Network and system automation and remote system operation

by B. W. Irlbeck

The rapid growth in the size and complexity of today's information system networks highlights the importance of automation and remote system operation in managing these networks. Version 2 Release 2 of the NetView® program provides improved facilities in these two areas to assist enterprises in consolidating their operations staff in central locations and managing their information systems more efficiently and reliably. This paper describes the Systems Network Architecture (SNA) framework for remote system operation and the NetView remote operations platform, including the LU 6.2 data transport mechanisms and related management services applications. Extensions to the NetView automation platform that permit automation based directly on the receipt of SNA alerts or other structured data are described. In addition, enhancements that improve the performance, usability, and functional capabilities of the NetView automation table are discussed.

Throughout the past several years, information systems and networks have become more complex at an ever-increasing rate. A near explosion has taken place in the number and variety of systems and devices in today's networks. Further complicating the task of operating and managing these networks is the vast array of vendors producing the systems and devices that make up the network components.

At the same time, information networks have become ever more critical to the daily operation of today's businesses. System outages or slowdowns cost vast amounts in lost revenues and business opportunities. Business needs demand service levels with higher and higher performance and availability.

Along with the increasing size and complexity of the networks and the necessity of higher levels of service, the economics of today's marketplace have produced demands for greater efficiency in providing information system services. Businesses cannot afford to enlarge their staffs to operate their information networks. This constraint has led many businesses to concentrate their key operations personnel at a central location or a few regional sites.

To meet business objectives and cope with the pressures of all of these concurrent demands, automation of system and network operations is essential. Similarly, the ability to control remote systems operating in unattended mode is crucial in containing the cost of personnel and resources needed to operate the information network.

Remote operations. Recognizing that the proliferation of multivendor equipment mandates that remote operations be done according to specified standards, IBM put together a team of hardware engineers, software designers, and computer architects from several development laboratories. This team developed an architecture and de-

signed a way to implement automated and unattended operation of remote systems. The features of the solution include:

- An architecture for the basic commands and responses needed to operate remote systems
- An architecture for transporting the remote operations data over logical unit (LU) 6.2 sessions
- An architecture for defining and maintaining focal-point systems for operations management, alerts, and other categories of data (an alert is an error message in Systems Network Architecture that is sent to the system services control point at the host system)
- Applications in the central controlling system to format the commands to send to the remote system and to recognize and handle responses and error conditions
- Applications in the remote systems to recognize and act upon the commands and to format and send responses
- Automation of routine operations tasks and responses to error conditions at the system where they are detected, and suppression of notifications except when intervention is required

Network and system automation. To meet the need for more and better automation facilities and solutions, IBM is building on the automation capabilities provided in the NetView* program[1] and the various products that exploit the NetView automation platform. The NetView automation table previously provided the capability for text messages received from subsystems and applications to be compared against a number of criteria. Automation actions taken as a result of matching the criteria could include executing command lists or programs and displaying, suppressing, or logging the message.

Expanded NetView automation table capabilities provide ways to improve the performance, usability, and flexibility of the table for automating text messages. Because alerts and other data received in structured vector formats are very important in managing information system networks, extensions have been made to the automation table language to enable the automation of alerts and other structured data in their vector formats. The automation table can also be used to control the logging and display of alert information by the hardware monitor.

The remainder of this paper describes in more detail the remote operations architectures and the NetView implementations, including the LU 6.2 transports and related applications, and the automation platform enhancements provided in NetView Version 2 Release 2.

Remote operations platform

In enabling an operator or an automation program to remotely control the systems within a network, systems programmers were previously faced with many complexities and limitations.

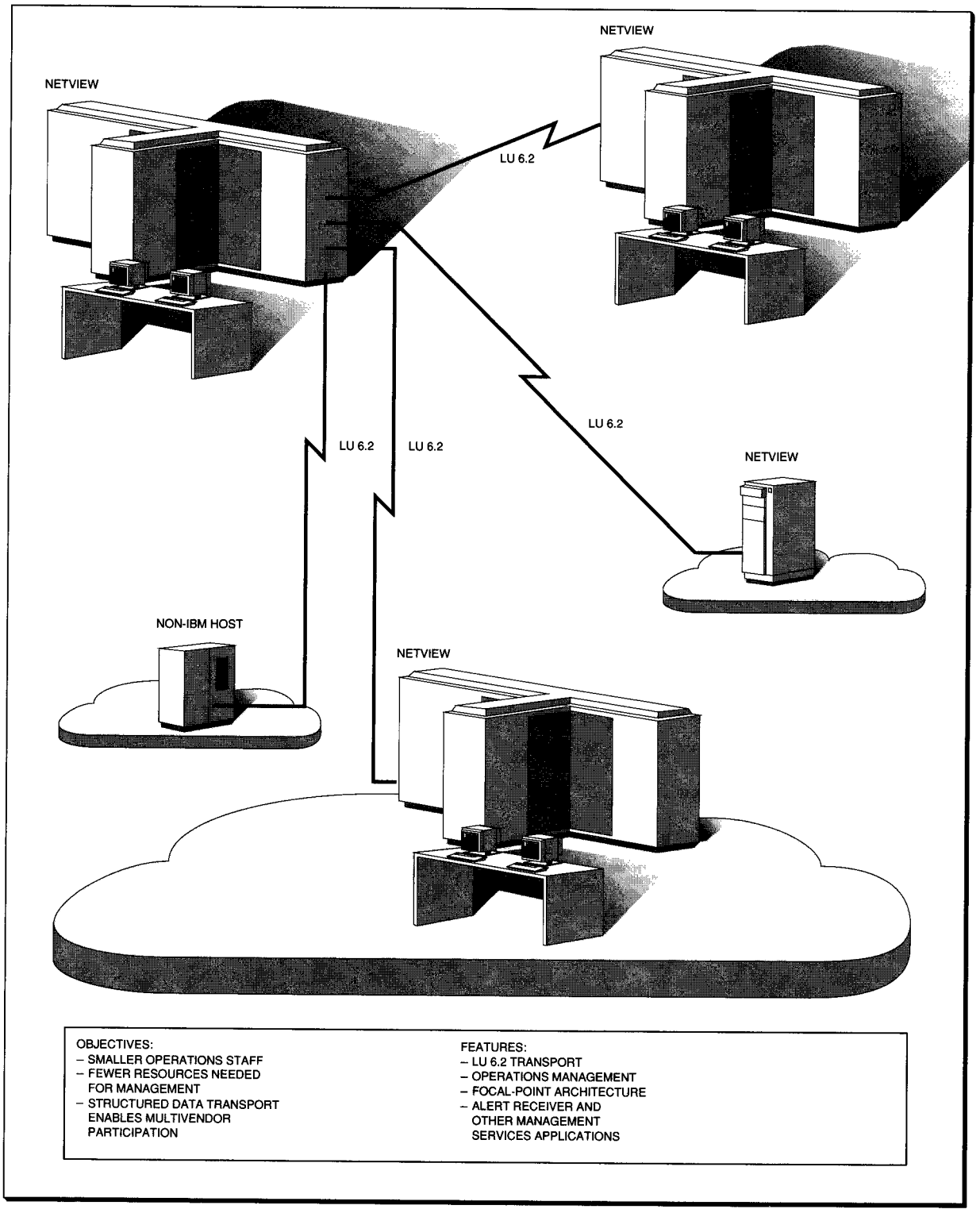
The controlling host system was required to have a predefined relationship with the remote system known as a *system services control point* (SSCP) *to physical unit* (PU) relationship. In the SSCP-PU relationship, the SSCP activates the SSCP-PU session. The SSCP-PU session is the mechanism for sending commands from an operator at the controlling system to the remote system and for returning responses and data from the remote system to the operator. This session also provides the means for transporting unsolicited management services data, such as alerts about error conditions, from the remote system to the SSCP. The exclusive use of SSCP-PU sessions for transporting management services data significantly limits the network configurations for remotely operated systems.

