# SNA Perspective On Internetworking

Volume 14, Number 8
August, 1993
ISSN 0270-7284

The single source, objective monthly newsletter covering internetworking with IBM's Systems Network Architecture

## New TCP/IP-Like Applications for APPC

We in the networking business tend to view networking architectures and protocols as market driving forces, but in reality it is the end users of networks and their applications that have reshaped enterprise networks. The networking protocols that have enjoyed success are those which have been tied to successful applications and application platforms.

Two examples of protocols that have achieved widespread use in enterprise networks are TCP/IP and SPX/IPX. In each case, their success has been driven mainly by buying decisions that were made by application developers and the end users of the network. These people were buying application solutions, not protocols!

One of the reasons that TCP/IP has been so successful is that it is almost always packaged with a suite of application programs. These applications include file transfer and electronic mail software. Equally important is the fact that these applications are standardized across all TCP/IP products. The TCP/IP standard for file transfers is the File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) is the standard electronic mail application, and TELNET is the standard interactive terminal

*(continued on page 2)*

## SNMP Management of SNA Systems

SNA networks have traditionally relied on SNA/Management Services (SNA/MS) to provide network management. While SNA/MS does a good job of managing the SNA portion of enterprise networks, most other types of networking equipment employ the Simple Network Management Protocol (SNMP). SNMP is the management protocol used in TCP/IP networks and it has become the industry standard for managing most networks components. Examples of networking products which rely almost entirely on SNMP management includes bridges, routers, intelligent hubs, and interfaces to networking services such as frame relay.

SNMP was developed in the late 1980s as a protocol for managing TCP/IP networks. While SNMP standards are still mainly targeted at the management of TCP/IP-com-

*(continued on page 12)*

(Continued from page 1)

protocol. Other generic applications are also included in most TCP/IP products.

This application standardization and packaging positions TCP/IP as an end user solution rather than as a protocol suite. When usersinstall TCP/IP on an application platform, they gain the ability to transfer files to virtually any other TCP/IP user. The same is true for other applications within the TCP/IP protocol suite.

## APPC Has Been a Tool, Not a Solution

Advanced Program-toProgram Communications (APPC) has not been packaged and marketed in the same way as TCP/IP. APPC has been positioned strictly as a program-to-program communications interface and protocol. APPC was not useable until someone bought or built an application to run on top of APPC. This, in itself, might not have been a major problem, but it led to a situation where there were no de facto standards set for generic applications such file transfer or e-mail.

The APPC market for these generic applications became fragmented resulting in a situation where APPC users did not gain the near universal interoperability enjoyed by TCP/IP users. Fortunately this situation is changing.

IBM has adopted the strategy of bundling basic applications with APPC products. As we will see, the applications even have names which are derived from those of their TCP/IP counterparts.

## The APPC Application Suite

Each of the applications in the APPC application suite performs functions which are required on virtually all networked systems. Some of the applications have been in use for over a year while others are just being made available.

An interesting characteristic of these applications is that they are very similar in both function and design to applications that exist within the TCP/IP protocol suite. As we mentioned earlier, one of the keys to the success of TCP/IP has been the availability of basic functions such as file transfer on virtually every system which implements the protocol suite. IBM is following a similar course which should greatly improve the usability of APPC and broaden its appeal.

Some of the programs are currently being shipped with IBM APPC products, while others are available on CompuServe and can be downloaded for use on APPC systems. IBM's intent is to make them widely available for both IBM and non-IBM APPC products. IBM's APPC CompuServe forum (GO APPC) is being used to expedite the distribution of the software even before it is formally packaged and shipped with APPC products.

The following applications have been available for some time:

- APING - a message echo application
- AREXEC - runs a command on a remote system
- ATELL - sends a message to a remote system

These applications ship with the OS/2 Communications Manager and with IBM's Networking Services/DOS product. In addition, there are available versions which run on MVS, VM, OS/400, and AIX platforms as well as under CICS.

IBM has just released the documentation for APPC File Transfer protocol (AFTP), a generalized file transfer protocol which is designed to be used across all APPC platforms. An OS/2 version of AFTP is now available on CompuServe and versions for DOS and VM will be available in the near future. IBM's objective is to get AFTP applications running on most IBM and non-IBM platforms by early in 1994.

The APPC Name Server is an application program which can be used to map a user name or nickname to an SNA logical unit name. In the past, users and application programs had to know the names of the LUs that they accessed across the network. The APPC Name Server allows users to use a meaningful name such as "Laser_Printer" to access network resources rather than an LU name such as XYZCORP.RS55033B. OS/2 and VM versions of the APPC Name Server will soon be available.

Each of these APPC applications has a TCP/IP counterpart and, as the following table shows, the names of the APPC applications have been derived from those of the corresponding TCP/IP applications.

| **APPC Application** | **TCP/IP Application** |
|---|---|
| APING | PING |
| AREXEC | REXEC |
| ATELL | TELL |
| AFTP | FTP |
| APPC Name Server | Domain Name Server |

The functions of the APPC applications are not exactly the same as their TCP/IP counterparts, but they are very similar and the user interfaces of some of the applications are very similar.

## APING

APING is a very simple APPC application which simply sends data packets to another computer and then reads the packets which were echoed back by the remote partner. APING is modeled after a similar application, called PING, which is part of the TCP/IP protocol suite. APING is actually a package of two APPC transaction programs. The client program which originates the transmissions is called APING and the server which echoes them back to the sender is called APINGD.

Why would anyone want to simply ping-pong a message across a network? There a couple of situations where APING can be very handy. The first is to perform a basic test of a newly installed APPC system. After an installation is completed, APING can be used to ensure that the system can communicate with other designated systems in the network.

