M-Based Clients and Servers in a TCP/IP Environment

by Lee Hirz

Introduction

The world is becoming one where technologies need to be open in order for them to be competitive. Client-server architectures that allow interaction between applications that run on different platforms create openness where direct interaction is not practical or necessary. The old methods of directly accessing the databases of applications die out slowly as the more modern approaches replace them. Using these old methods, it has always been relatively easy for an application written in M to get data from an application written in another language. Most languages keep data in files that are relatively easy to read. If the file structure is very difficult to read, there is usually a methodology for calling a program to get the data out. But how can M globals be equated with sequential data files? And how can an M routine be invoked to extract data from an M environment? These are not easy questions to answer.

M is an insular language, you see. M applications share data with each other, but do not share with non-M applications. This is where the client-server paradigm starts to work for M Technology-based applications. Most of the major M Technology vendors can now take advantage of client-server technology using the very popular TCP/IP suite of protocols. The TCP/IP suite of protocols can bridge the gap between M and non-M environments. As with ASCII, the protocols are a worldwide standard. As with M, they are hardware- and operating-system independent.

In the June 1993 issue of *M Computing* (the proceedings of the Annual Meeting of the M Technology Association in Washington, D.C.), I discussed the isolation of MUMPS applications from non-MUMPS systems.[1] I discussed the advantages of using TCP/IP channels for electronic mail (e-mail) transmissions, such as speed, easy maintenance, and error-free environments. My concentration then was on the needs of e-mail within the U.S. Department of Veterans Affairs (VA).

All the reasons discussed then are still pertinent, but the focus has expanded. Although many work environments do not have the requirements of the VA for transferring a large amount of data, there is still the advantage of being able to communicate with non-M systems. In addition, providing M-based derivatives of other TCP/IP services such as FTP (File Transfer Protocol), SQL (Standard Query Language), TelNet (virtual terminal linkages), and HL7 services are highly relevant. The possibilities are broader than standard TCP/IP services. Very diverse services, such as patient or library information or a service that provides drug-interaction information, can be supported from an M-based application. Since the VA's MailMan now can communicate in a TCP/IP environment, it is likely that this ability will be expanded into many similarly important client-server areas.

Setting Up for TCP/IP Services and Clients

At the Reno, Nevada, M Technology Annual Meeting in June 1994, I led a discussion session that I titled "How to Run MailMan Across a TCP/IP Channel." When I thought about it, though, I realized that the session could have been better titled. I had solved a problem to accomplish a limited goal. And then I realized that I had really entered into a new arena for M Technology to grow into.

What was the problem? I wanted MailMan to be able to exchange e-mail with other e-mail systems. These other mail systems could be based in DOS, UNIX, or VMS and could be running on PCs, VAXs, or RISC platforms (to name a few). The standard way for e-mail systems to exchange mail is across TCP/IP channels. For some time I had been able to use the TCP/IP channel from M and send mail to other systems. I knew that my system was capable of exchanging mail. But the connection was incomplete—the bridge was only one way. I needed a way for MailMan to receive e-mail from these other systems.

In the client-server paradigm, a client makes a connection to a service. MailMan worked as a client. MailMan could establish a session across a TCP/IP channel and exchange mail with a service. But, to be complete and to receive e-mail from as well as send e-mail to other e-mail applications,

MailMan needed to supply the service as well. Therefore, I coveted a way to create an M service on a TCP/IP communications channel.

The problem was initially solved when Steve Pollock of the Kansas City VA Medical Center wrote C code that made it possible to set MailMan up as a service in the TCP/IP service table. It was a good way to run an M-based e-mail application as a peer to other e-mail applications on the Internet. *M Computing* published the technical paper on the methodology.[2]

This article expands on that original article by taking the concept from solving the problem of running an e-mail system on the Internet to a wider concept of setting up an M service that can supply services to M and non-M clients. This allows us to participate in a much wider world and be a full peer in all client-server-based systems.

The Technical Framework

TCP/IP channels guarantee reliable delivery. Transmitted data are put into packets, checksummed (enumerated), and delivered to a destination where they are removed from the packets. A checksum is calculated and checked against the one sent. If there is a match, the packet is accepted. Otherwise the recipient requests retransmission. This reduces significant work of the application, as will be described.