Different systems from different vendors all have their own syntax for the commands needed to activate them. Systems programmers must also learn the unique commands to set the system clock for each different type of system in order to synchronize events in the network and execute commands in each system at desired times. Likewise, canceling commands and deactivating systems require system-unique commands.

Because of the restrictions on the size of data records and on the types of data that can be transported on SSCP-PU sessions, many applications require their own type of session for transporting data between nodes. This requirement has led to a proliferation of sessions that must be managed and maintained.

The architectures and applications selected for the remote operations solution were developed to reduce these complexities and to provide the flexibility needed to extend remote operations capabilities to many more configurations (Figure 1).

Figure 1 Remote operations from NetView



Management services transport and applications. We now discuss the transport technique and the support provided for applications.

Management services transport. The first piece of the remote operations solution is the architecture for transferring data between a controlling host and the remotely operated systems. The architecture for Systems Network Architecture (SNA) Management Services (MS) has been extended[2] to define a transport technique for management services data using LU 6.2 sessions rather than SSCP-PU sessions. This technique is called *multiple-domain support* or sometimes *management services transport*, abbreviated as MS transport. NetView has chosen the latter term to describe its implementation of this architecture.

The MS transport uses LU 6.2 sessions to maximize the flexibility of configurations that can be remotely managed, thus avoiding the restrictions imposed when using SSCP-PU sessions. The use of brief conversations over shared sessions minimizes the overall network impact for transporting management services data. It allows the MS transport to handle communication with many remote nodes concurrently. In addition, the MS transport requires confirmation for each send to guarantee the reliable delivery of the application program data.

These specifications make the MS transport especially well-suited for use by management services applications. Examples of tasks performed more effectively are:

- Operations management applications that send commands and receive responses in order to operate remote systems
- Focal-point applications that establish focal-point relationships with entry-point applications
- Problem management applications that send and receive alerts

The MS transport architecture specifies:

- The formats for requests or unsolicited message units, replies to requests that require them, and error messages
- The method for correlating requests and replies
- The transaction programs to send and receive data
- The log mode to be used in establishing the LU 6.2 sessions
- The LU 6.2 protocols to be used when sending and receiving data

These specifications provide a highly reliable transport for management services data with guaranteed error notification in case of failure to deliver the data to the partner application in the remote node. They also ensure that management services applications using the implementation of the MS transport architecture by any product can communicate with management services applications using the implementation of the MS transport by any other product. No coordination of LU 6.2 or session protocols is necessary because they are completely specified in the architecture.

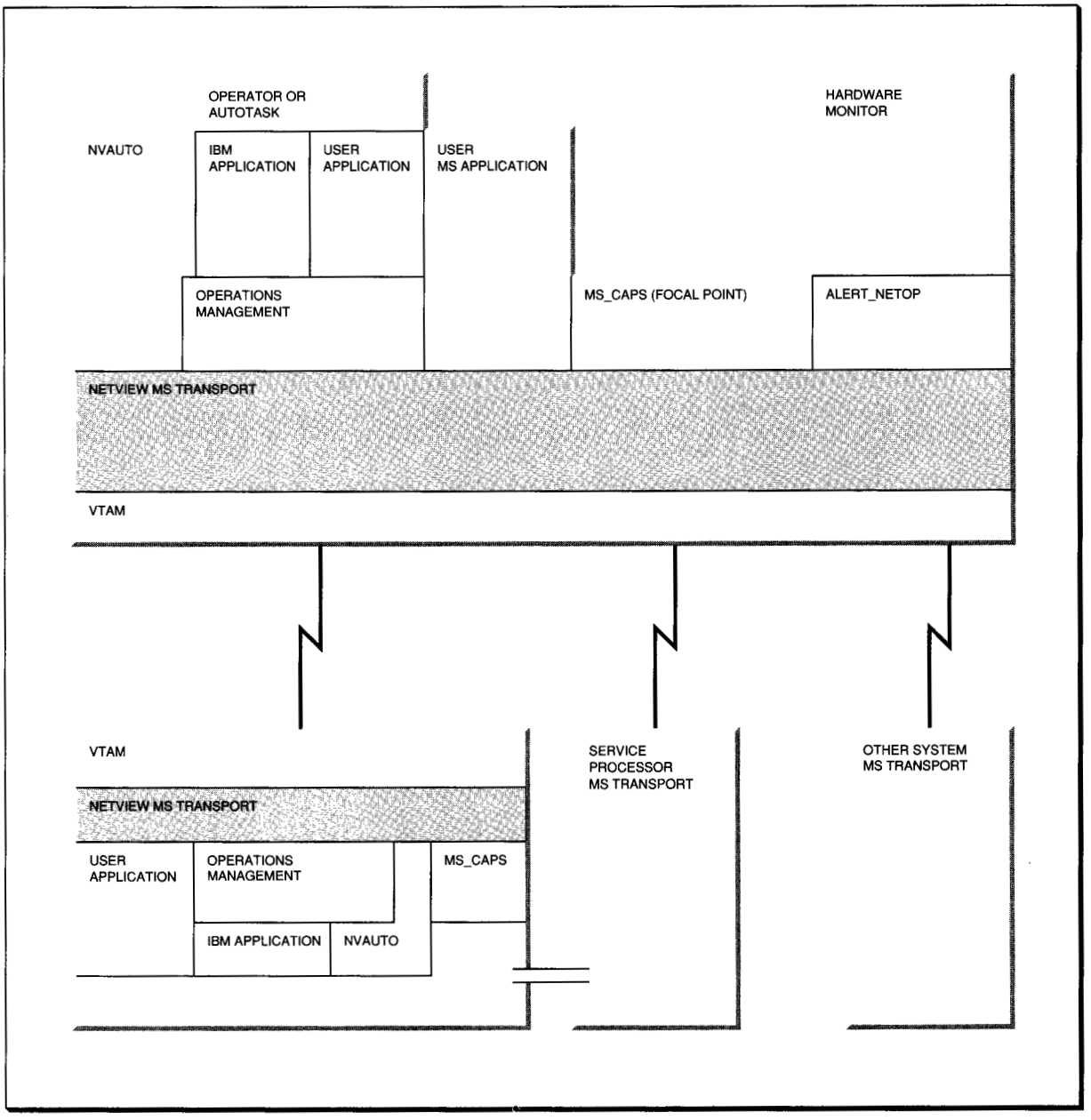
The NetView MS transport provides a high-level application programming interface (API) for management services applications running in NetView. An application registers with the MS transport, supplying its name and the name of a command to process data sent to it. The application can then send data in structured formats to a partner application in the same or a remote system and receive data in return. Data can be sent or received using PL/I or C service routines or assembler macros.[3]