In order to carry out these simple connectivity tests, APING can be started with the following command line entry:

```
APING  dest_name
```

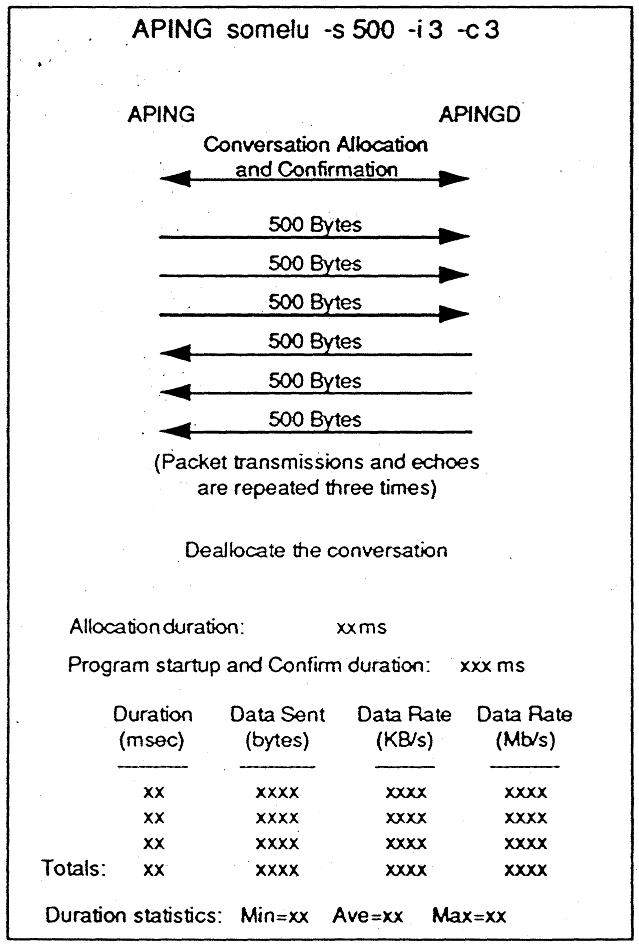
The destination name can either be an LU name or a CPI-C symbolic destination The CPI-C symbolic destinations are names which can be used by users of the Common Programming Interface for Communications (CPI-C). These names are mapped by CPI-C into LU names. When APING is executed with no optional parameters, it simply sends a 100 byte data packet to the designated LU and waits for the packet to be echoed back. It then perfroms this sequence of operations a second time. Successful completion means that the local system can establish a session and allocate a conversation with the remote LU. APING will also record the length of time required to allocate the conversation, and the amount of time required to start the transaction program. The amount of time required to send the data back and forth is also recorded and the effective data rates are calculated.

APING can be started using optional parameters which provide additional functionality. The following is a list of these parameters.

| | |
|---|---|
| -m mode_name | sets the mode name used for the session - default is #INTER |
| -t tp_name | name of the transaction program on partner system - default is APINGD |
| -s N | transmit packet size - default is 100 bytes, valid range 0 - 32,767 |
| -i | number of iterations - default is 2, valid range 0-32,767 |

| | |
|---|---|
| -c N | number of consecutive packets sent during each iteration - default is 1, valid range is 1-32,767 |
| -u userid | the user ID sent to the partner |
| -p password | the password sent to the partner |
| -n | forces APING to use no conversation-level security |
| -1 | send data from client to server with no echo |

Combinations of these options could be used to obtain a rough measure of the throughput that users are likely to experience when connecting to various partners. For example, figure 1 shows an APING start-up sequence and the data flows and performance measurements that result.


```
APING somelu -s 500 -i 3 -c 3
```
APING
APINGD
Conversation Allocation and Confirmation
500 Bytes
500 Bytes
500 Bytes
500 Bytes
500 Bytes
500 Bytes
(Packet transmissions and echoes are repeated three times)
Deallocate the conversation

```
Allocation duration:                 xx ms
Program startup and Confirm duration:    xxx ms

         Duration   Data Sent   Data Rate   Data Rate
          (msec)     (bytes)     (KB/s)      (Mb/s)
         -------    -------     -----       -------
            xx        xxxx        xxxx        xxxx
            xx        xxxx        xxxx        xxxx
            xx        xxxx        xxxx        xxxx
Totals:     xx        xxxx        xxxx        xxxx

Duration statistics:  Min=xx   Ave=xx   Max=xx
```

*Figure 1*

The command line start-up sequence specifies that a sequence of three 500 byte data blocks should be sent to APINGD. This sequence will be sent to APINGD three times. This results in the data flows shown in the middle of the diagram.

The timing and throughput statistics recorded by APING are shown at the bottom of the diagram. This information gives the user a general idea of how long it takes to allocate a conversation and start an application program. It also shows the kind of throughput that is being experienced.

Even though APING is a very simple application, it is useful as an installation verification tool and as a method for estimating a network's effective data rates to specific destinations.

## AREXEC

The AREXEC application allows a user on a client system to send an operating system command to a system acting as a AREXEC server. The AREXEC application is executed by the following command.

```
AREXEC  parameters  destination  command
```

AREXEC supports some of the same option parameters as we described under the APING application, including -m, -t, -u, -p, and -n. The destination can be either the name of the partner LU or a CPI-C symbolic destination name. The command field is simply the command to be executed on the remote system. The following is an example of an AREXEC command.

```
AREXEC destlu dir\mydir\subdir1
```

This command will cause the AREXEC client software, called AREXEC, to begin executing on the local system. The server transaction program, called AREXECD, will be started on the system designated as destlu. The directory display command will be sent to the server and executed. The results will then be sent back to the client system and displayed.

## ATELL

The ATELL application allows a user to send a message to another system in the network. The following is the format of the ATELL command.

```
ATELL parameters destination message
```

The parameters are the same as those used by the AREXEC command and their use is optional. The destination can be either the name of the partner LU or a CPI-C symbolic destination name. The message field is simply the text of the message to be sent. The following is an example of an ATELL command.

```
ATELL destlu Good morning!
```

This command starts the client application, called ATELL, on the local system. The server transaction program, called ATELLD, is started on the system designated as destlu which displays the good morning message.

## The APPC Name Server

End users and network services within SNA networks have traditionally been identified by unique logical unit names. This is true of both subarea SNA networks as well as APPN networks. In subarea SNA networks VTAM maps the LU names into addresses which are used to send data to specific LUs. APPN provides a directory service which is used to locate logical units. Note that APPN directory services does not return the address of the target LU. Instead, the name of the Control Point (CP) with which the LU is associated is returned, and then APPN's route selection services calculates an appropriate route to get to the LU.

In both APPN and subarea SNA networks the directory services assumes that the requester of the directory service knows the name of the LU that is to be located. There are several problems that result from exposing users directly to LU names. First, the eight character limitation on LU names results in the creation of names that are not very meaningful to users. The 17-character fully-qualified names are generally no more descriptive because the additional characters designate the network ID. Secondly, if users are allowed to access resources by aliases or nickname, their requests can be transparently redirected to different LU in the network. It is also useful to allow users to reference the names of generic services rather than specific LUs.

### User Names and Group Names

The APPC name server allows users to reference network resources using names that can be up to 64 bytes in length. This allows much more meaningful names to be created for resources. The name server also allows users to specify a group name to retrieve groups of related LU names. This makes it possible to assign names which are much more meaningful to the network users than the 17-byte fully-qualified logical unit (FQLU) Names which are used by SNA.

