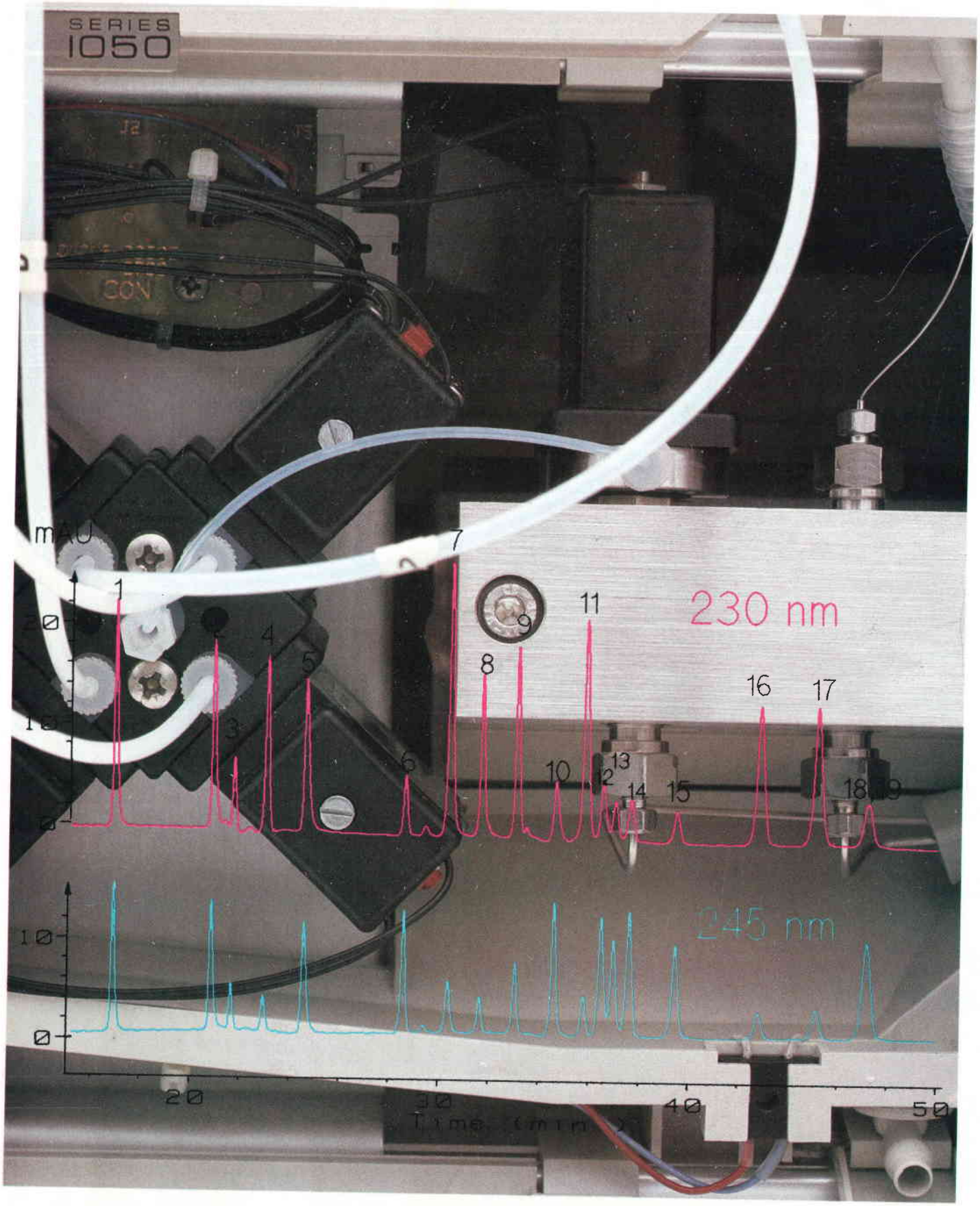
HEWLETT-PACKARD
JOURNAL

APRIL 1990



HEWLETT
PACKARD

HEWLETT-PACKARD JOURNAL

April 1990 Volume 41 • Number 2

Articles

Departments

The **Hewlett-Packard Journal** is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company makes no warranties, express or implied, as to the accuracy or reliability of such information. The Hewlett-Packard Company disclaims all warranties of merchantability and fitness for a particular purpose and all obligations and liabilities for damages, including but not limited to indirect, special, or consequential damages, attorney's and expert's fees, and court costs, arising out of or in connection with this publication.

