnection with other systems. By emulating the IBM 3271 and IBM 3274 display terminal controllers, Pass-Through allows VM/SP users to access the non-VM/SP systems in a network. Many IBM systems, such as OS/VS2, OS/VS1, DOS/VSE, TSO, CICS/VS, and IMS/VS, support the 3271 and 3274 terminal controllers and may therefore be connected to Pass-Through. Because Pass-Through itself does not run on the non-VM systems, they cannot fully participate in the Pass-Through network. Display terminals connected to non-VM systems cannot access other computers across Pass-Through networks, and non-VM systems cannot function as the CHICAGOA and NEWYORK systems in Figure 4, forwarding data to other machines.

Figure 5 illustrates a more elaborate Pass-Through network consisting of five VM/SP systems and two non-VM systems: a CICS/VS query application and an MVS/TSO time-sharing system. The seven systems are all available from terminals connected to any of the five VM systems. Since Pass-Through routes data through intermediate processors as necessary, the line from Vancouver to Chicago may be eliminated without altering the services available to users. Seemingly redundant connections of this sort are often valuable in improving the performance of the system. In this instance, traffic from Chicago is delivered directly without passing through either San Francisco or Portland.

Use of Pass-Through by the IBM Corporation

The most complex Pass-Through network yet built connects over two hundred thirty-five IBM System/370 computers for use within the IBM Corporation. Diagrammed in Figure 6, this network performs a variety of functions. Pass-Through allows development teams located throughout the world to share access to computer systems, facilitating cooperative projects involving programmers at many sites. A link between Poughkeepsie in New York and Montpellier in France is used for load sharing. Due to the difference in time of day between the two countries, prime-time work from France may be done on under-utilized machines in New York. This experimental project has resulted in significant savings over the past several years. For many IBM employees who use computers in their daily work, access to Pass-Through has been particularly valuable when traveling. Pass-Through allows travelers visiting IBM sites to remotely access systems at their home locations. This provides a convenient means of keeping in touch with colleagues and of reviewing electronic correspondence stored on the home systems.

Within individual IBM sites, Pass-Through is frequently used to grant access to several computers from a single terminal. Systems programmers responsible for maintaining internal IBM systems find Pass-Through to be a convenient way to access the various computers for which they are responsible. Pass-Through can be valuable in remote diagnosis of systems problems, and has been used occasionally for diagnosing problems with the network itself.

The IBM Pass-Through Network has been growing rapidly, complementing the large RSCS/JES2 network (known informally as VNET) that provides file transfer and electronic correspondence services

within IBM. Using multiplexing modems (special equipment that allows several independent data connections on a single telephone line), these two networks often share long-distance transmission facilities, and have together changed the nature of daily communication among IBM sites. The IBM Pass-Through network, and the many smaller networks from which it grew, have also provided a valuable test environment during the development of the Pass-Through product. Using VNET and Pass-Through itself, we are able to stay in close touch with internal users, distributing frequent releases of the Pass-Through software for evaluation and testing.

Evolution of Pass-Through

The program that evolved into Pass-Through was written in early 1975 to interconnect two particular systems within the IBM Corporation. In that config-