The name server actually allows users to define two categories of names. User Names map to a FQLU name and simply provide an easy-to-remember alternative to the use of FQLU names. The second type of name is the Group Name which allows one or more users to be referenced by a single name. The Name Server maintains a database which provides the mappings between the User Names and Group Names, and the FQLU Names used by SNA.

### Name Server Functions

The name server provides a set of functions which can be accessed by client systems using either a command line interface or an application programming interface. The following categories of functions are provided by the name server.

- Register user names in the server database
- Query the name server database
- Delete information from the database

### The Command Line Interface

The client system command line interface supports the following name server operations:

- NSREG - registers information in the server database
- NSQRY - queries the server database for user information
- NSQRYQ - queries group information
- NSDEL - deletes information from the server database

Each of these commands can include one or more parameters which are identified by the following flags:

- -F - Fully-qualified logical unit name
- -G - Group name
- -U - User name

### Registering User Names

Register commands add information to the name server database. The register command can be user to register a User Name with either a FQLU Name or a group name. For example, the following command registers a user named Fred with an FQLU Name:

```
NSREG  -U Fred  -F netid.fredslu
```

The following command registers Fred with the Group called Mkting:

```
NSREG  -U Fred  -G Mkting
```

### Querying The Name Server Database

The NSQRY command can be used to retrieve database records based on combinations of a user name, group name, and FQLU name. One or more database records may be retrieved in response to a query. For example, the following command will return the database record which was created in our previous registration example:

```
NSQRY  -U Fred
```

```
The result would be:  Fred   netid.fredslu  Mkting
```

We can also request a database record for every user in the Mkting group:

```
NSQRY -G Mkting
```

```
The result would be:  Susan netid. sueslu  Mkting

                      Fred   netid.fredslu  Mkting

                      Don    netid.donslu   Mkting
```

The name server database can also be queried by FQLU name as in the following example:

```
NSQRY  -F netid.fredslu
```

```
The result would be:  Fred    netid.fredslu Mkting

                      Adminnetid.fredslu Mkting
```

Note that more than one user name can be registered to a single FQLU name.

The group query command can be used to retrieve information regarding groups which have been registered with the name server. The following command retrieves the names of all of the groups which have been registered.

```
NSQRYG
```

```
The result would be:    Accounting
                        Manufacturing
                        Marketing
                        Sales
```

The NSDEL command is used to remove entries from the name server database. The following command removes all records which contain the user name Fred.

```
NSDEL  -U Fred
```

### Role of the APPC Name Server

The name server can be accessed by users through the command line interface as we have seen, or through an application programming interface. The command line interface is a tool that can be employed by the user to manually retrieve the FQLU names required by existing SNA applications. Note that even though the name server uses APPC for its client-server communications, the users and FQLU names represented in the database could be associated with any type of LU including dependent LUs such as LU 0, 1, 2, and 3.

New applications will be able to take advantage of the name server's application programming interface to transparently access the name server's database. This means that users of these types of applications will only have to deal with user names and need not ever be exposed to the FQLU names.

### APPC File Transfer Protocol

One of the most basic, and widely used, networking services is file transfer. In order to meet this almost universal requirement, IBM is providing the APPC File Transfer Protocol (AFTP). AFTP is a package of client-server applications which can be used to transfer either text or binary files between systems.

AFTP is packaged as a set of file transfer programs. The AFTP package is actually made up of the following three programs:

- AFTD - the server program
- ACOPY - a client program which is driven by a command line interface
- AFTP - a client program with an interactive application programming interface

All three programs can run on a single system. IBM recommends that all three applications be installed on most systems. This would allow most platforms to operate as both clients and servers, as required.

### Mapping Incompatible File Systems

The AFTP applications are designed to transfer files across dissimilar operating systems. This presents a challenge because operating systems each have their own directory structures and file naming conventions. AFTP uses the following common format to identify files:

/directory1/directory2/filename

Note that the designations of files and directories can either be in the native format used by the server's operating system or an AFTP standard notation can be used. When the AFTP standard, system-independent notation is used, it is translated into the native format when it reaches the server. The forward slashes which separate directories in our example represent the AFTP standard notation. If the server is a PC-DOS system, for example, the forward slashes would be converted to backslashes at the server.

In some cases the directory names will identify real directories, but they can be used to represent any method that an operating system might use to partition files within the file system. An example of this partitioning would be the identification of logical disk drives within the DOS and OS/2 operating systems.

As shown in figure 2 AFTP treats DOS and OS/2 logical disk drives as if they where directories rooted in a virtual root directory which is created by AFTP.

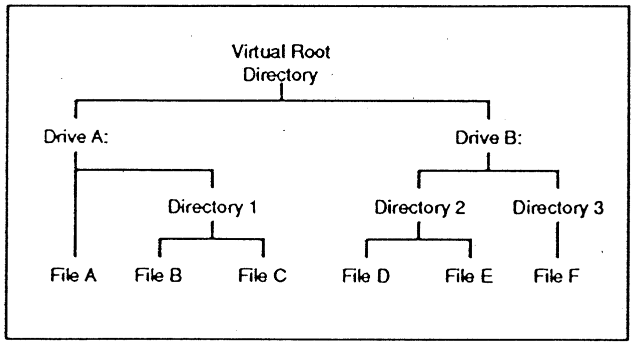


*Figure 2*

### Mapping File Names Across Operating Systems

Since operating systems have different naming conventions for files, AFTP allows users to specify how names are mapped as they are moved from one system to another. These mappings are defined within the AFTP Initialization File. The AFTP Initialization file is used by all three of the AFTP client and server application programs. Various types of verbs can be included in the initialization file, but the Map verb is used to control the mapping of file name formats across systems.

The initialization file may contain one or more Map verbs with the following syntax.

```
map    source_mask(mask)
       target_mask(mask)
       options(binary or ascii)
```

The source_mask field is used to select those file names that will be processed by this Map verb. The mask can be made up of any valid file name characters plus wildcard characters. The target_mask field specifies the changes, if any, that will be made to the file names which match the source_mask field. The options field specifies the type of file transfer, binary or ascii, that will be used to transfer files whose names match the source_mask field.

The following Map verb could be used to specify that files with an SRC suffix are to be transfered as ASCII files with no changes made to the file names.

```
map    source_mask(*.SRC)
       target_mask(*.*)
       options(ascii)
```

The following Map verb specifies that files with an EXE suffix are to be transfered as binary files and the suffix should be changed to PRG at the target system.

```
map    source_mask(*.EXE)
       target_mask(*.PRG)
       options(binary)
```

Initalization files containing Map verbs may exist on both the sending and receiving systems. File name mapping is always performed based on the Map verbs in the receiving system's initialization file. The binary/ascii transmission option is always selected by the Map verbs in the client's initialization file.