If the partner application also runs in a NetView node, it can use the same high-level API. If the partner application runs in a node that does not contain NetView or another implementation of the MS transport such as that provided by Networking Services/2 with Operating System/2* (OS/2*) Extended Edition, the specifications for the architecture that must be implemented in the non-NetView node to ensure communication with the NetView MS transport are contained in the *Systems Network Architecture Management Services Reference*.[4]

Figure 2 shows a diagram of the NetView MS transport and some of the management services applications that make use of it. Additional applications, such as one to forward performance statistics to a central location, can use the MS transport and avoid the proliferation of special-purpose sessions. The MS applications supplied by NetView are described in the following subsections.

Operations management. The development of an architecture for remote system operation stan-

Figure 2 NetView management services transport and applications



dardizes the commands and procedures required to operate systems in a network from a central location. Systems programmers can provide remote operations applications using fewer system-specific command and response formats.

The SNA operations management architecture[4] specifies the formats for the requests and corresponding reports and replies needed to supply the following five major functions of operations management:

- The activation of systems includes such functions as power-on and initial microcode load (IML).
- Being able to set a remote system clock allows

a central site to synchronize the clocks in a network and to indicate offsets from different timing sources.
- The architecture allows a single, specific command to be canceled or all commands meeting a wide variety of criteria to be canceled.
- Deactivation performs the opposite of activation, reducing the target system from being a communicating entity to being logically inoperative.
- The ability to transport system-specific commands and responses in a structured and well-defined format is also required and is provided by the architecture.

This architecture provides a major step toward unattended operation of remote systems.

The NetView operations management application is a management services application that runs on top of the MS transport. Operations management acts as a second-level router. Second-level applications can register with operations management and are then referred to as *operations-management-served applications*.

The NetView operations management support allows IBM-supplied and user-written operations-management-served applications to send the operations management commands specified in the architecture to remote systems for execution and to receive replies and reports from the target systems. In this way, the system can be activated or deactivated by command, the system clock can be set, and other product-specific commands can be sent to operate the system remotely. The same operator or automation task within NetView that issues the initial command to activate the remote system can receive all of the subsequent replies and reports about that system. This permits the operation of multiple remote systems to be split up between human or automated operators as desired.

When an enterprise also takes advantage of the NetView automation capabilities, such as the improvements to the automation table for processing SNA structured data, it can achieve automated, unattended operations.

In an entry-point node, operations management, in cooperation with the focal-point support described next, allows a served application to be informed of the identity of the focal point for unsolicited operations management data. Having this information permits a served application in an entry point that is manually activated to notify a served application in the focal-point node to take over and begin operating the node remotely. It also allows the served application at the entry point to know when the focal-point node changes and data should be sent to a new focal point.

The interfaces provided by the NetView operations management support for registering, sending data, and receiving data are similar to those of the MS transport. They are provided as PL/I or C service routines or assembler macros.

Focal-point support. The growing need to be able to centralize important data of various categories has highlighted the need for a focal-point architecture. Customers have expressed the need for different focal points for different categories of data. For example, they may want a different focal point for accounting data than for alerts about error conditions in hardware or software. In addition, they need the capability to maintain continuous availability of services by switching to a backup focal point when the primary focal-point system fails or is recycled for periodic maintenance.

These needs are met by implementing the focal-point architecture defined in the *Systems Network Architecture Management Services Reference*.[4] This architecture makes use of the MS transport to provide the flexibility needed to support multiple configurations and remove the restrictions inherent in predefined SSCP-PU relationships.

NetView has implemented a subset of the SNA focal-point architecture that allows NetView to be a focal point for alerts, operations management, and user-defined focal-point categories. NetView has also implemented the architecture to be an entry point for operations management and user-defined categories.

Primary and backup focal points for operations management can be defined at system initialization time or later by command. The FOCALPT command allows entry-point to focal-point relationships to be defined, changed, or established from both focal-point and entry-point nodes. The FOCALPT command can be used for the NetView-unique alert focal-pointing as well as for focal-pointing that implements the focal-point architec-

ture for alerts, operations management, and user-defined categories.

The NetView focal-point application in an entry point keeps track of the current active focal point for each category and informs local applications that are registered as being "interested" in that category whenever a change occurs. The focal-point application automatically detects the loss of an active focal point. If a remote primary focal-point is lost, the focal-point application sets a timer to try later to reacquire it. Also, if a backup focal point is known, the focal-point application attempts to acquire it as the current active focal point. The focal-point application maintains current focal-point information in a VSAM (virtual storage access method) database so that the last known focal point for each category can be reacquired if the entry-point NetView is recycled.

The exchange of data that establishes an entry-point to focal-point relationship between two nodes for a specified category of data is referred to as an exchange of *MS capabilities*, and the name specified in the architecture for the application that handles the MS capabilities data is *MS_CAPS*.[2] This name is used in Figure 2.

LU 6.2 alert receiver (ALERT_NETOP). Applications in nodes in a wide variety of configurations can now use an implementation of the MS transport architecture to send alerts to NetView over LU 6.2 conversations. The NetView LU 6.2 alert receiver serves as the MS focal-point application for receiving alerts specified in the architecture as *ALERT_NETOP*. The LU 6.2 alert receiver function enables the hardware monitor to receive alerts about error conditions across the MS transport. Resolution major vectors specifying that an alert condition has been resolved can also be received and processed by an automation procedure maintaining status information about the recovered resource. Multiple Alerts or Resolution major vectors can now be consolidated. The hardware monitor processes the major vectors as if they were received individually.