Figure 5 Mixed network of VM/SP and non-VM systems

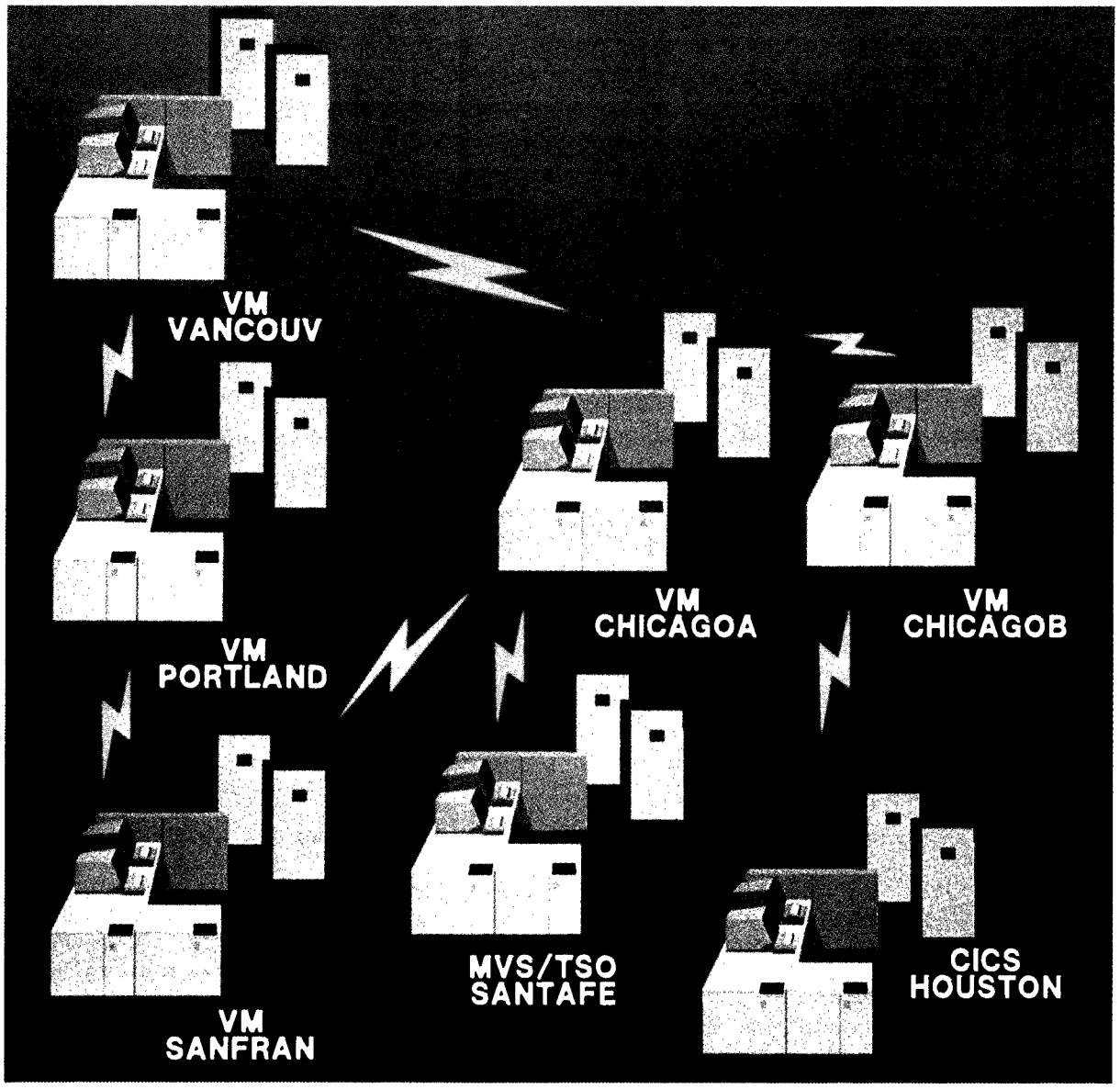
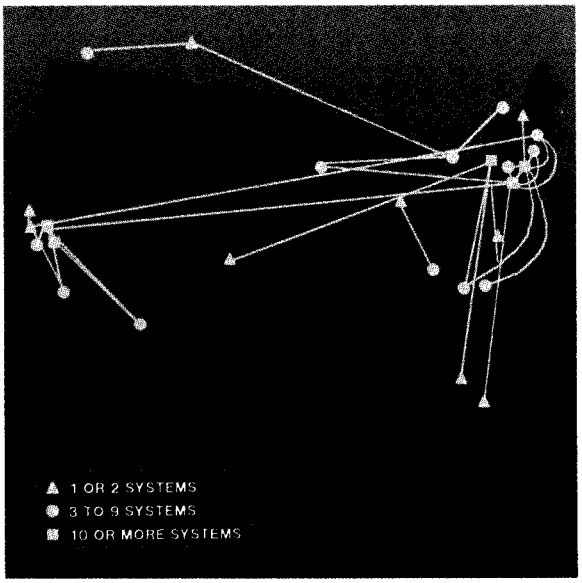


Figure 6 The IBM internal Pass-Through network on March 1, 1982; European systems not shown



uration, users of a VM/SP system were to log onto a remote data base system, execute queries, and then return to work again on the local system. No requirements existed to run the Pass-Through program in other environments, to support complex networks, or even to operate over more than a single link. The decision to create Pass-Through as a product was still over four years away.

The initial version of Pass-Through was designed, implemented, and tested during six months of concentrated work by one person. The result was a simple terminal emulation system with a modular internal structure. Separate tasks (processes) were developed to communicate with individual users and to drive the teleprocessing link; services for buffer manipulation and for other basic support functions were provided in a simple but general manner. A multitasking supervisor was also provided to coordinate activity within the system. Not yet known as Pass-Through, the system entered production use in late spring of 1975.

Other uses were soon found for this program, which could make VM appear to other systems as a standard display cluster. Using Pass-Through, one could potentially log onto any system that supported the widely used IBM 3271 display controller. Indeed, Pass-Through was adopted by many IBM locations during the years that followed, and various enhancements were introduced.

The details of the growing use of Pass-Through are not of great interest, but it is worthwhile to examine closely the characteristics that led to its acceptance.

No requirements existed to run the Pass-Through program in other environments, to support complex networks, or even to operate over more than a single link.