Subscriptions: The Hewlett-Packard Journal is distributed free of charge to HP research, design, and manufacturing engineering personnel, as well as to qualified non-HP individuals, libraries, and educational institutions. Please address subscription or change of address requests on printed letterhead (or include a business card) to the HP address on the back cover that is closest to you. When submitting a change of address, please include your zip or postal code and a copy of your old label.

Submissions: Although articles in the Hewlett-Packard Journal are primarily authored by HP employees, articles from non-HP authors dealing with HP-related research or solutions to technical problems made possible by using HP equipment are also considered for publication. Please contact the Editor before submitting such articles. Also, the Hewlett-Packard Journal encourages technical discussions of the topics presented in recent articles and may publish letters expected to be of interest to readers. Letters should be brief, and are subject to editing by HP.

Please address inquiries, submissions, and requests to: Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304, U.S.A.

HP OpenView Network Management

HP OpenView is HP's first set of integrated hardware and software products designed to address the needs of managing open, standards-based, multivendor networks in a consistent, user-friendly manner.

by Anthony S. Ridolfo

NETWORK OPERATORS RUNNING a large computer center are often confronted with questions like the following:

Why can't I connect to the system?
How come response is so slow?
What's wrong with the electronic mail system?
What computers can I connect to from my terminal?

Network managers are also confronted with questions dealing with network planning, inventory control, and accounting. Some typical network management questions include:

What is the overall load on the network?
Which systems are bottlenecks in my network?
How can I plan for new components in my network?
What equipment do I really have, and where?

As departments, sites, and whole companies (referred to as "enterprises") integrate and interconnect their data processing tasks, the job of managing these complex networks of computer and data communications equipment becomes increasingly more important to a company's bottom line. Monitoring the health of a network, diagnosing problems as quickly as possible, controlling the individual components, adjusting network-wide parameters and configurations, accounting for use of network resources, and planning for future expansion are the objectives of network management applications.

These tasks would be relatively simple if all the components in a network came from the same vendor. However, it is indeed rare that all links in a network such as modems, modem cables, multiplexers, pads, LAN connectors, LAN cards, LAN cable types, hubs, bridges, servers, gateways, and PBXs come from the same vendor. There are also different personal computers, workstations, minicomputers, and mainframes. And there are different wide area point-to-point networks, public and private X.25 packet switching networks, and a company's enterprise-wide networks that may even link to their suppliers' and distributors' networks.

The computer industry and international organizations such as ISO (International Organization for Standardization) recognize the need to standardize communication and network management protocols. It is HP's stated objective to be the leader in open, standards-based, multivendor networking. This includes the capability to manage and be managed by standards-compliant applications running on equipment supplied by any vendor.

However, having applications that can access and control a network is not enough to address the needs of network managers and network operators. Each type of equipment to be managed has its own particular command interface and management capabilities. Therefore, a unifying, consistent, user-friendly interface is needed for these applications.

HP OpenView

The HP OpenView network management family is designed to address these needs of managing open, standards-based, multivendor networks in an open, consistent, user-friendly manner. The HP OpenView products provide a consistent user interface and an integrated environment for monitoring, diagnosing, controlling, and measuring the performance of network components. From a single display, a network operator can see a graphical representation of the network components and their interrelationships, make configuration changes, and run diagnostic and performance gathering applications.