Applications running on OS/2 Extended Edition with Networking Services/2 can use the alert entry-point function of Networking Services/2 to send alerts to the NetView LU 6.2 alert receiver. Likewise, NetView can serve as the alert focal point for an application running on any other product that implements an MS transport and an alert entry-point function.

Generic automation receiver. The generic automation receiver, NVAUTO, is a NetView application that receives structured data records across the MS transport and ensures that they are scanned against the NetView automation table for possible automation. The generic automation receiver enables a user to send data from an application in a local or remote node to NetView automation without creating a specific receiving application. This ability eliminates the need to explicitly develop a program that registers and receives the data to call for automation table processing.

The generic automation receiver registers with both the MS transport and operations management and registers in both entry-point and focal-point NetView nodes. Multiple registration allows an application to send data to the generic receiver either as a management services application or an operations-management-served application and from either an entry-point or a focal-point node.

High-performance transport and applications. We now discuss a transport option and its use in applications.

High-performance transport option. Many applications need to transfer larger amounts of data between pairs of nodes that would adversely affect the performance of the basic MS transport. For example, applications forwarding bursts of accounting data or several database records in succession require higher transaction rates that could impact the use of the basic MS transport for its intended purpose of carrying operations management, alert, and other management services data. The architecture for SNA Management Services includes a high-performance option of multiple-domain support[4] to serve this need. The NetView high-performance transport is an implementation of this architecture.

The MS transport uses brief conversations over shared sessions to minimize the overall network impact for transporting management services data. In addition, the MS transport requires a confirmation on each send to guarantee the reliable delivery of the application program data.

For those applications requiring higher transaction rates between fewer nodes, the high-performance transport allows the use of persistent conversations over dedicated sessions between the originating and destination nodes and eliminates the transport-level confirmations. In place of transport-level confirmations, the applications may either implement application-level error detection (such as the use of consecutive sequence numbers on messages) or accept the occasional loss of messages for those cases in which data integrity is not critical.

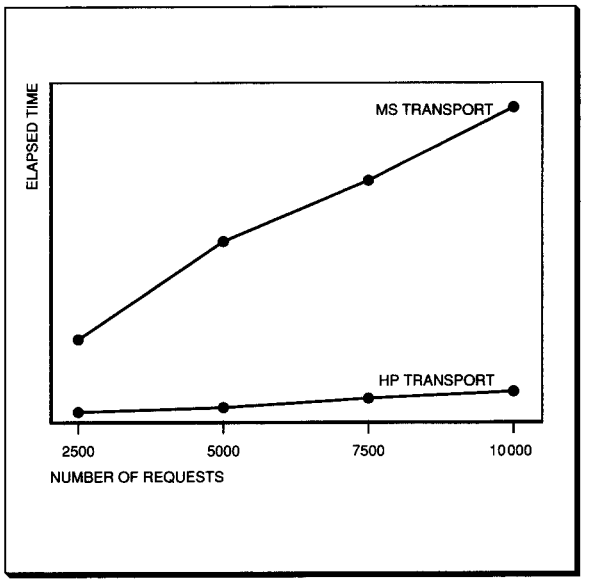
The NetView high-performance transport provides applications with a high-level programming interface similar to the MS transport. That is, an application registers with the transport and can then send and receive data as desired. Registration, sending, and receiving data can all be done with PL/I or C service routines or assembler language macros.

Figure 3 compares elapsed time for sending bursts of requests with no reply expected over a single session using the NetView MS and high-performance transports. Throughput for the MS transport is constrained because of the delay between sends waiting for the send confirmation. In this measurement, elapsed time for the high-performance transport was 90 percent less than for the MS transport.

Two NetView applications that use the high-performance transport are an application providing the capability to execute commands at a remote NetView and an application providing access to a remote database using the NetView Bridge. These applications are described in the following two subsections.

Remote NetView command execution. An important requirement for managing a multisystem network is the ability to route system, subsystem, and network commands to another host for execution at that host. Although such routing was previously possible from NetView, the operator had to issue a command and then log on to each remote NetView. Logging on required a password to be sent across the network, and operators needing to execute commands at many remote NetView systems spent a significant amount of time each day logging on to each NetView. Sending passwords across the network also presented a security exposure. Command lists were written to make it easier for operators to log on to multiple systems. However, these command lists increased the security exposure because passwords were coded in the command lists.

Figure 3 LU 6.2 transport throughput



In NetView Version 2 Release 2, the RMTCMD command uses the high-performance transport to provide remote command capability without requiring an operator to log on to the remote NetView and without requiring passwords to be sent through the network. The RMTCMD command can be used to route to another NetView any command that produces single-line or multiline messages or any command that produces no output. Security checking is provided to allow the receiving NetView to check the authority of any user at a remote NetView before starting or stopping the NetView task that will be used to execute the commands.

The RMTCMD command has several advantages over the use of NetView-NetView (NNT) sessions previously available in NetView. These include:

- Using a single pair of LU 6.2 sessions to multiplex requests from multiple operators instead of separate proprietary LU 0 sessions for each operator
- Enabling command execution and responses without requiring a START DOMAIN command followed by logging on to a remote NetView task
- Eliminating the need to send passwords across the network

- Increasing the 256-byte cross-domain data transfer limit to a 31 743-byte limit

Accessing a remote database using the NetView Bridge. The NetView program collects data from the resources in a network. These data include information about session awareness, system configuration, and problems within the network. The NetView Bridge provides a way to use these data to create, retrieve, and update problem, configuration, and other records in a non-NetView database such as an Information/Management database.

For example, when an alert is received, an automation program can use the NetView Bridge programming interface to connect to an external database and issue a query transaction to see if a problem record exists for the condition that produced the alert. If no problem record exists, one can be created using the data in the alert. Alternatively, the alert data can be used to update an already existing record with more information.

Although the external database must reside in the same host as a NetView running on Multiple Virtual Storage (MVS), remote access is provided for applications in a NetView running on the MVS, virtual machine (VM), or Virtual Storage Extended (VSE) operating systems (Figure 4). This remote access makes use of the NetView high-performance transport.

NetView automation platform

Automation plays an essential role in the management of today's information systems, in single-system environments as well as in multisystem networks. To enable a collection of remote systems to be operated from a central location, responses to many routine situations must be automated. These situations may include error recovery as well as normal operational conditions. Message and data traffic must be reduced to the point where remote operation becomes feasible. In addition, the level of performance and availability necessary for today's information systems cannot be achieved without a significant amount of system automation.

The NetView product provides several capabilities[1] that make up a platform for automating system and network operations. Automation programs can be written in the Restructured Extended Executor (REXX) language, the NetView command list language, PL/I, C, or assembler language. They can be executed in the NetView environment as commands or as installation exits at various points within NetView processing of commands or event notifications such as text messages or alerts. Timers can be set to execute commands or command lists at specified times, after a certain interval, or periodically at fixed time intervals.