Pass-Through not only provided a valuable service, but it was also easy to install and test. Installers were frequently unaware that Pass-Through, packaged as an ordinary program, actually contained its own small operating system. Assuming that an appropriate hardware configuration was available, Pass-Through could be installed and tested in about a day.

Running in its own virtual machine, Pass-Through posed little threat to the integrity of the systems on which it ran; errors in the Pass-Through program were likely to affect only persons actually using the remote connections. Since it simulated a standard device, Pass-Through was also unlikely to cause failures of the remote systems to which it connected. Other characteristics of Pass-Through proved valuable as extensions were developed. The simple modular internal organization created an environment in which new function could easily be added.

During the first few years, users of the remote connections found Pass-Through to be a simple and reasonably reliable means of accessing other systems. In spite of some performance problems that were later resolved, Pass-Through allowed rapid switching among systems and often relieved users of the need to wait for dedicated terminals. By March

of 1978, more than fifteen IBM sites were using Pass-Through for a variety of purposes.

The earliest version of Pass-Through supported a single link, and could create two-node networks only. With the advent of multiple link support, star-shaped networks became common. Realizing the need for more sophisticated configurations, we began work on a major functional enhancement to the system. Facilities were added for communication between Pass-Through programs running at different sites, and a routing service was also created to direct information through large and complex networks. As part of this effort, an enhancement to the VM/SP operating system display terminal drivers was necessary.

Gradually, the small networks grew together, resulting in a single large network.

When the new support became available, information could, for the first time, be relayed through many intermediate nodes to reach a final destination. All systems became peers, playing equal roles in moving data and allowing any terminal to access any system in a network. Once again, the simpler modular structure of the original system and the generality of the services available provided a natural environment for enhancement.

The new version of Pass-Through gained rapid acceptance within IBM, and many small Pass-Through networks came into existence. Most of these were built of links installed by individual organizations that had immediate needs for communication services. Gradually, the small networks grew together, resulting in a single large network. At no time was there a corporate mandate to build such a network; there was no centralized planning or management. As a courtesy, most sites relayed traffic on behalf of anyone who might need occasional use of their links. In this informal manner, we built the large network that is now in daily use by many hundreds of people.

Pass-Through is the second IBM internal network to grow without centralized management or planning. The Corporate Job Network (the official name for VNET) is used to transfer correspondence, computer files, and other bulk data throughout the world. This sprawling network now connects over six hundred IBM System/370s, and has an estimated growth rate of two systems per week. The development and growth of VNET, in many respects the archetype for Pass-Through, have been documented by Hendricks and Hartmann.[14]

The elements necessary to justify the release of Pass-Through as a product began to coalesce during 1978. As experience within the corporation revealed Pass-Through to be a valuable tool for interactive networking, IBM prepared to release the first of its 4300 series of processors. These inexpensive System/370-compatible machines were well suited for distributed computing, particularly as satellite processors for large centralized hosts.

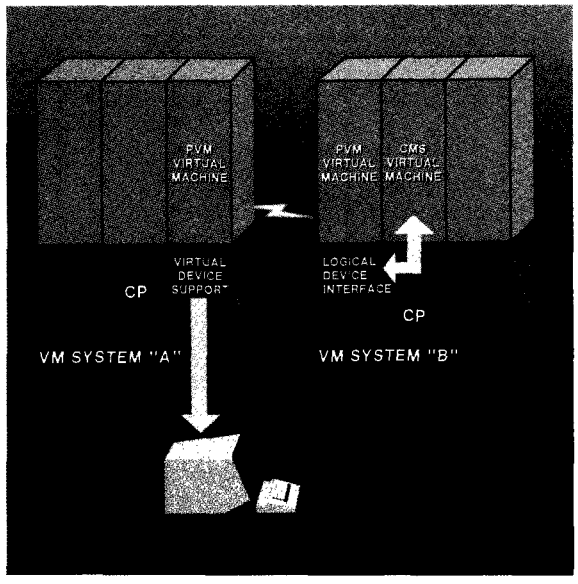
The Remote Spooling Communication Subsystem (RSCS) met the need to move bulk data within these distributed systems, and Pass-Through was seen to provide a complementary interactive networking function. It provided a sensible, inexpensive means of building an interactive network, and opened a migration path into a more sophisticated, comprehensive SNA network. The first release of the VM/Pass-Through Program Product was announced on January 30, 1980. A second release announced in October 1981 provided support for remote printers, for larger display screens, and for the emulation of the IBM 3275 display terminal.

Architecture and implementation of Pass-Through

Although a full discussion of the internals of Pass-Through is beyond the scope of this paper, these sections explore technical topics of particular interest and some of the approaches that distinguish Pass-Through from other networks.