### AFTP Security

System administrators can control remote access to the files which reside on an AFTD server. Access to files is based on user IDs and passwords. The AFTD server itself does not process the user IDs and passwords because AFTP uses the conversation security features of APPC. AFTP also uses the security mechanisms of the server's operating system when they are available.

When an AFTD server is running on an operating system which does not provide file security based on user IDs, AFTP provides its own file access security. AFTP controls access to directories rather than individual files.

When AFTP provides file access security, the server's initialization file will contain Provide_Access verbs which specify the users who can access each directory along with the type of access which is granted. The general format of the Provide_Access verbs is:

```
provide_access   directory(directory name)
                 permissions(read and/or write)
                 users(user IDs)
```

The following Provide_Access verb would allow anyone who connects with the server to read all of the files contained within the \public directory and all of its subdirectories.

```
provide_access   directory(\public\)
                 permissions(read)
                 users(anonymous)
```

Note that AFTP allows the use of "anonymous" as a user ID. This follows the TCP/IP file transfer convention which is used to make certain files on a server available to anyone regardless of user ID.

The following Provide_Access verb will allow users "donc" and "stever" to read and write any of the files on the AFTD server.

```
provide_access    directory(*)
                  permissions(read write)
                  users(donc stever)
```

### AFTD Command Line Client

ACOPY is a client program which supports a very simple command line interface. The syntax of the ACOPY command line is:

```
ACOPY flags  source: source_file  destination:
                        target_file
```

flags - used to specify file transfer options such as text or binary, and to specify security options

source - LU name of the system that the file will be sent from

source_file - the name of the file to be sent

destination - LU name of the system that the file will be sent to

target_file - the name that will be assigned to the file at the destination system

The ACOPY program is limited to very simple file transfers. If features such as directory and file manipulation are required, the AFTP program and its interactive API must be used.

### AFTP Interactive Client

The AFTP interactive client program supports a range of file management functions in addition to transfering files. A key characteristic of the AFTP interactive interface is that it is very similar to that of TCP/IP's File Transfer Protocol (FTP) application. This will minimize retraining as users move between TCP/IP and APPC platforms.

### Starting an AFTP Connection

A connection between the AFTP client and the AFTD server must be opened before any file transfer or file management operations can take place. A typical connection start-up sequence would include the following interactions.

```
aftp> open mynet.serverlu
userid: myname
password: secret
Connected to mynet.serverlu
aftp>
```

### Setting File Transfer Attributes

A set of commands are provided to set attributes which will apply to all subsequent file transfer operations. The following is a summary of the available commands.

| | |
|---|---|
| ascii | assigns a data type of ascii to all subsequent file transfers |
| bell [on off] | controls bell signal which indicates that a transmission is under way |
| binary | assigns a data type of binary to all subsequent file transfers |
| date [new old] | controls the setting of the data attribute on transfered files. "new" indicates that the current date will be used while "old" indicates that the date on the original file will be used. |
| modename mode_name | sets the APPC modename used to transfer files |
| prompt [on off] | controls prompts which allow users to selectively get and put groups of files |
| status | requests a display of the current transfer attributes |
| system | requests a display of information describing the server computer |
| type | displays the current data type (ascii or binary) being used for transfers |

### Transfering Files

The actual file transfer operations are initiated by GET and PUT commands or by their respective aliases, SEND and RECEIVE.

| | |
|---|---|
| get remote_file [local_file] | copies one or more files from a remote server |
| put local_file [remote_file] | copies one or more files to a remote server |
| send local_file [remote_file] | an alias of the PUT command |
| receive remote_file [local_file] | an alias of the GET command |

If local_file name is not specified on GET and RECEIVE commands, the name of the file on the remote system is assigned. Likewise, if the remote_file names are not present on the PUT and SEND commands, the local_name of the file is assigned to the copy on the remote system.

### Directory and File Manipulation Commands

In addition to file transfer operations, AFTP implements a number of commands which can be used to manipulate files and to perform both local and remote directory operations. Again, these commands are patterned after those used used by TCP/IP's FTP.

Remote (Server) Operations:

| | |
|---|---|
| cd directory | changes the current working directory on the server |
| delete file_name | deletes a file on the server |
| dir [file_spec] | displays a listing of a directory on the server |
| ls [file_spec] | displays a condensed listing (only file and directory names) of a directory on the server |
| lsd [directory_spec] | lists only the subdirectory names of a directory on the server |
| mkdir directory_spec | creates a new directory on the server |
| pwd | displays the name of the current working directory on the server |
| rename old_filename new_filename | renames a file on the server and, optionally, moves it to a new directory |
| rmdir directory_spec | removes a directory from the server |

**Local (Client) Operations:**

| | |
|---|---|
| ! | allows users to issue miscellaneous commands on the local system |
| lcd [directory_spec] | changes the client's current directory |
| lpwd | displays the name of the client's current working directory |

### Terminating AFTP Operations

The AFTP interactive interface defines commands which terminate the AFTP session with a file server, and terminate the AFTP application.

| | |
|---|---|
| bye | an alias of the EXIT command |
| close | terminates the AFTP session with the server without terminating the AFTP application program |

| | |
|---|---|
| disconnect | an alias of the CLOSE command |
| exit | exits from the AFTP program |
| quit | an alias of the EXIT command |

### Sample AFTP Interaction

connect to the server
```
aftp>open uscorp.srvr01lu
userid:user1
password:password1
Connected to uscorp.srvr01lu
```

set current directory on server
```
aftp>cd  /c:/project
CD command successful
```

delete all files with "src" suffix
```
aftp>delete *.src
DELETE command successful
```

set local current directory
```
aftp>lcd /d:/working
Local directory now /d:/working
```

transfer files in ASCII format
```
aftp>type ascii
Using ascii mode to transfer files
```

transfer file - keep local name
```
aftp>put tester.src
Sending /d:/working/tester.src in ascii
9876 bytes sent in 1.633 seconds (6.06 Kbytes/s)
```

list the contents of the remote directory
```
aftp>ls
directory for /c:/project
TESTER.SRC
Directory listing complete
```

close the session to the server
```
aftp>close
Goodbye.
```

## Impact of The APPC Application Suite

These APPC applications are not very complex and they are not very sexy, but in our opinion they are very important to the success of APPC. While APPC, in partnership with APPN, has succeeded in providing SNA users with any-to-any connectivity, it has not provided much in the way of application interoperability. Application interoperability was always treated as a separate issue. This application suite will change that situation.