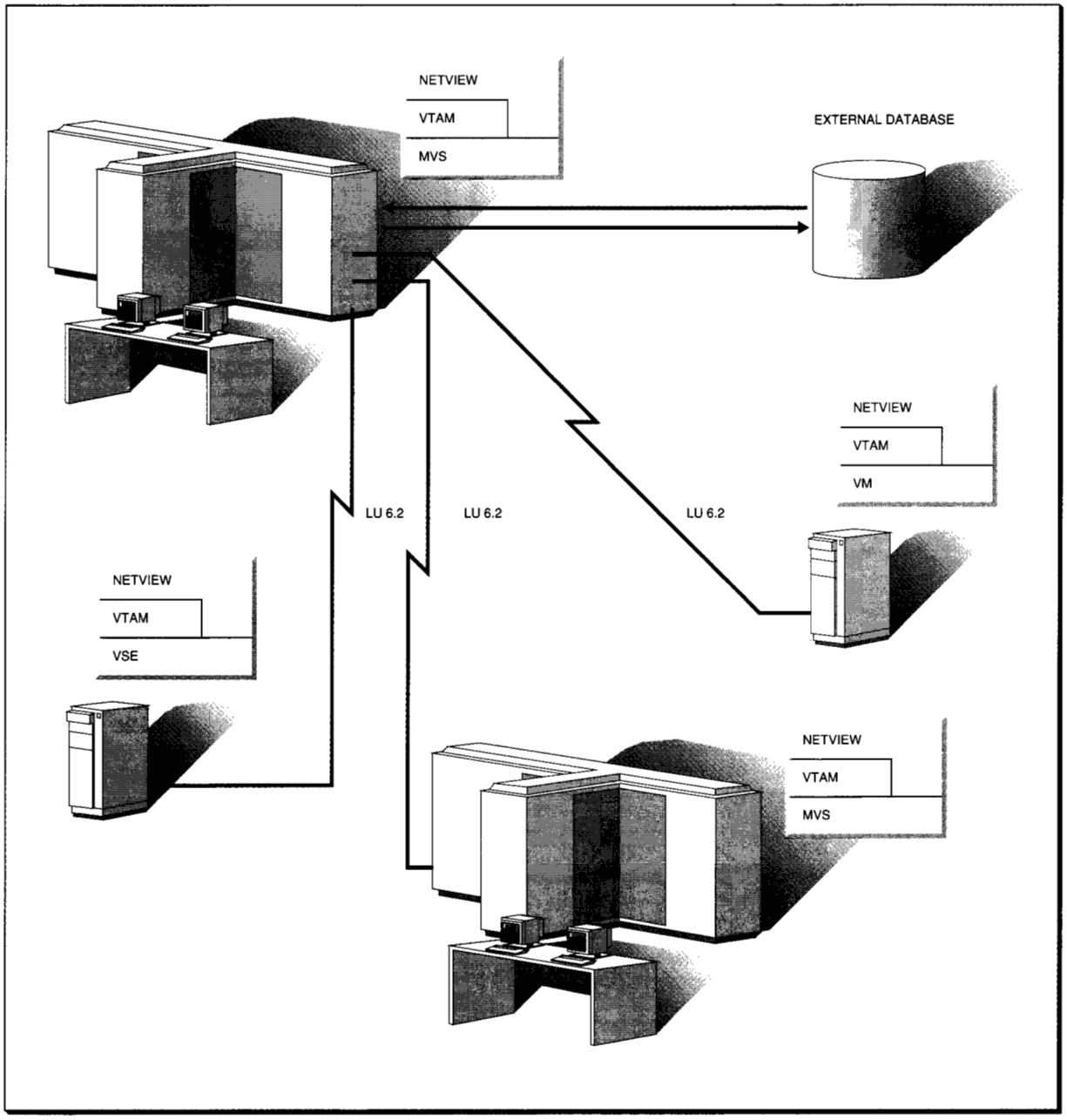
The NetView automation table provides a method for specifying automation actions to be performed when event notifications are received in the form of text messages or structured data such as alerts. These actions include executing an automation program and suppressing, logging, or displaying the message or alert information. The automation table language consists of IF-THEN and related statements that express criteria to be matched against incoming event notifications and actions to be taken when the criteria are matched. The specifications in a table can be activated by command. The same command allows an active table to be dynamically refreshed or replaced by another table.

As more and more customers have used the automation capabilities of NetView to automate parts of their system and network operations, several needs have been identified.

Previous to NetView Version 2 Release 2, the automation table handled only text messages. To use the table to take recovery actions based on the receipt of an alert about a problem with a resource in the network, the alert information had to be put into a text message and sent through NetView message processing. This procedure complicated the specification of criteria in automation table statements, and frequently important data were lost in the translation from structured alert formats into text messages.

As automation tables grew to several hundred statements, users reported concerns about the amount of processing time used to scan the table criteria for each message processed by NetView. Automation table performance was somewhat improved by placing statements with frequently matched criteria near the top of the table. However, this positioning increased the complexity of maintaining the table because statements to au-

Figure 4 Remote access to the NetView Bridge

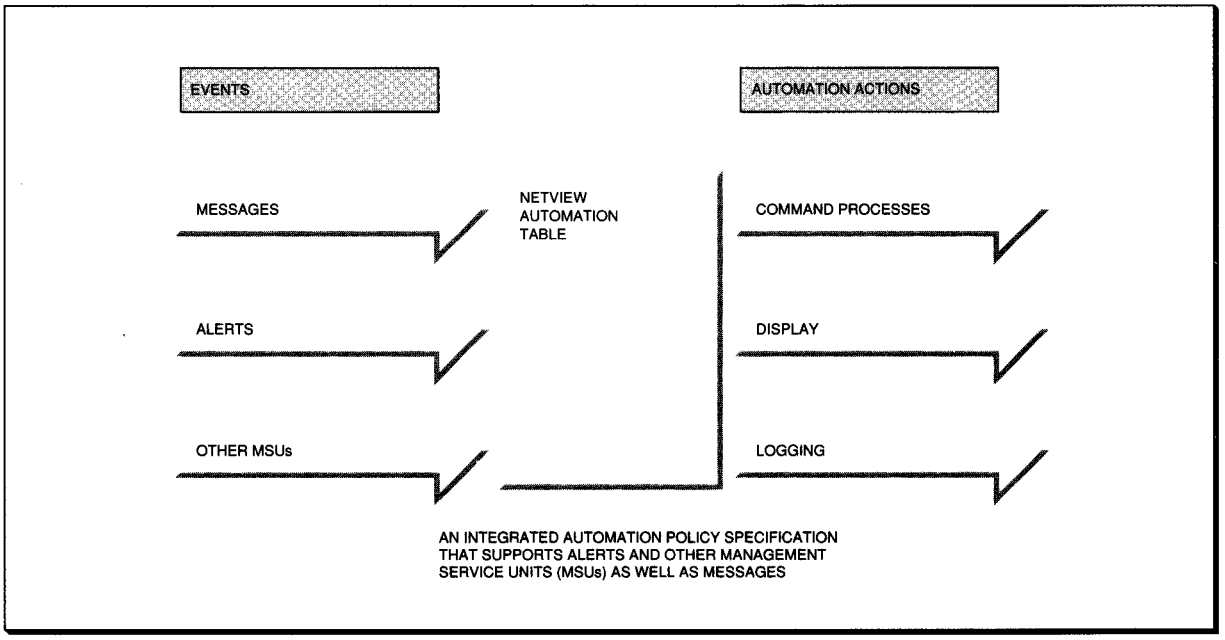


tomate messages from a single application or subsystem were spread throughout the table.