There are several ways to build subsystems within a virtual machine environment. One popular method is to build facilities of the subsystem into the operating system kernel (the protected Control Program or CP). Special operations are then provided by which virtual machines may interact with sub-

Figure 7 The Logical Device Interface (LDI)



system functions. In this respect, the subsystem represents an extension to the architecture of the virtual machine. If Pass-Through were constructed in such a manner, the virtual machines created by VM/SP would be extended to include features for access to their remote counterparts. Although performance is theoretically better than with any other approach, there are the following major disadvantages:

- The operating system kernel is not protected from the subsystem, so subsystem problems may affect the integrity of the operating system.
- Installation involves changes to the control program, thereby disrupting the ongoing operation of the system.
- Poor isolation complicates the debugging of the subsystem.
- Extension of virtual System/370s to include a full network function is contrary to the basic philosophy of VM/SP: the functions provided to each user are to be as close as possible to those described in the IBM System/370 Principles of Operation.

On the basis of experience acquired in developing Pass-Through, we feel strongly that the appropriate environment for such subsystems is within a separate virtual machine. Although some changes to the kernel may be necessary, these should be held to an absolute minimum. There are a number of important advantages to our approach, among which are the following:

- The subsystem is theoretically incapable of jeopardizing the rest of the operating system. Although this is not always true in practice, there is very little risk that programs residing in the Pass-Through Virtual Machine can interfere with system users or their data.
- Installation and maintenance of Pass-Through may be done while the rest of the system continues to operate.

On the basis of experience acquired in developing Pass-Through, we feel strongly that the appropriate environment for such subsystems is within a separate virtual machine.

- Entire networks may be tested on a single physical processor by creating a separate virtual machine for each network node.
- New versions of the network software may be tested without change to the system kernel. Switching between versions of the software requires only that the old version be stopped and the new started.
- Excellent isolation is possible when more than one network is connected to a single physical processor.

These advantages outweigh the modest performance gain of a kernel-based implementation. The ability to install Pass-Through on an operational production system, incurring only slight risk to system integrity, is extremely valuable. We feel that others who are building subsystems, whether networking, data base, device support, or otherwise, should avoid kernel-based implementations whenever possible.

Display terminal simulation

Since Pass-Through was the first interactive network supported from within a virtual machine, new interfaces to the CP kernel were necessary. Our aim was not only to develop a minimum extension to the kernel, but also to provide facilities that would fit naturally within the VM/SP framework. We developed an interface that allowed software in a virtual machine to simulate a locally attached display terminal. The virtual machine could simulate any permissible data entry, and could receive all device orders for the simulated terminal. Figure 7 illustrates the role of the *Logical Device Interface* (LDI) in a Pass-Through network.

Local users of VM/SP access the system in two modes: (1) the LOGON command is used to create a virtual machine, and (2) the DIAL command is used for attachment to an existing virtual machine. An important goal of the LDI was to extend both of these capabilities to users of Pass-Through. This was achieved by intercepting the lowest level of terminal control within CP. Remote Pass-Through terminals and local terminals are supported by the same device management logic. Thus a remote terminal has exactly the same capabilities as a local terminal. Below the device management level, the LDI support provides interfaces to the network virtual machine, while ordinary physical device drivers continue to issue channel commands for locally attached terminals.

The LDI approach is a natural counterpart to the existing VM/SP *virtual device* concept. (Virtual device support is a standard feature of the VM/SP operating system. No changes to VM/SP virtual device support were made during the development of Pass-Through.) Virtual devices are used to provide device control, while logical devices provide device simulations. The LDI allows a virtual machine (Pass-Through in this case) to simulate a terminal to CP, so that CP can operate as though a terminal were connected directly to the system. In Figure 7, the Pass-Through Virtual Machine (PVM) receives the real terminal as a virtual device on system A and creates a logical device on system B. This logical device is transparent to most of CP and to applications running in the CMS virtual machine. Any application that supports a local terminal works equally well through the logical device. Logical and virtual devices are thus distinct concepts that complement each other in creating interactive networks. As a natural extension to VM/SP, the LDI support has also been used by others with a need to simulate terminals from within a virtual machine.

Routing strategies

Support of *multihop* connections in large networks is a complex problem. (Multihop connections are those in which data must traverse several telecommunications links before reaching a final destination.) By optimizing for small configurations, Pass-Through avoids some of this complexity; Pass-Through uses extremely simple routing algorithms that yield surprisingly effective results.

Logical and virtual devices are thus distinct concepts that complement each other in creating interactive networks.

Pass-Through is a virtual-circuit network with static routing. Each network node is supplied with a simple table in a disk file that contains next-hop information for each possible destination. No individual node has a map of the entire network. Each node knows only who its immediate neighbors are and which neighbors provide effective paths to more distant hosts. The tables are maintained manually and must be hand-checked for consistency. Although table maintenance is a difficult chore on the large IBM network, it is usually no problem on smaller network configurations. From the routing tables, virtual circuit paths may be determined for any desired connection.