Most often, TCP/IP uses a synchronous communications channel, usually X.25 across WANs (wide area networks). A TCP/IP channel is really a logical definition. There is often only one X.25 board in a machine that is used to sustain connections to multiple other central processing units (CPU). Each connection uses part of the channel, which is segmented. For instance, a connection from CPU A to CPU B for e-mail needs to be requested by CPU A (the client) to CPU B for the SMTP (Simple Mail Transfer Protocol) service (server). A service is represented by a socket number, in this case, number 25. When the service is requested, CPU A sends out a packet that requests SMTP service from the machine located at an address (known as an IP address). This packet has a return address. A process (known as INETD in UNIX environments) on CPU B identifies this request, and a new logical channel (socket) is assigned for the communication so that socket 25 can be reused for future requests for SMTP service.

There are millions of CPUs connected to the Internet today and every computer has an IP address. For systems to be able to look up IP addresses for CPUs, there is a system for naming CPUs and a system for advertising and looking up the associated IP addresses for them. The CPUs are assigned domain names. The service for looking up the IP addresses for a CPU (domain) is called a DNS (domain name service). Each entity on the Internet needs only to know the address of a DNS service so that it can connect and request IP addresses for the domains it needs.

Integrating MailMan with TCP/IP

Before TCP/IP became an option, all sites running the VA's DHCP (Decentralized Hospital Computer Program) software used asynchronous channels with the XON-XOFF protocol to transmit and receive e-mail. Interference with smooth communications are many in this environment. There can be data lost due to the limitations of the primitive protocols used, such as XON-XOFF. The channel can be interrupted or totally lost. Noise can add erroneous data or can garble transmission. It is a very hostile environment. For these reasons, it has not been difficult to convince sites to convert to the use of TCP/IP channels. About 20 percent of DHCP sites have converted to TCP/IP channels with MailMan. Asynchronous XON-XOFF channels are used as a backup.

In the XON-XOFF environment, error-checking is implemented at the application level. Asynchronous XON-XOFF channels had to be treated as very hostile environments. Transmissions often were retransmitted before a successful completion. Since M is an interpreted language, calculating checksums is an expensive process. Sending checksums across the network adds overhead to the transmission. All the work occurs in the CPU.

It is even better that we can piggy-back easily onto an existing technology. My first experiments with TCP/IP channels set an M process up with the ownership of a specific TCP/IP socket. Socket 25 is the e-mail socket. All e-mail applications that run in a TCP/IP environment connect to socket 25 to transmit messages. The M process, however, could not own socket 25 since that is the sole property of an operating system process usually called INETD. I could use socket 1234 because INETD only owns socket numbers 1 through 1024. Consequently I began an M process on CPU A with IP address 333.33.33.3. Now I could have another M process (a client) connect to socket 1234 at IP address 333.33.33.3 and exchange information with the first. A third process could not talk to socket 1234 at 333.33.33.3. But the standard mail service is expected by each of the millions of e-mail services to be on socket 25, not 1234, so no other e-mail systems will talk to mine. Under these conditions, I either need to run my M process many times (once for each site that needs to contact me) or configure my communications like a rotary-telephone system. In either case, I have jobs running and eating up CPU unnecessarily and I have a complex management chore.

There is a standard scenario in the TCP/IP environment that simplifies all of this in a very elegant fashion available through the intelligence of a program commonly called INETD. INETD performs some interesting services. When a client requests service on a socket that INETD owns, INETD renegotiates a different socket on which actual work will take place. Now the original socket is free so that additional clients can make requests. In addition, when there is no activity, the only process running is INETD. This is really significant. With INETD on my side, a large set of services can be offered, but only one job runs unless a client connects to the system and requests a service.

Thus, MailMan becomes a TCP/IP service, associated (through a table of services) with a particular socket. Now all e-mail systems will be able to talk to MailMan (we will be talking the same language [protocol] and on the same wave length [socket]). Each improvement in speed and reliability achieved in the asynchronous environment had to be bought with sweat or money. Algorithms could be optimized. Time lags during the acknowledgement of sets of data could be put to other uses.