Within their organization, many customers have more than one group responsible for automating different facets of their information system operations. Sharing a single automation table file among several groups complicated table maintenance.

As customers have become more sophisticated in their automation techniques, they have reported

Figure 5 NetView automation table



limitations in the automation table capabilities that inhibit their ability to create the automation specifications they need to completely automate their operations. NetView provides the capability for automation programs to store important information in global variables for access at a later time and by other programs. Automation table users have frequently mentioned that accessing global variables in comparison criteria in the automation table would be very useful. Similarly, many have requested the ability to access information stored in control blocks in memory during automation table processing.

Previously the only automation actions taken for a message were those in the first automation table statement for which the message matched the criteria. Several customers have requested the capability for a single message to match the criteria and cause the actions to be taken for more than one automation table statement.

To meet these customer needs, Version 2 Release 2 of the NetView product enhanced the capabilities of the NetView automation table in several significant ways. These enhancements, listed below, are discussed in successive subsections:

- The automation table language allows the use of alerts and other structured data to be integrated with the use of text messages to cause automation actions to occur (Figure 5).
- Additional comparison criteria are provided for alerts received by the hardware monitor. Actions to control hardware monitor processing of alerts are also provided.
- Synonyms may be defined for complicated or difficult-to-understand portions of automation statements to make the table easier to understand and maintain.
- An automation table can be composed of multiple files.
- A listing can be produced containing all of the automation table statements in included files with all synonyms resolved.
- Automation table language enhancements make it easier to structure the table to simplify table maintenance and improve the performance of automation table processing.
- Event notifications can match the criteria in more than one automation table statement and cause the actions to occur.
- Automation table users can create their own functions to access data stored in memory for use in evaluating automation table statement criteria.
- Values in NetView global variables can be ac-

cessed in criteria comparisons in automation table statements.

MSU automation. The receipt of an SNA alert indicating a problem with a resource in the network is an event that frequently calls for a predictable response in an attempt to correct the problem. This response is especially predictable when the alert contains information about the probable cause of the problem and recommended actions to correct it. Clearly, for such a problem, a programmed, automated response is more reliable and efficient than relying upon an operator to see the alert, view the probable cause and recommended action information, and enter appropriate commands.

Since previous automation capabilities relied on the receipt of a text message to invoke an automation program, various methods have been devised to capture some of the information within an alert in order to issue a message that is sent to an operator so that it goes through message processing and causes automation to occur. Not only are these methods inefficient and difficult to program, but frequently the automation program must rely on incomplete data. Fields from the alert such as the "user text" field are not included in the text message. An automation method based directly on the alert itself was clearly needed.

With the advent of the management services and high-performance transports, another important source of information about applications and systems in the network is the data that are sent across these transports. As more and more applications are written with automation in mind, the use of structured data formats for program-to-program communication will increase.

The similarities between the data formats for alerts and other management services data sent across the LU 6.2 transports led to the term *management services unit* (MSU) for data in one of these formats and other formats using the same encoding scheme[5] specified in the architecture, such as the Network Management Vector Transport (NMVT) used on SSCP-PU sessions.

The NetView Version 2 Release 2 automation table provides a way of extracting any desired information from an MSU to be compared with predefined criteria. When the criteria are satisfied by an MSU that has been received, automated actions are performed such as executing an automation program or command list. A program that is executed as a result of automation receives as input the entire MSU. The MSU can be examined further to make processing decisions.

The extraction function is called *MSUSEG*. The MSUSEG function provides a natural method of extracting information from the vectors of an MSU that makes use of the SNA encoding format. The information in any major vector, subvector, or subfield within a subvector can be accessed directly for comparison purposes within the automation table. The capability of comparing against hexadecimal criteria has also been added to the automation table language since information in an MSU will frequently be in a hexadecimal format.

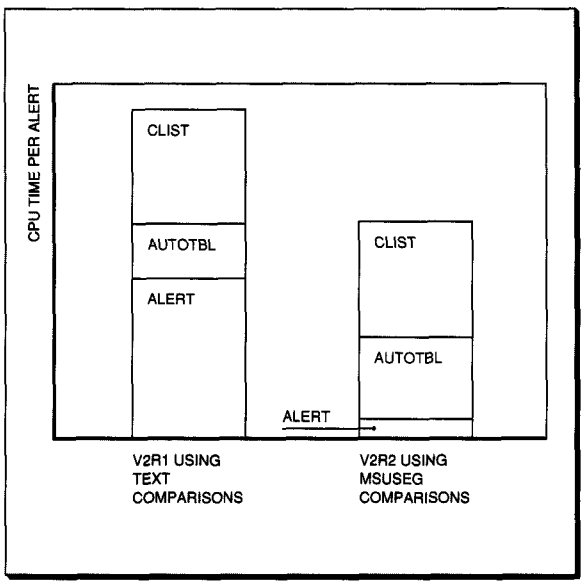
When an automation table is activated using the AUTOTBL command, NetView creates separate internal tables for messages and MSUs. To optimize performance, incoming messages are compared only against message criteria and incoming MSUs only against MSU criteria. This optimization reduces the number of unsuccessful comparisons that will be made.

PL/I, C, and assembler service routines are provided that enable applications receiving MSUs through the MS transport, high-performance transport, or SSCP-PU sessions to call for automation table processing of an MSU. The table is scanned, and if the criteria of any statement are met, the specified automation actions are performed. Thus applications are allowed to invoke automation processing directly with their data.

Alert automation. The hardware monitor component of NetView calls for automation table processing for any Alert or Resolution major vector that it receives in MSU format. In addition to the MSU containing the Alert or Resolution major vector, information about the source of the MSU that may be available in the hardware monitor resource hierarchy list is made available to automation table processing. This resource hierarchy information can be checked in the comparison criteria, and it is available to any automation program or command list that is executed because of a match in the table.