At the time a user requests a Pass-Through session, a set-up message begins a round trip to the desired target system. Following the directions in the routing tables at each successive node, this message permanently establishes the path to be used for the duration of the session. At each node, data structures are created that monitor the status of the session and facilitate rapid routing of subsequent transmissions.

Since circuit routing is static, error-free connections are maintained without end-to-end checking. Soft transmission errors are recovered by the low-level link protocols and are transparent at the circuit level. Any hard (unrecoverable) errors on a link are recognized unambiguously on both sides of the failure and trigger the disconnection of sessions. All virtual circuit connections on failing links are terminated prior to link reactivation. Virtual circuits provide error-free, end-to-end connections until an intermediate link experiences an unrecoverable error (usually a system stoppage at an intermediate node). When such a failure occurs, users are disconnected from Pass-Through and are expected to retry their connections at a later time. This is a modest inconvenience, since most unrecoverable errors reflect problems that take several minutes or more to resolve.

We have built an operational network and have resolved problems as they have been encountered.

Although these mechanisms are primitive, they have proved to be effective for our purposes. More complex approaches[10] might provide dynamic error recovery, improved performance, or load balancing, but they would be of value only when alternate paths were available. Since Pass-Through is frequently used to support networks of simple topology, the gain from the added complexity would be limited in many cases. We feel that our approach is an effective compromise, weighed against the considerable complexities of implementing and debugging an adaptive routing scheme and the possible overhead in associated protocols.[17]

Admittedly, formal studies would be necessary to fully justify this conclusion. We have not made elaborate measurements of Pass-Through traffic, and have neither modeled nor simulated the possible impact of other networking algorithms on the performance of Pass-Through. Instead, we have built an operational network and have resolved problems as they have been encountered. So far, this process has led us to a simple strategy for routing and error recovery. It must, however, be clear that the more complex approaches that we have eschewed are essential in other circumstances. ARPANET, for example, has performance and reliability objectives that far exceed those of Pass-Through, goals that can be achieved only by using very sophisticated technology. We do feel that Pass-Through demonstrates that alternative, less elaborate approaches to networking can be effective in certain less demanding environments.

Performance considerations

Performance of Pass-Through must be considered from two separate perspectives:

- Pass-Through should be responsive for its users.
- Pass-Through should not place an unacceptable burden on its host systems.

These are potentially conflicting goals. It is easy enough to make a program responsive by allowing it unlimited access to system resources. Conversely, any program may be limited in its impact on overall system performance by sacrificing responsiveness. Analysis of these issues is further complicated by the variety of environments in which Pass-Through is to be operated. The same program that causes negligible overhead on a large processor may represent a significant burden to a smaller system.

Earlier studies[18] have underscored the importance of rapid response in interactive systems. In essence, these studies conclude that an interactive system must respond to short commands in well under a second to avoid disrupting the users' patterns of interaction. Although some installations may choose to run Pass-Through over limited-speed communication lines, Pass-Through is intended to add delays of less than 500 milliseconds per hop under expected operating conditions. Such performance is particularly important when Pass-Through is used as a local terminal switcher. The high-speed data links commonly employed in these short-haul applications are wasted if delays in the networking software are excessive.

To clarify the difficulties that arise in building a responsive system, we consider the significant events that transpire between the time a user sends a query to a remote system and the time a response arrives:

1. Data are entered for transmission to the remote system. The entry is from a terminal owned by the Pass-Through Virtual Machine (PVM).
2. PVM enters a queue of virtual machines waiting to be run. PVM is effectively contending with other users of the system for access to the CPU.

The high-speed data links commonly employed in these short-haul applications are wasted if delays in the networking software are excessive.

3. PVM is dispatched by VM/SP and begins to consume CPU time. During this period, PVM may be interrupted repeatedly for page-in of system code and data areas.
4. The Pass-Through program processes the data from the user and puts it in a queue of data destined for the appropriate outgoing communication link.
5. When the link becomes available, transmission of the data begins. The time required to transmit the data depends primarily on the speed of the communication link and on the amount of data to be sent.
6. Transmission completes. The remote system begins computation.
7. The remote system begins transmitting a response. The time required for transmission is again determined by the length of the response and speed of the communication link.
8. Transmission completes. PVM again enters the queue of virtual machines waiting for access to the CPU.
9. Pass-Through examines the data, determines the display on which the response is to be presented, and writes the response.