If IBM is successful in deploying this application suite across all APPC products it will open APPC to a new category of users. New users could buy into APPC with the knowledge that it could immediately be used to perform a wide range of useful functions.

## Bottom-Up Buying Decisions

Our experience indicates that most decisions to use APPC have been top-down decisions. APPC has been adopted and deployed by many of the corporate Information Services departments who have traditionally been SNA users. APPC has already achieved a good degree of success among these users, but many network buying decisions are now being made by workgroups and departments rather than by corporate management.

These users are not interested in assembling applications and protocols - they require turnkey solutions. The availability of this application suite could open up this fast growing market to APPC. ■

(Continued from page 1)

patible products, the use of SNMP has evolved to include a wide range of products that have little or nothing to do with TCP/IP. SNMP has become the industry standard for management of mixed protocol networks.

SNMP-managed systems and SNA-managed systems are destined to coexist in enterprise networks, so the problem of dealing with this mixed management environment must be addressed. One alternative that many network managers are now employing is the use of the SNA/MS managers, usually NetView, and one or more SNMP management platforms. This approach can be effective, but most network managers would prefer to use a single network management scheme to manage the entire network.

In order to allow network managers to use a single management protocol, IBM and other SNA product vendors are developing strategies for using SNMP to manage SNA devices. In this article we will review the SNMP management environment and show how it is being positioned to manage APPN systems.

## The SNMP Management Environment

Before we get into a discussion of how SNMP works, we will look at its basic design and how it compares to that of SNA/MS. Like most network management schemes, SNMP is based on a network model which is made up of two categories of devices - the elements which perform the management functions and those which are managed. The systems which perform the management functions are called (not surprisingly) managers and the managed systems are represented by software elements called agents. SNA/MS's counterpart of the SNMP manager is the focal point and the SNA/MS entry point would provide functions similar to that of the SNMP agents.

SNMP defines the protocols which allow the managers and agents to interact with one another in a consistent way. Figure 3 shows a network in which two managers are managing a collection of devices which include bridges, routers and other types of devices. Each of the managed devices communicates with its managers via an agent.

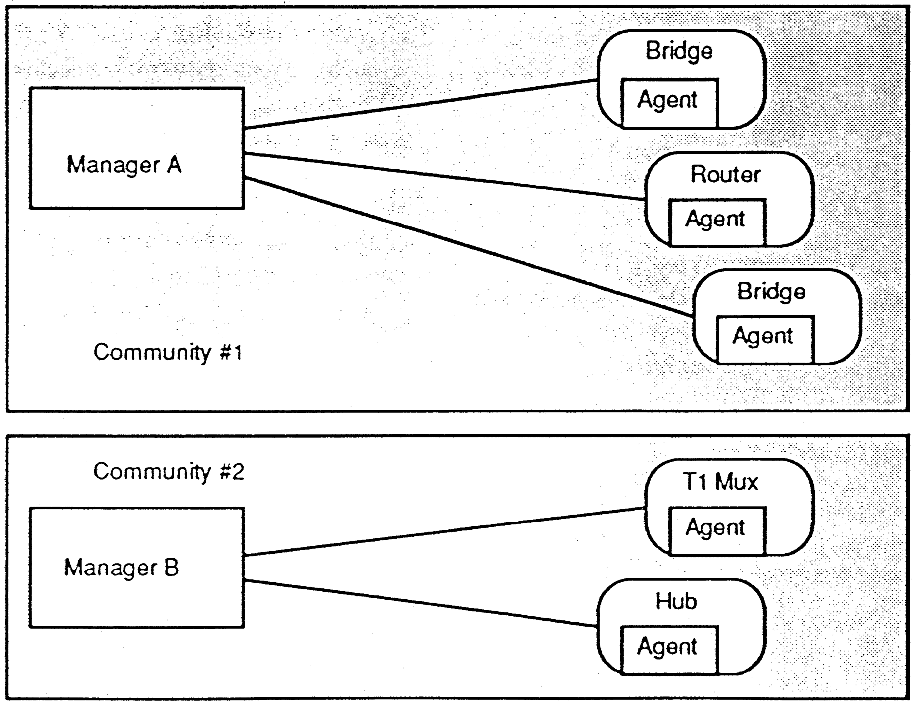


*Figure 3*

SNMP generally uses TCP/IP's User Datagram protocol (UDP) to transport the commands and responses between managers and agents. UPD is a connectionless protocol that allows managers and agents to exchange datagrams. It is also interesting to note that UDP does not provide reliable delivery. Unlike SNA/MS, which uses a reliable connection-oriented SNA session, SNMP commands and responses can be lost in transit. Each SNMP command or response is sent as an independent packet across an IP internet. Another weakness of SNMP is its lack of security. SNMP does not provide for the authentication of network management messages. SNMP messages do carry an identifier which is used to match up agents with their managers, but this is not really intended as a security feature. The security issue is being addressed in a new version of SNMP which is called SNMP II.

Before we examine how the SNMP network management model is actually implemented, we need to discuss two other important parts of the network management framework.

## *Using Objects to Represent the Real World*

The devices which are managed by SNMP are represented as abstract collections of objects. An object, as defined within SNMP, is a data element, such as a character string, integer, or counter, which represents some attribute of a managed system. For example, an object might be a counter that represents the number of data link level errors that have been detected by an Ethernet LAN adapter or a timer that indicates the time interval that has elapsed since a device was last initialized.

Devices within a network, such as bridges and routers, are represented as objects and collections of related objects. There are many different types of objects and each object is assigned a name and given a set of attributes. Product vendors can define their own objects to represent the state or characteristics of their own products.

SNMP is based on industry standards which define the structure of individual objects as well as standard collections of objects which represent various categories of networking devices. The rules for defining objects are contained within a document which is called the Structure of Management Information (SMI).

A collection of managed objects is referred to as a Management Information Base (MIB). A MIB can be thought of as a database of objects which represent a managed device. A MIB can consist of counters, addresses, timers, service descriptions, and other information used for network management. Each individual piece of information is a separately defined object in the MIB that can be accessed and/or modified.

Objects residing in a MIB can be accessed by managers. The SNMP standard does not define how a MIB is actually implemented in a real system. Some vendors may implement a MIB as a sequential file while others may implement it as a relational database. SNMP commands refer to objects that reside in a MIB using a unique object identifier for each object.