With the use of TCP/IP channels, the algorithm remains the same. But speed and reliability increase with more bandwidth, better hardware, or new physical layer protocols (ATM or "Frame Relay" versus X.25, for instance). As the technology improves, the reliability and speed will improve in a direct one-to-one relationship.

A Link to the Internet

The use of TCP/IP channels in an M environment was a good idea. But I thought of using it only for delivering e-mail. In this case, it worked beautifully. Not only can all DHCP MailMan installations use TCP/IP channels when communicating with each other (as soon as the appropriate hardware and software are installed), but the TCP/IP suite of protocols includes the major protocols "spoken" on the Internet. All mail systems that operate on the Internet use the same protocol and operate the same basic way when sending and receiving mail. They all speak SMTP, the TCP/IP mail delivery protocol, as a minimum. They all operate as a service on socket 25 on the TCP/IP channel. That means we do not need to tell anyone anything about us except our Internet address and they can exchange mail with us.

In fact, VA MailMan speaks SMTP really well. Last year, we started communicating directly on the Internet. We have received messages from literally hundreds of mail systems. The great majority of these mail systems are UNIX-based. The VA is one of very few nodes (fewer than I could count on my fingers and toes compared with more than a million nodes total) that has a service that is programmed in M. Few (if any) of the systems we communicate with know (or care) that MailMan is an M application. When we sell our products, do we really want to be talking about M, or do we want to be telling the buyers about the functionality, bells, and whistles? (These questions are sure to fan the debate about exposing versus not exposing M—the Wizard of Oz—to the myriad technology users out there.)

Is it significant to have created a pathway for M systems to talk directly to non-M systems? Yes! Now, in retrospect, title that earlier discussion session, "How to RUN an M TCP/IP Service." This title implies more of the power that is here. We can now write M code to interoperate in open-systems environments. I can write a service in M and deliver it to a public (and I know of at least one other person who has done the same). I can write an FTP service, perhaps one that delivers M global data to a requester. There can be an M SQL service and an M MOSAIC service.

Practical Applications Using This Model

One definition of medical informatics relates the large amounts of data that computers can store to the needs of medical professionals for supplementary knowledge. If the need for hyperspecialization can be eliminated by providing supplementary medical knowledge on request or even as a general part of the system functionality, then patient care should improve.[3] With a few word changes, this informatics definition could apply to any discipline. For instance, in business: "One definition of business informatics relates the large amounts of data that computers can store to the needs of management for supplementary knowledge . . ."

Using the client-server model to enable different systems to query each other for supplementary knowledge would be a large step forward. For example, a system that reports and analyzes relationships of groupings between laboratory results against possible diagnoses could be queried. When prescribing a drug for a patient, a database that would accept inputs of the drugs the patient uses along with important demographic information could warn about possible interactions, whether beneficial or harmful. A doctor could state the patient's need, and receive a listing of the appropriate drugs together with detailed analysis of the conditions that certain medicines or procedures might cause or improve. The goal of DHCP is to help to give VA hospital patients the best care possible. The ability for DHCP applications to interact with applications from other public and commercial sources can improve the care we give to veterans today.

In an interview published in *Government Computer News*, Robert M. Kolodner, M.D., said that DHCP needs to build the best information system at VA health-care facilities that we can afford.[4] Clearly we cannot supply every software application that may be needed now or in the future. But we can make our system an open one, bridging the gap between what DHCP does very well and the applications that may be needed.

Beyond Today's MailMan

E-mail is very important at the Veterans Health Administration (VHA), which is actively establishing its employees as e-mail users. There are tens of thousands of users linked in the e-mail system across more than two hundred sites. The national e-mail system at the Washington Information Systems Center, FORUM, has more than thirty-five thousand active users and sustains more than eight thousand logons per day. FORUM averages one-half million message units delivered on an average work day, a total of almost six million lines of message text. They are interconnected with VA MailMan, a product whose communications are based on IPS (Internet Protocol Suite) standards and which can interoperate with all similar standards-based products.