Much of the academic work devoted to analysis of computer networks has focused on the time required between events 4 and 5, i.e., the delay in waiting for earlier transmissions to be completed so that the data link may be used. Queuing theory[19] may be used along with modeling and simulation techniques to analyze these delays. Results show that in a heavily used network where links are likely to saturate, these queuing delays may be the limiting factor in network performance. On the basis of these analyses, networks such as ARPANET use sophisticated techniques to reroute data around congested links.[10]

Although some large Pass-Through networks have been built, the primary goal of Pass-Through is to provide a simple, cost-effective approach to the construction of modest networks. Since Pass-Through is intended for bursty transmissions of terminal traffic, but not for bulk transmission of files, sustained heavy data loads are unlikely. In analyzing the installations where Pass-Through is used, we find the following characteristics to be fairly representative:

- Network topologies are simple. Two-node networks, three-node rings, and simple stars are common configurations.
- Even slow data links are statistically unlikely to have long queues. Data links are frequently idle.

Pass-Through has been designed so that dispatching delays can be minimized in a well-run installation.

This suggests that efforts to build sophisticated adaptive flow control and routing schemes would be misplaced. If link delays are unacceptable, the best cure for the problem is to invest in faster links. In analyzing the large Pass-Through network within IBM, we find that user loads on the data links are usually insignificant in determining responsiveness of the network. Critical factors are the number of hops in a route, the speed of the data links, and the responsiveness of the systems supporting the links.

This last factor often dominates network response for short transmissions, and it manifests itself in several ways. In the scenario above, there are two events (2 and 8) at which PVM waits to be dispatched by the VM/SP host. If the host system is

overburdened, a delay of several seconds may be encountered before PVM can even begin to process the requested transmission. We have found that this dispatch delay tends to be the limiting factor in determining the performance of Pass-Through in a heavily loaded environment. This delay is primarily a characteristic of the host operating system dispatching algorithms and only indirectly of the Pass-Through program itself. The problem is compounded when users access Pass-Through from their own CMS virtual machines. The number of virtual machine dispatches per interaction then doubles from two to four.

Pass-Through has been designed so that dispatching delays can be minimized in a well-run installation. The fundamental reason why all users of a computer encounter dispatch delays is that the operating system attempts to ensure fair allocation of the computer's resources. If an individual were allowed free use of the machine, that person might unfairly monopolize the CPU, storage, or other system resources. To mitigate the problem, each user is forced to wait in line before using resources and is periodically returned to the back of the line for rescheduling. Since Pass-Through runs in an ordinary virtual machine, it tends to go through the same delays as any other user in receiving access to the CPU.

Finding that these delays are unacceptable, we recommend that Pass-Through be given high (or even absolute) priority in accessing the CPU. We believe this is justified primarily in view of the studies by Doherty.[18] But what is the price for granting Pass-Through unlimited access to the CPU? It is essentially negligible. Pass-Through is not a compute-bound program. (Our informal studies consistently show that path length per interaction, which does depend somewhat on network configuration and available storage, is not a problem in most installations.) We do not believe that the overall performance of most systems is enhanced when Pass-Through is delayed in its access to the CPU. In short, Pass-Through provides a vital service to its users, and the consumption of resources by Pass-Through is essentially independent of its dispatch priority. Interactive networks should generally receive service from the operating system on a high-priority basis.

Ironically, many networks that are far more sophisticated than Pass-Through seem to be limited in performance by dispatching delays. Our observations have shown that users of TELNET (the ARPANET interactive networking facility) commonly experience host system delays that far exceed those of the network itself. Similar delays are also common in accessing other high-performance networks, including high-speed local networks. Users care little that a network backbone is highly responsive if access is delayed for other reasons. Although scheduling remains a significant problem on some heavily loaded systems, the attention given to this issue has significantly improved the responsiveness of Pass-Through.

Although path lengths for Pass-Through may be short enough to be considered negligible for many purposes, other factors may affect the burden of Pass-Through on the host system. If Pass-Through places too much demand on computing resources, we may not fairly expect high-priority access to the CPU. Use of the memory resource is a particularly significant aspect of the burden of Pass-Through on its host.

If we expect Pass-Through to be given rapid access to the CPU, memory requirements for Pass-Through must be low.

Pass-Through runs under VM/SP, a paged virtual memory operating system. Although Pass-Through is granted the illusion of nearly unlimited memory, excessive use of that resource may be detrimental to system performance. A useful measure of the short-term virtual memory requirement of any program is given by Denning's working-set theory.[20] Informally, the working set is the group of pages in memory that a program requires to execute for a brief period. Classical theories of virtual memory management suggest that it is unwise to run a program unless sufficient space is available to satisfy its short-term memory requirements, i.e., its working set. If we expect Pass-Through to be given rapid access to the CPU, memory requirements for Pass-Through must be low.