In order to allow managed objects to be protocol-independent, it was decided to split apart how objects are defined from the actual objects themselves. This led to the SMI and MIB recommendations discussed earlier. Let' s look at these more closely.

## Structure of Management Information

The SMI recommendation specifies that managed objects are described using the Abstract Syntax Notation One (ASN.1) which is an OSI standard. ASN.1 is a data description language that allows objects to be defined in a machine-independent manner. SMI describes the rules for defining and naming objects that reside in aMIB. It also sets rules for defining different object types that may reside in a MIB.

The SMI includes the definition of an ASN. 1 macro that is used to describe each managed object. Each object is identified with a name. The definition for each managed object also includes the data type which models the object, the level of access supported (e.g., read-only, write-only, read-write, and not-accessible) and requirements for implementing this object (e.g, mandatory or optional implementation or obsolete).

The object-type macro used to define objects includes the following components:

Name - the textual name of the object

Type - the type of variable which is associated with the object such as a timer, a counter, a text string, or an integer

Description - a description of the object

Access - the type of access which is allowed such as read-only or read/write

## Management Information Base (MIB)

Each of the individual objects which are defined within the rules of SMI represent a single piece of management information such as a counter, a timer, or a text string.Real networking devices exhibit many characteristics which must be monitored and/or altered for network management purposes. SNMP network management isbased on monitoring and manipulating objects within a MIB which represent actual device characteristics.

In order to provide a standardized management environment across products from multiple vendors, an industry standard SNMP MIB has been developed. The current version of the industry standard MIB is called MIB-II. The basic MIB-II definition specifies a generic collection of objects which pertain to the management of TCP/IP products. MIB-II defines objects which fall into several major categories. The following groups contain general systems information:

- System Group - general system information including system name and location
- Interfaces Group - includes a set of entries describing each LAN or WAN communications port on the device and statistics relating to that interface

Each of the following groups include statistics and parameters which relate to a specific protocol used within the TCP/IP protocol suite:

- IP Group - Internet Protocol
- ICMP Group -Internet Control Message Protocol
- TCP Group - Transmission Control Protocol
- UDP Group - User Datagram Protocol
- SNMP Group - contains statistics that describe the use of SNMP management protocols

Several other miscellaneous groups also exist.

MIB-II defines the basic collection of SNMP management objects and is obviously focused on the management of TCP/IP-based systems. This doesn't mean that SNMP is only capable of managing pure TCP/IP systems. The management requirements of other types of products can be addressed by defining extensions to the basic MIB which are targeted at managing different types of systems. Later in this article we will look at a MIB which has been recently made available to manage SNA APPN systems.

A series of extensions to MIB-II have been defined by the Internet community to manage a variety of systems and protocols. These include:

- Token-Ring and Ethernet LANs
- Tl wide area links and frame relay services
- DECnet Phase IV
- Transparent and source routing bridges

In addition to these industry standard MIB extensions, vendors and users of networking products can add their own product or installation-specific extensions to the MIB.

MIBs employ a tree structure to organize their management objects. This tree structure also creates a naming scheme that is used to access the individual objects. The naming scheme also ensures that extensions to the MIB can be made by independent organizations while protecting the integrity of the basic MIB.

The tree structure used by SNMP MIBs is actually part of a larger industry standard naming tree that has been defined and maintained by the the International Standards Organization (ISO). All of the information, including that contained in SNMP MIBs, is uniquely named using ASN.1 constructs refered to as object identifiers. These object identifiers are combined to represent the unique path between the root of the tree and the object being identified. Figure 4 shows some of the branches of the naming tree which are used to support SNMP

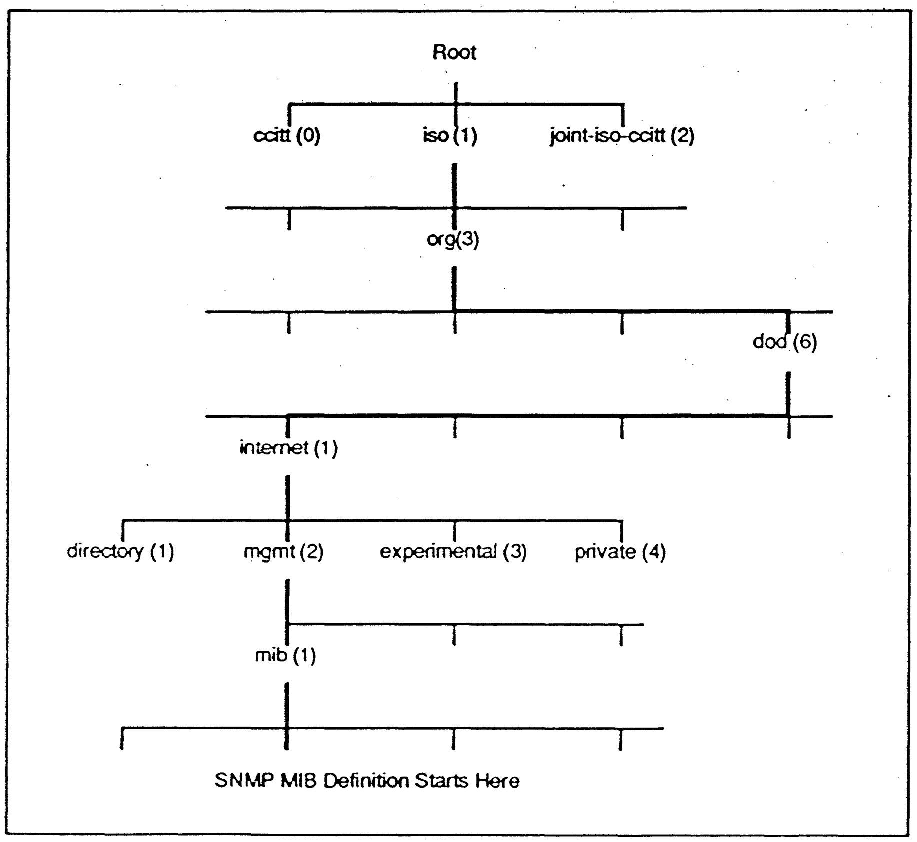


*Figure 4*

Note that each node of the tree is identified by both a name and a number. The objects defined at each level are assigned a unique number and a unique name. The numbers only need to be unique among the identifiers within a single level. Globally unique identifiers are created by concatenating the numbers of the objects between the root and the object which is being named. Names, on the other hand, must be globally unique.

The following concatenated number notation identifies the beginning of the SNMP MIB.