The hardware monitor provides the SRFILTER command to set recording filters that determine the extent of hardware monitor processing for an

Figure 6 Alert automation comparison



alert that is received. These filters determine whether an alert is logged as an event in the hardware monitor database, whether it is logged as a hardware monitor alert for display on the hardware monitor alerts dynamic panel, whether a text message is created from the information in the alert and sent to a NetView operator, and whether the alert is forwarded to a NetView hardware monitor focal point.

The call to automation table processing occurs after the hardware monitor has checked the recording filter settings for an alert set with the SRFILTER command but before the hardware monitor acts on the settings. An automation table action called *SRF* can be used to override the recording filter settings. For example, an alert may meet general criteria set with SRFILTER to be logged as an event and a hardware monitor alert, but by matching more stringent criteria in an automation table statement, the SRF action can reset the filters to block recording by the hardware monitor and display on the hardware monitor alerts dynamic panel.

For events logged as hardware monitor alerts, other automation table actions can be used to control the color and highlighting to be used when the alert is displayed on hardware monitor panels.

The direct automation of alerts from the hardware monitor eliminates the need for alerts to be transformed into messages to drive automation procedures. In addition to providing access to all of the data in the alert, additional message processing is avoided. If automation can completely handle the alert, and if the SRF action is used to block the alert from hardware monitor recording or alerts dynamic processing, considerable hardware monitor processing is avoided as well.

Figure 6 compares processing time for automating an alert in NetView Version 2 Release 1 (V2R1) and in NetView Version 2 Release 2 (V2R2).

The hardware monitor processing is denoted by "alert" in the figure. In NetView V2R1, the alert passed the hardware monitor filter to create a hardware monitor text message suitable for automation table processing. This required it to also be logged in the hardware monitor database as an event and a hardware monitor alert to be displayed on the hardware monitor alerts dynamic panel. In NetView V2R2, the alert was sent through automation table processing and blocked from further hardware monitor processing. Hardware monitor processing time was reduced in V2R2 by 90 percent.

In each case, automation table processing produced a match with the criteria in the twenty-fifth entry of the table. Using 25 comparisons in each case increased the automation table processing time by 48 percent because MSUSEG criterion comparisons require more processing than simple text comparisons.

In both cases, the same simple REXX command list was executed as a result of matching the automation table statement criteria.

The total processing time to automate a single alert decreased by 35 percent in NetView V2R2. Further improvement in the overall system processing to automate alerts in NetView V2R2 is expected for the following two reasons:

- By using BEGIN-END statements as described in a subsequent subsection of this paper, unsuccessful MSUSEG comparisons can be minimized and automation table processing time can be improved.
- In most installations using NetView, the arrival rate for alerts is much less than the arrival rate

for text messages. MSUSEG comparison processing consequently occurs less frequently than text message comparisons. Using MSUSEG instead of text message criteria to automate alerts will decrease processing by decreasing the number of times the criteria must be evaluated.

Additional automation enhancements. Described in this subsection are several enhancements in the automation platform.

Defining synonyms. The SYN automation table statement allows synonyms to be defined for any part of an automation table statement. The synonym can be used in any subsequent automation table statements. When the automation table is activated, NetView substitutes the synonym value for the name.

Since MSUs are constructed of major vectors, subvectors, and subfields with fixed hexadecimal keys, and frequently contain values that are hexadecimal code points specified in the architecture to carry specific information, MSUs are well-suited for communication between programs. With the MSUSEG function, precise automation table criteria can be specified to automate MSUs. However, hexadecimal specifications are difficult for people to read and understand. Using the SYN statement to define natural language synonyms for MSUSEG and other hexadecimal specifications and using the synonyms in subsequent automation table statements makes the automation table easier to read and understand.

Using a word or short phrase as a synonym for a long, repetitive string also makes an automation table easier to construct and read. In addition, using synonyms for values that may change, such as the name of the NetView domain in which the table will be used, makes it easier to modify and maintain an automation table because a value can be changed throughout the table by changing it in one place.

Including subtables. As automation has become more important in meeting their business objectives, providers of information system services have written automation procedures and automation table statements to automate more and more messages.

In many organizations, those who handle automation of systems and applications are not the same people who handle automation of network events. In addition, the people with the expertise needed to automate messages from one application or subsystem are frequently not the same as those who are familiar with another subsystem. Sharing and updating a single automation table member by several groups, each maintaining the portions they are responsible for, is difficult and error-prone.

The %INCLUDE function provided in NetView V2R2 makes it possible to create an automation table from several files or data set members. The separate files are included in one master file to be activated when an AUTOTBL command is issued. This makes automation table maintenance easier by allowing each group within the organization that has responsibility for a portion of the automation definitions to have its own separate file.

The %INCLUDE function also makes it easier to maintain automation tables in a multisystem environment. The system-specific portion of the table can be a separately included file, and the master file containing the automation definitions common to all systems can be distributed to each system without change.

Creating listings. Since an automation table can be composed of multiple files created by different people and can contain synonyms for several complex statements, it is important to have a complete listing of the table for auditing and error-checking purposes. An option on the AUTOTBL command provides an output listing with all included files and resolved synonyms whenever an automation table is activated or tested for correct syntax. Error messages describing any syntax errors are also included in the listing.

Structure and performance improvements. Many automation tables have grown to include hundreds of IF-THEN statements. As an unstructured list of statements, these tables are difficult to maintain even if only one person has access to them. Moreover, since the table is searched sequentially when a message is received, considerable processing time is expended searching to see if the message meets the specified criteria for every statement until there is a match.

The addition of a BEGIN-END syntax to the automation table language enables the table to be logically structured. For example, by checking to see

Figure 7 Partial automation table with BEGIN-END example

```
*
*-------------------------------------------------------------*
* Perform high level check to determine if VTAM IST message   *
*-------------------------------------------------------------*
IF MSGID = 'IST'. THEN
  BEGIN;
*   *-----------------------------------------------------*
*   * Perform intermediate check to determine if IST0 message *
*   *-----------------------------------------------------*
    IF MSGID = 'IST0'. THEN
      BEGIN;
*       *---------------------------------------*
*       * IST020I  VTAM INITIALIZATION COMPLETE  *
*       *---------------------------------------*
        IF MSGID='IST020I' THEN
          EXEC(CMD('MVS V NET,ACT,ID=A19A53C,SCOPE=ONLY')
            ROUTE(ONE MVSAUTO))
          EXEC(CMD('MVS D A,TSO') ROUTE(ONE MVSAUTO));
*         .
*       * Other messages beginning with IST0 would go here  *
*         .
      END;
*         .
*   * Remaining automation for VTAM IST messages would be defined here  *
*         .
  END;
*
*-------------------------------------------------------------*
* Perform high level check to determine if NetView DSI message  *
*-------------------------------------------------------------*
IF MSGID = 'DSI'. THEN
  BEGIN;
*         .
*   *  DSI automation would be defined here    *
*         .
  END;
```

if the message identifier begins with *IST*, all automation of Virtual Telecommunications Access Method (VTAM*) messages beginning with IST can be placed within a single BEGIN-END statement group as in Figure 7. This capability improves not only the ease of maintaining the table in logical segments, but also the performance of automation table comparison processing when the table is activated. In the example in Figure 7, any message received that did not come from VTAM is checked only to see if its identifier does not begin with IST and then to see if its identifier begins with *DSI*. The criteria for all the specific VTAM messages beginning with IST are skipped over.