Public access to VA MailMan is easy. MailMan is available free as public domain software. It runs on many different platforms, even small personal computers. This makes it a perfect system for the public to acquire, install, and use. An oversight group assures that there is appropriate security and use of the system by all users.

On the national mail node, there are many nongovernment and non-VHA users (such users have access at the behest of the VHA to conduct activities that are in its interest), participating in a supportive and cooperative fashion. There are many non-VA sites running MailMan, notably, the U.S. Indian Health and Public Health services, and the Smithsonian Institution has a MailMan site communicating across TCP/IP channels and using the Internet. Also on the list are Mercer Medical School in Macon, Georgia; University of Tennesee Veterinary School in Knoxville; and the West Virginia University/Rural Healthcare Program based in Morgantown, West Virginia. Outside this country, MailMan is used by the German Heart Institute in Berlin and sites in Finland and across Africa. In addition, there is the ability to forward mail to non-VHA users at their home systems.

DHCP's e-mail system, MailMan, now can communicate in a TCP/IP environment as well as in all the other communications environments it has always supported. FORUM (the national e-mail node of VHA) sent more than eleven thousand messages to Internet addresses and received more than fifteen thousand from Internet addresses during a recent month.

With the release of MailMan 7.1 in June 1994, there are already apparent services needed. Adding a DNS (a service that keeps track of the routes that should be taken to send e-mail) is necessary. Also needed is an X.500 directory service, a server that will give people e-mail addresses of other people they wish to contact, even if those people's accounts are on computers thousands of miles and many network hops away. A patient locator would be very desirable, too.

Now that we have e-mail experience with this methodology, my expectation is that we will use the same modus operandi for other applications. Since DHCP needs to integrate with non-M applications, the applications likely to use this technology first will be those that can supply services to non-M systems and those that need information (as clients of non-M systems).

The Patient Data eXchange (PDX) package needs to dispatch information about patients quickly while users wait. The DHCP messaging system team has expressed interest. The MailMan development team is working on at least three additional services: a domain name server (DNS); a user locator (that can be expanded to include patients); and the implementation of Post Office Protocol. This is another protocol from the suite of TCP/IP protocols defined in the federal government's RFC (Request For Comments) 1460 that will enable other mail applications to get mail from a MailMan-based system so it can be read and otherwise manipulated outside of MailMan. There are other M applications that are using TCP/IP channels for communications: I know of an SQL server that provides SQL services on a standard TCP/IP socket.

Conclusion

The future of M Technology can be one of explosive growth since M has advantages over other languages that cannot be ignored. Some of the most promising areas for growth are in client-server applications where turnkey systems supply value-added services. Setting up M applications as TCP/IP services can allow non-M applications to get information from M environments using internationally standard client-server methodologies. It is very important that M Technology users be given access to this new capability so that they can interact and become peers with other technology communities in the open-systems environment today and in the future.

Lee Hirz has been with the VA for nearly eight years, devoting his experience to M using VA FileMan and Kernel. He is the lead programmer and project manager for the VA MailMan application. Previously, he worked in a MIIS environment concentrating on financial and clinical applications. His degree is in business administration and accounting. He loves baseball. You may contact him on his Internet address using hirz.lee@forum.va.gov.

Endnotes

1. L. Hirz, "TCP/IP Communications in a MUMPS Environment," *M Computing* 1:3 (June 1993): 55–57.
2. Hirz.
3. G. Francois, "Scientific Context and the Human Meaning of Informatics Applied to Health Care," *MD Computing* (May-June 1994).
4. "For Unifying VA Systems, the Doctor Is 'in.'U'." *Government Computer News* 12:25 (November 22, 1993): 12.

Additional Information

1. M. Meighan, A.C. Curtis, and L. Hirz, "Servers and Filegrams," *MUG Quarterly* 22:3 (June 1991): 89–92.
2. L. Hirz and A.C. Curtis, "E-mail and the Corporate Sub-culture," Discussion Session, June 1992 MUMPS Users' Group Annual Meeting (Phoenix, Arizona).
3. T.W. Malone and J.F. Rockart, "Computers, Networks and the Corporation," *Scientific American* (September 1991): 128–136.