1.5.6. 1.2. 1

This dotted decimal notation is frequently used by itself or in combination with names of the corresponding objects as in this example:

iso(1) org(3) dod(6) internet(1) mgmt(2) mib(1)

Responsibility for the assignment of object identifiers is delegated to organizations which are responsible for the assignment of unique identifiers within their subtrees. For example, the Internet Engineering Task Force (IETF) is responsible for assigning identifiers within its subtree which includes the SNMP MIB definition. The IETF can, in turn, delegate responsibility for its subtrees to other organizations. This is how vendor specific MIBs are created.

The standard MIB is defined under the branch called mgmt(2) while vendors can define their MIBs under the branch of the tree which is labeled private(4). The IETF assigns a unique number to each vendor which gives each vendor their own subtree for defining private MIBs. IBM, for example, has its own subtree for product-specific MIBs which currently includes the 6611 multiprotocol router and the 3172 controller.

Now that we have described how objects are defined and used, we can look at how SNMP actually operates. SNMP defines a set of protocols which are used for communications between SNMP managers and the systems which they manage.

## Interactions Between Managers and Agents

As its name indicates, SNMP was designed to be simple. This means that it was designed with a minimal amount of overhead and can be relatively easily implemented on systems without requiring huge amounts of memory. Following the network management model described earlier, SNMP is based on managers, agents, and MIBs. Managers and agents interact using SNMP protocols in order to carry out network management services.

A manager is resident in a Network Management Station. Manager functions are implemented in software executing on a Network Management Station. A manager sends SNMP commands to agents in order to obtain network management information from those agents or to modify variables maintained by agents.

An SNMP agent resides in a system or device that is to be managed. Agent functions are implemented in software that resides in systems such as hosts, gateways, or terminal servers. An agent responds to commands issued by an SNMP manager. An agent either sends back information requested by a manager or alters variables as directed by a manager.

An agent has access to objects in a MIB. Agents can inspect variables or alter variables in a MIB. Each agent has its own view of the MIB. The view can include the standard MIB objects as well as other extensions not supported by all agents.

Not all systems in a network will support SNMP directly. A proxy agent is used to provide network management support for a system or device that does not implement SNMP agent functions. In effect, a proxy agent acts as a protocol converter for non-SNMP systems. For example, a non-SNMP device can be managed by an SNMP manager via a proxy agent. The proxy agent acts as a standard SNMP agent in communicating with the manager and communicates with the non-SNMP device in its native mode. The role of the proxy agent in SNMP is roughly analogous to that of the service point in networks that are managed using SNA/MS.

## SNMP Commands

The SNMP protocol is based on a set of commands that a manager can issue to an agent and the response that an agent can return. There is also a single command that can be sent from an agent to a manager. The following is the complete list of SNNP commands:

- GetRequest - sent from a manager to an agent to retrieve the value of a specific object variable
- GetNextRequest - sent from the manager to an agent to retrieve the next variable in the hierarchical MIB structure
- GetResponse - sent from an agent to a manager to deliver the contents of a variable which was requested by either a Get, a Get Next, or a Set request
- SetRequest - sent from a manager to an agent to modify the contents of a variable in a MIB
- Trap - an unsolicited message which is sent from an agent to a manager to indicate that some event or exception condition has ocurred

One of the reasons that SNMP is called "simple" is because there are just a few commands used. The management protocol can be as simple as using read and write type operations directed to variables within a managed node. However, although the set of SNMP commands is simple, the "back-end" processing of these commands can be complex.

The interactions between an SNMP manager and its agents using these commands demonstrates some of the major design differences between SNMP and IBM's SNA/MS management environment.

## SNMP - SNA/MS Design Differences

One of the most significant design differences between SNMP and SNA/MS is in the way that management intelligence is distributed between the managing systems and the managed systems. In the SNMP environment that we've described, the managed systems and their agents are very simple entities that do little other than to respond to specific requests for information and actions. The actions of the agents are almost totally driven by their managers. The software in the managed systems is also responsible for collecting the information which resides in the management objects in the MIB.

The entry points and services points which represent the managed systems in an SNA/MS environment differ from the SNMP model in that they implement more of the management process in their software. In an SNA/MS environment managed systems actually contain a considerable amount of information about how they should be managed. This results in a management environment where the management intelligence is more distributed than in SNMP management. The interactions shown in the following diagram demonstrate the impact of these differing design philosophies on the network management process.

The SNA/MS approach which is shown at the top of the figure 5 differs from the interactions which are typically used in an SNMP management environ-

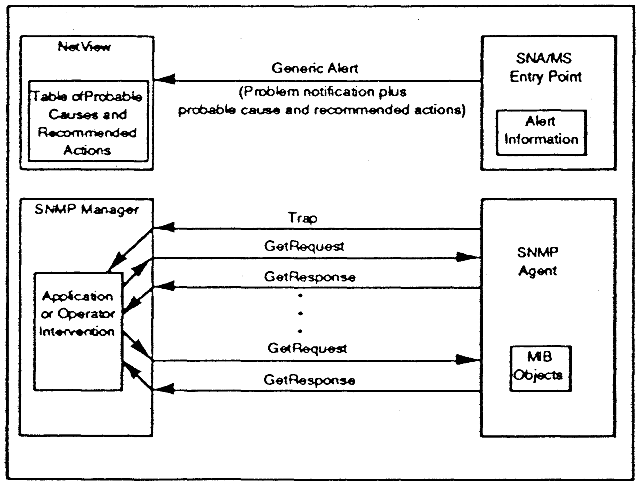

NetView
Table of Probable Causes and Recommended Actions
Generic Alert
(Problem notification plus probable cause and recommended actions)
SNA/MS Entry Point
Alert Information
SNMP Manager
Application or Operator Intervention
Trap
GetRequest
GetResponse
GetRequest
GetResponse
SNMP Agent
MIB Objects

Figure 5

ment as shown at the bottom of the diagram.

The first difference between the two approaches is the fact that SNA/MS relies much more heavily on the unsolicited reporting of exception conditions by the managed systems. These alerts are generated by entry point or service point software in the managed systems. If, for example, the operations staff wants to be notified when the number of data link errors exceeds a certain threshold, that threshold value is set in the managed system which then monitors the appropriate error counters and automatically generates an alert when the threshold is exceeded.

Software in the managed systems is capable of setting thresholds that are used to determine when alerts should be generated and also to generate the alert if the threshold is exceeded. This is a good example of the added management intelligence that resides in SNA/MS managed systems.