Holding down the working set for Pass-Through has the following benefits related to scheduling and performance:

- If a working set is small enough, the host operating system may be able to keep the entire working set resident for long periods. Overhead for the paging of the working set is thus reduced.
- If paging is necessary, delays are minimized for small working sets. These delays result from disk transfers necessary to free space in memory, as well as from those that bring in the Pass-Through working set.
- The *path length* of the operating system logic for paging tends to dominate the CPU overhead for Pass-Through. Excessive use of memory thus translates into additional CPU time billed to Pass-Through.

For all of these reasons, more effort has been spent on reducing the Pass-Through working set than on directly minimizing path length. The two primary uses of memory in Pass-Through are for data buffers and for program code. To minimize the working set for data buffers, a page-sensitive buffer allocation scheme has been developed, whereby the memory allocators in Pass-Through always attempt to allocate new buffers within pages that already contain active buffers. Two buffer sizes are used internally: one-eighth page and full page. The page-sensitive scheme allocates up to eight small buffers in a single page, and clusters buffers within active pages, to the extent possible. In comparison with more commonly used methods,[21] we believe that this approach significantly lowers the working set for a heavily loaded Pass-Through system. When the last buffer in a given page is freed, a special interface to the host operating system is used to indicate that the page is no longer in use and may be replaced. Pass-Through thus provides memory management hints to the host operating sytem.

To minimize the working set for data buffers, a page-sensitive buffer allocation scheme has been developed.

In order to further minimize working set requirements for the Pass-Through program, re-entrant code is used wherever possible. The primary examples are line drivers, which issue device orders to the communication lines, and user-interface tasks, which coordinate interactions with the various users. By using the same re-entrant drivers to drive multiple lines or multiple users, memory requirements for program code are significantly reduced.

We find path lengths to be sufficiently low that Pass-Through may safely be given absolute dispatch priority in most installations.

In summary, we find that the most important practical issues for the performance of simple interactive networks are not necessarily those that have received the widest attention. In many cases, network links are so lightly loaded that complex interactions between users in the data link queues are rare. Simple topologies combined with high overhead per hop reduce the effectiveness of adaptive routing in these environments. We have chosen fixed, predetermined routes as a more straightforward approach. The speed of the data links does set a lower bound on response time for a given interaction, and users of Pass-Through are expected to purchase data links with sufficient speed to provide acceptable throughput.

Operating system dispatch delays are the other significant factors that affect performance of Pass-Through. We find path lengths to be sufficiently low that Pass-Through may safely be given absolute dispatch priority in most installations. Particular efforts have been made to minimize the working set of Pass-Through, thus reducing the Pass-Through burden on main memory and associated overhead for paging.

Concluding remarks

VM/Pass-Through is a simple networking facility that has gained widespread acceptance within IBM and with IBM customers. It allows a single terminal to access many different computers, including those at distant locations. Pass-Through has proved to be easy for most persons to use, and it is easily installed and maintained. These factors, coupled with low overhead and good reliability, underlie the popularity of Pass-Through.

Pass-Through demonstrates that significant networks may be built using simple technology.

Pass-Through demonstrates that significant networks may be built using simple technology. Although adaptive routing and other sophisticated techniques are known to be of value in large mesh networks, many practical networks exhibit simpler topologies. We conclude that simple networking algorithms are more than adequate in these smaller networks. We have also studied issues affecting the performance of interactive subsystems operating in a virtual memory environment. Dispatching delays and excessive working set, rather than simple path length, are often found to limit responsiveness.

In designing Pass-Through to meet the practical needs of its users, great care was taken to lay an early foundation for long-term growth. Simple, powerful internal structures became the basis of an incremental software development process. We have come to accept this as an effective approach to the development of many types of large systems.

The most striking achievement of the Pass-Through system is clearly the large internal IBM network. In addition to its practical influence on the daily work of many people, this network dramatically demonstrates the potential of straightforward approaches to networking software. Sophisticated algorithms might enhance the capacity of this large network, but they would not otherwise improve the services that users have come to value. If such performance is ever required, we will consider enhancement of Pass-Through or the adoption of other networking software. In the meantime, we hope that our efforts with Pass-Through will inspire other innovative approaches to networking and software development.

