Networked Terminals in a UNIX/M Environment

by David Schulz

Hapo Federal Credit Union is a $100 million financial institution that has used M since 1981. In September 1991, the credit union migrated from a Digital Equipment Corporation (DEC) PDP 11/70 running InterSystems' M/11+ to a networked configuration consisting of an IBM RS 6000 (Model 550) and two secondary 486 PCs all running InterSystems' M/SQL for UNIX systems.

This article discusses issues related to character-based terminal (VT-type) and serial printer connectivity utilizing terminal servers in an Ethernet network. Although specific vendors are mentioned, this article is not a review of vendors. Its sole purpose is to share what we have found that actually works in practice.

Direct Connect Versus Networked Terminals

For years, life was simple. The RS-232 plug on the back of a terminal ultimately connected to a corresponding RS-232 plug on the back of the computer. It may have snaked its way under counters, into jumper boxes, through ceilings, and out of jumper boxes, but when a user logged on, the same device number was assigned. That device number always corresponded to a given plug on a distribution panel, and best of all, $I was always that device number.

Printers were simple, too. A printer's consistent device number was known and tables could be built to "tie" terminals to printers. The teller's receipt on terminal #100 could be automatically sent to device #101.

Finally, in the stand-alone M/11+ environment, terminals could be "tied" to specific applications so that at the press of RETURN the user was directly in the application and subject to total control. The user never saw M and certainly never saw an operating system. (Indeed, there wasn't one other than M.)

In the world of networked terminal servers, devices are connected directly only as far as the server. From there, it is onto the Ethernet and to one or more computers available on the network. There is no need for A/B switches. Every computer on the network is potentially at one's finger tips. Further, every printer on the network is potentially eligible to receive output from any computer on the network.

As is often the case when changing environments, with something gained, something is lost. In this network, with each new log-on connection a new $I value is assigned. It is no longer possible to rely on $I being a consistent device number relative to a physical location. The application can no longer determine that the user is on device X located at teller station Z. Therefore, it is necessary to send the receipt to device Y which is also at teller Z. Also, how can the application even find a printer in this network? What if it is desirable to send some e-mail to a terminal, for example? To top it all off, when a connection is made, the user has an operating system to log onto prior to even getting to the M application.

Wanting It All

The objective of the credit union's configuration was to take advantage of the flexibility of networked connections without sacrificing abilities upon which existing applications were built.

- The ability to determine the physical location of a given terminal in order to determine the physical location of a printer corresponding to that terminal.
- The ability to "tie" terminals to applications, so that when users press RETURN they immediately roll into the application, bypassing any UNIX log on.
- The ability to open and output to a connected terminal that is not running an M application.
- The ability to print over the network and have printers shared by multiple computers on the network.

The LAT Protocol under UNIX

Ki Research of Columbia, Maryland, provides DEC's LAT (Local Area Transport) protocol for more than thirty different UNIX hardware platforms. LAT is particularly efficient in handling character terminal communication over a network,

while also providing host-initiated connections ("reverse-LAT") to printers also on the network. Utilizing Ki's product, known as KiNET, in conjunction with M/SQL's ability to communicate with UNIX via function calls and "named pipes," we were able to meet all of the goals stated above.

KiNET allows the system administrator to create LAT "services" that are broadcast for availability on the network. Users stipulate at the terminal-server level the service to which they desire connection. At installation, KiNET creates an initial service to the host, which is a standard UNIX log-on process. Users connecting to this service appear as any other UNIX user.

Having It All

In addition to providing a standard UNIX log-on service, KiNET affords the system administrator the ability to create additional services that connect to a host and *proceed directly to the operation of a specified task:* for example, a user-written shell script. The system administrator may optionally stipulate the UNIX user identification associated with the service so as to maintain security on a UNIX level if desired.

The primary machine in the network is the RS6000. The PCs are connected to it in the event of its failure. We anticipated that our users would typically desire to establish connection to the RS6000 and maintain that connection until specifically requesting to terminate. They were accustomed to pressing RETURN, running an M application until completion, and eventually seeing "Exit." An additional step of connection with each iteration of that process was undesirable.

". . . security is maintained at the application level . . . the overhead . . . is negligible, in management's view."

A UNIX shell script was developed to be the "front-end" for KiNET connections, the task that KiNET executes upon connection. The shell script was designed to be a loop which reads 0-n characters and then proceeds to M and a central M routine. Upon halting, M passes control back to the shell script which displays "Exit" on the terminal and waits for the next iteration of the loop. Should the user type "BYE" instead of simply pressing RETURN, the shell script terminates and the connection is terminated. Control is then passed back to the terminal server.

For connections established in this method, KiNET maintains special UNIX environmental variables containing the physical port number associated with the server and the server's name or address. Once control is passed to the central M routine, these environment variables are examined via M to UNIX communication methods. We then may use this information to pass control effectively over to an application and directory that the physical terminal is "tied" to.

By maintaining the connection to the host until the user expressly terminates it, we are able to open and output to terminals that are in "Exit" (not running an M application). As a device initially connects, we maintain a table of its physical address and its $I value. Upon disconnection, this table also is updated. Prior to sending output to the terminal, the software is able to determine if the connection is still established by checking the table in conjunction with a UNIX system status of devices running the shell script.

KiNET creates consistent specific device names for printers that ultimately link to the physical port on the server. Thus, printer device locations are always consistent.

From the average user's standpoint, the system appears as it always has. There is no interface to UNIX and security is maintained at the application level. Further, the overhead associated with this method is negligible, in management's view.

Summary

We have found that utilizing LAT in a networked UNIX environment allows desirable flexibility and speed with a comfortable and secure user interface. ❖

David Schulz is data processing manager for Hapo Federal Credit Union in Richland, WA. He has been in charge of hardware and software development for the credit union since 1981. He previously wrote an article, "MUMPS Applications in Business," for the *MUG Quarterly*, vol. XX, no. 2.