This technique has been used to restructure the sample automation table supplied by NetView. To compare the automation table processing for criterion comparisons between NetView V2R1 and NetView V2R2, three measurements were made for a message that matches the criteria in the last statement of the NetView sample automation table:

1. The V2R1 sample was measured on a V2R1 NetView.
2. The V2R1 sample was measured on a V2R2 NetView.
3. The V2R2 sample was measured on a V2R2 NetView.

The results of these measurements are shown in Figure 8. Because of improvements made in NetView V2R2 to the criteria comparison algorithm, the processing time for the NetView V2R1 sample automation table was reduced by 16 percent in NetView V2R2. The processing time for the NetView V2R2 sample table using BEGIN-END statements improved by 97 percent over NetView

V2R1. Although the improvement for different automation tables will vary considerably, significant improvement can be expected from appropriate restructuring using BEGIN-END segments.

Matching more than one statement. The CONTINUE automation table action allows a single message or MSU to satisfy the criteria of more than one automation table statement and cause the execution of automation processes defined for each statement where the criteria are matched. When the message or MSU matches the criteria of a statement without CONTINUE specified, scanning of the table for that message or MSU ends, and all of the actions accumulated from matched statements up to that point in the table are performed.

Creating additional automation table functions. Automation table function (ATF) support allows a synchronous call to an IBM-supplied or user-written assembler program within automation table criteria processing. The program can process the message or MSU, check or set values in internal storage, and return information to the automation table to be used as match criteria for the automation table statement. For example, an ATF could return the name of the NetView task in which the message or MSU is being processed, and this information could be used in determining whether to perform the automation actions for the automation table statement. NetView provides such an ATF called *OPERID* as a sample. Using it in an automation table would look something like the following:

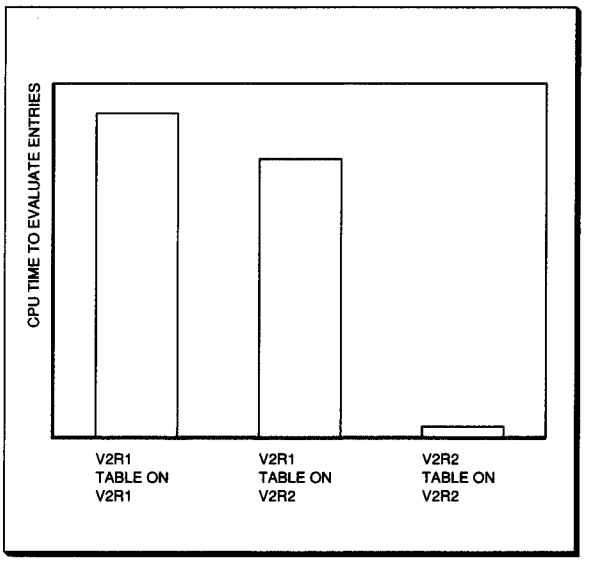
```
IF MSGID = 'DSI020I' & ATF('OPERID')
    = 'OPER1' THEN ...
```

Accessing global variable values. Two IBM-supplied ATFs allow access to NetView global variable values from the automation table. The values of common and task global variables can be used as criteria for a match when scanning the table for a message or MSU that has been received.

Summary

As information systems have become more complex and more critical in the daily operation of today's businesses, the ability to operate remote systems from a central location has become increasingly important. New architectures under SNA lay the framework for achieving unattended remote system operation. The NetView Version 2 Release 2 implementation of these architectures provides a platform for remote system operation and management. This platform helps reduce the operations personnel and resources needed to manage today's information system networks. Using NetView V2R2, information services providers can more easily meet their business objectives and maintain their service levels in a cost-effective manner.

Figure 8 V2R1 and V2R2 sample automation table comparison



Similarly, automation of system and network operations is crucial for providers of information systems services. Extensive enhancements to the automation capabilities in NetView have been made in NetView V2R2. They include the ability to automate alerts and other structured data directly as well as other enhancements that improve the performance, usability, and functional capabilities of the NetView automation table. Although many challenges remain in providing the automation capabilities and solutions needed to manage systems and networks of the future, the automation support in NetView V2R2 goes a long way toward providing today's automation needs. Coupled with the object-oriented automation capabilities that will be available using the Resource Object Data Manager (RODM) in NetView Version 2 Release 3, the NetView V2R2 automation enhancements have made NetView the SystemView* automation platform.

Acknowledgment

The performance measurements and the graphs illustrating them in this paper were supplied by J. T. Jennings from the IBM Network Management Systems Performance group.

*Trademark or registered trademark of International Business Machines Corporation.

Cited references and notes

1. For more information on the origin and composition of NetView and the functions available in NetView Release 2, see D. Kanyuh, "An Integrated Network Management Product," *IBM Systems Journal* **27**, No. 1, 45–59 (1988).
2. For more information on the MS transport architecture and the SNA focal-point architecture, see M. O. Allen and S. L. Benedict, "SNA Management Services Architecture for APPN Networks," *IBM Systems Journal* **31**, No. 2, 336–351 (1992, this issue).
3. For more information on the NetView MS transport programming interface, see *NetView Application Programming Guide*, SC31-6098-0, IBM Corporation; available through IBM branch offices.
4. *Systems Network Architecture Management Services Reference*, SC30-3346, IBM Corporation; available through IBM branch offices.
5. For an overview of SNA encoding formats, major vectors, subvectors, and subfields, see Reference 2.

Accepted for publication December 20, 1991.

B. William Irlbeck *IBM Networking Systems, 3039 Cornwallis Road, P.O. Box 12195, Research Triangle Park, North Carolina 27709.* Dr. Irlbeck is currently an advisory programmer in NetView design and was the chief designer for NetView V2R2. Since joining IBM in 1982, he has worked in network management development and design. He received a B.S. degree in mathematics from West Texas State University and M.A. and Ph.D. degrees from the University of Montana. Prior to joining IBM, Dr. Irlbeck taught mathematics at Xavier University of New Orleans.

Reprint Order No. G321-5470.