Cited references

1. *IBM Virtual Machine Facility 370: VM/Pass-Through Facility Guide and Reference*, SC24-5208, available through IBM branch offices.
2. G. Glaser, "The centralization vs. decentralization issue: Arguments, alternatives and guidelines," *Data Base* **2,** No. 3, 1–7 (1970).
3. *IBM Systems Network Architecture General Information*, GA27-3202, available through IBM branch offices.
4. M. Schwartz, R. R. Boorstyn, and R. L. Pickholtz, "Terminal oriented computer communication networks," *Proceedings of the IEEE* **60,** 1408–1423 (November 1972).
5. D. C. Wood, "A survey of the capabilities of 8 packet switching networks," *Computer Networks: Text and References for a Tutorial*, M. Abrams, R. P. Blanc, and I. W. Cotton, Editors, IEEE, Long Beach, CA (1978), pp. 2–44 to 2–50.
6. N. Abramson, "The ALOHA System—Another alternative for computer communications," *AFIPS Conference Proceedings, Fall Joint Computer Conference* **37,** 281–285 (1970).
7. R. Binder, N. Abramson, F. Kuo, A. Okinaka, and D. Wax, "ALOHA packet broadcasting—A retrospect," *AFIPS Conference Proceedings, National Computer Conference* **44,** 203–215 (1975).
8. L. G. Roberts and B. D. Wessler, "Computer network development to achieve resource sharing," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **36,** 543–549 (1970).
9. H. Frank, R. E. Kahn, and L. Kleinrock, "Computer communication network design—Experience with theory and practice," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **40,** 255–268 (1972).
10. F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The interface message processor for the ARPA network," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **36,** 561–567 (1970).
11. M. N. Mimno, B. P. Cosell, D. C. Walden, S. C. Butterfield, and J. B. Levin, "Terminal access to the ARPA network: Experience and improvements," *Computer Networks: Text and References for a Tutorial*, M. Abrams, R. P. Blanc, and I. W. Cotton, Editors, IEEE, Long Beach, CA (1978), pp. 5–36 to 5–40.
12. L. G. Roberts, "Data by the packet," *Computer Networks: Text and References for a Tutorial*, M. Abrams, R. P. Blanc, and I. W. Cotton, Editors, IEEE, Long Beach, CA (1978), pp. 7–4 to 7–9.
13. *IBM 3270 Information Display System Component Description*, GA27-2749-9, available through IBM branch offices.
14. E. C. Hendricks and T. C. Hartmann, "Evolution of a virtual machine subsystem," *IBM Systems Journal* **18,** No. 1, 111–142 (1979).

15. *IBM Virtual Machine Facility/370: Remote Spooling Communications Subsystem Networking Program Reference and Operations Manual*, SH24-5005-1, available through IBM branch offices.
16. *System/370 Special Feature: Channel-to-Channel Adapter*, GA22-6983, available through IBM branch offices.
17. L. Kleinrock, W. Naylor, and H. Opderbeck, "A study of line overhead in the ARPANET," *Communications of the ACM* **19,** No. 1, 3–127 (January 1976).
18. W. J. Doherty and R. P. Kelisky, "Managing VM/CMS systems for user effectiveness," *IBM Systems Journal* **18,** No. 1, 143–163 (1979).
19. L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*, John Wiley & Sons, Inc., New York, NY (1976).
20. P. J. Denning, "The working set model for program behavior," *Communications of the ACM* **11,** No. 5, 323–333 (May 1968).
21. D. E. Knuth, *The Art of Computer Programming, Volume 1, Fundamental Algorithms*, Addison-Wesley Publishing Co., Inc., Reading, MA (1968).

Reprint Order No. G321-5183.

Noah Mendelsohn *IBM National Accounts Division, Scientific Center, 1530 Page Mill Road, P.O. Box 10500, Palo Alto, California 94304.* After receiving his S.B. degree in physics from MIT in 1974, Mr. Mendelsohn joined the IBM Advanced Systems Development Division. His work with IBM has included research on distributed systems, computer networking, and user interfaces. In 1978, Mr. Mendelsohn moved to the IBM Scientific Center at Palo Alto, where he participated for two years in joint research with Stanford University. In 1980, he returned to the university as a student and received his M.S. degree in computer science in 1982. While at Stanford, Mr. Mendelsohn did research on compilers and taught undergraduate classes in computer science. He recently rejoined IBM at Palo Alto.

Mark H. Linehan *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Linehan joined IBM in 1974 after receiving a B.A. degree in political science from Case Western Reserve University. He held various assignments as a VM/370 systems programmer before joining the Thomas J. Watson Research Center in 1979. For the past three years he has worked on VM display terminal support and on user minidisk management systems. He is currently a member of the Technical Planning Staff in the Research Division.

William J. Anzick *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Mr. Anzick is an advisory systems programmer in the Computer Services Department of the Thomas J. Watson Research Center. He joined IBM in 1968 as a senior systems programmer in Harrison, New York. He has worked in software support for systems ranging from the 1130 through the 1401, 1410, and System/360 and System/370. The software support has included such operating systems as BPS, DOS, OS/PCP, OS/MVT, and VM/370. Mr. Anzick has been active in telecommunications on VM/370 since 1974. Since joining the Research Division, Mr. Anzick has been responsible for the development and enhancement of the VM/Pass-Through Program Product.