SNMP, on the other hand, relies much more heavily on explicit requests for status information from the manager. SNMP generally makes use of a technique which is called trap-directed polling. Trap-directed polling is a hybrid technique in which an unsolicited message, called a Trap in SNMP, is sent to the manager to indicate that an exception condition has ocurred.

So far, this is similar to the SNA/MS technique in which an alert was generated. The difference is that the SNMP Trap, unlike a generic alert, simply signals the manager that an exception condition has ocurred and does not attempt to deliver all relevant information back to the manager. After the manager receives the trap it sends out one or more GetRequest or GetNextRequest commands to solicit any additional information needed to resolve the problem.

Let's assume that we again want to notify the network operations staff when excessive errors ocurr on a given data link. The agent software is configured with the information that describes the error threshold at which operator notification is required.

In an SNMP environment the manager software is responsible for performing the monitoring function. The error threshold is set in the manager software rather than in the managed system. The manager must periodically send a request to the SNMP agent in the managed system to retrieve the contents of the MIB object which is the error counter for the data link. The software in the manager then compares the counter to the threshold value which will trigger a notification to the operator. The agent sends a trap to the managers whenever the threshold is exceeded. The trap only indicates that the event has ocurred and does not contain a complete description of the problem. When the trap is received by the manager, the operator or software within the manager will determine what additional descriptive information is required and then that information will be solicited from the agent by sending GetRequest or GetNextRequest commands.

## Using SNMP to Manage SNA Networks

As we mentioned at the beginning of this article, there is an industry trend toward the use of SNMP to manage almost every element of enterprise networks. This includes SNA-compatible systems. What steps must be taken to manage SNA systems using SNMP?

First, a TCP/IP protocol stack must be available to handle communications between the agents representing the SNA systems and the SNMP management platforms. Secondly, a MIB must be available which defines all of the objects required to manage an SNA system. And finally, a management application must be developed to run on the management platform. This application would manipulate the objects within the MIB and interact with the network operator, usually through a graphical user interface. It is possible to allow the network operator to simply issue direct SNMP commands to manipulate the objects, but as a practical matter a management application will almost always be used.

To wrap-up our discussion let's now look at a MIB that has been developed to manage APPN systems.

## The APPN MIB

A MIB has been defined so that APPN nodes can be managed by SNMP managers. The overall structure of this MIB is shown in figure 6. Defined below the APPN node group are seven subtrees, each defining a set of characteristics which can be monitored or controlled by an SNMP manager. Some of these subtrees, which apply to elements which ocur more than once on APPN nodes, are defined as arrays of objects. For example, there will be a set of link station information defined for each communications port on an APPN node.

The following is a sample of a few of the objects which are defined for each communications port on an APPN node. It also shows the notation which is used to describe the objects within a MIB.

```
ibmappnNodePortName  OBJECT-TYPE
       SYNTAX                 INTEGER {
                                 inactive(1),
                                 pendactive(2),
                                 active(3),
                                 pendinact(4)
                                 }

       ACCESS          read-only
       STATUS          mandatory

       DESCRIPTION
                 "Indicates the current state of
                 this port"
```

The label ibmappnNodePortName is the name that has been assigned to this object which describes the current state of a communications port on an APPN node. The port status is represented by an integer variable where a value of 1 indicates that the port is inactive, 2 indicates that an activation is pending, etc. This variable has read-only access which means that it cannot be set by the SNMP manager. The mandatory status indicates that this is a required MIB variable, others types of objects may be optional. And, finally, there is a verbal description of the object.

The following examples from the APPN MIB show the objects which indicate the type of data links attached to the communications ports.

```
ibmappnNodePortDlcType    OBJECT-TYPE
                SYNTAX        INTEGER {
                                 other(1),
                                 sdlc(2),
                                 dls(3),
                                 socket(4),
                                 ethernet(5),
                                 tokenring(6)
                                 }
                ACCESS        read-only
                STATUS        mandatory
                DESCRIPTION
                        "The type of DLC inter-
                        face, distinguished
                        according to the protocol
                        immediately 'below' this
                        layer."
```

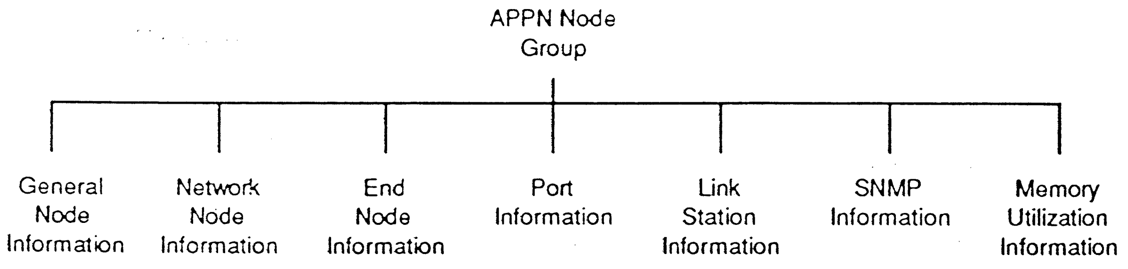


*Figure 6*

```
ibmappnNodePortPortType   OBJECT-TYPE
            SYNTAX        INTEGER {
                            leased(1),
                            switched(2),
                     sharedAccessFacilities(3)
                            }
            ACCESS        read-only
            STATUS        mandatory
            DESCRIPTION
                     "Identifies the type of line
                     used by this port."
```

In addition to the APPN MIB that we have described here, we are also likely to see MIBs which will be used to manage other SNA devices such as the 3174 controller. ■

## Summary

We expect to see IBM and other vendors implement SNMP management for their SNA products in the near future. One of the early implementations from IBM is SNMP management of the APPN Network Node support on the 6611 multiprotocol router.

While SNA/MS management is not going to go away anytime soon, SNMP is certainly the fastest growing area of network management today. If you are involved in SNA network management, don't overlook SNMP. It may become as important as SNA/MS for managing SNA-compatible systems. ■

Copyright © 1993 The Saratoga Group, all rights reserved. Reproduction is prohibited. • Subscription rates: U.S. - one year $395, two years $595. International - one year $445, two years $695 • *SNA Perspective* is published monthly by The Saratoga Group, 12930 Saratoga Avenue, Suite A-1, Saratoga, CA 95070 • Telephone (408) 446-9115 • Fax (408) 446-9134 • Managing Editor: Donald H. Czubek • Associate Editor: Stephen J. Randesi • Circulation Coordinator: Karen Stangel • Contributor: Dr. John R. Pickens • Typesetting and illustration: Three Marketeers Advertising, Inc. • The information and opinions within are based on the best information available, but completeness and accuracy cannot be guaranteed.