The design philosophy for the HP OpenView products is to create for the end user an easy to learn, easy to remember paradigm for accessing sophisticated network management applications. HP OpenView Windows, which is the HP OpenView product that provides the user interface, runs on an HP Vectra ES/12 personal computer as a MicroSoft® Windows application. Each of the HP OpenView products, in turn, runs under HP OpenView Windows. The user first runs the HP OpenView Windows draw (OVDraw) program that allows the user to draw a map of the network, with specific icons representing network devices, such as an HP 3000 business computer or a bridge (see Fig 1). The user then saves this map, and runs the HP OpenView Windows run-time program (OVRun). The map becomes a dynamic, graphical shell for accessing the applications running under HP OpenView Windows. In addition, if there exists an application reporting the state of a device to the HP OpenView system, the current operational health of the device is indicated by the color of the specific icon (e.g., red for critical, yellow for warning, and green for OK).

Suppose the network operator wishes to set some parameters on a specific bridge on the network. The operator would click the mouse on the symbol representing the specific bridge and select the menu item Set Parameters.... This selection causes HP OpenView Windows to pass program control to the HP OpenView BridgeManager software, which puts up a dialog box for this function. Suppose now the operator needs to set parameters on a specific HP 3000

Microsoft is a U.S. registered trademark of Microsoft Corp.

datacom and terminal controller (DTC). Again, the operator selects the specific icon and then the Set Parameters... menu item. Program control now passes to the HP OpenView DTC manager, which puts up the relevant dialog box for this function. At no time does the operator need to know which product to invoke, or how to invoke it. HP OpenView Windows and the graphical shell perform this function by controlling what menu items are enabled for which devices or icons. The user is relieved of the need to know what actions are legal or illegal for a particular type of device. The article on page 60 describes HP OpenView Windows in more detail.

This type of intelligent user interface is important in network management because network operators usually do tests as a response to an alert of some kind, such as an icon changing color on the display. They do not perform testing and diagnostic functions all the time, or even on the same equipment. It is therefore important to maximize the familiarity of the user interface as well as to minimize the number of ways the user invokes functions to do similar tasks. HP OpenView Windows and its supporting applications help with this task by providing a context sensitive, interactive system for guidance.

Network Management Applications

The HP OpenView articles in this issue describe the development efforts and the underlying architecture of the initial set of network management products in the HP OpenView family. These products include:

- **HP OpenView Windows.** This product provides the user interface to network management applications and a set of utilities that enable developers to create network management applications to run in the HP OpenView Windows environment. HP OpenView Windows is divided into two main parts: the end user run-time product and the HP OpenView Windows developer's kit. The end user run-time product includes the hardware and software required to use HP OpenView Windows. The developer's kit includes the end user run-time product and the necessary libraries, include files, sample source code, and documentation to facilitate the developer's task of creating a Microsoft Windows application that can be integrated into the HP OpenView end user run-time product. The HP OpenView developer's kit includes a style guide for helping developers ensure that the detailed look and feel of their dialog boxes are consistent with other, independently developed applications.
- **HP OpenView Windows BridgeManager.** This product provides centralized monitoring and control of HP 28648B 10-Mbit/s LAN Bridges and HP 28647B StarLAN Bridges (revision B bridges only). Using the network map feature of HP OpenView Windows, bridges can be easily labeled and identified. Menus allow quick access to a variety of bridge management features, including configuration, monitoring, control, performance management, and problem identification. The HP OpenView BridgeManager is described on page 66.
- **HP OpenView Data Line Monitor.** This product provides the ability to monitor 4-wire leased point-to-point analog data lines using an HP 4948A In-Service Transmission Impairment Measuring Set controlled from the HP OpenView workstation. It is a network monitoring system that can measure performance and aid fault isolation during troubleshooting while the network is being used. The HP OpenView Data Line Monitor is described on page 71.
- **HP OpenView DTC Manager.** This product provides centralized and integrated network management for both terminal connectivity and X.25 networking. It configures, monitors, diagnoses, controls, and downloads software to DTCs (datacom and terminal controllers). From a single HP Vectra running HP OpenView Windows and the DTC Manager, an operator can manage local and

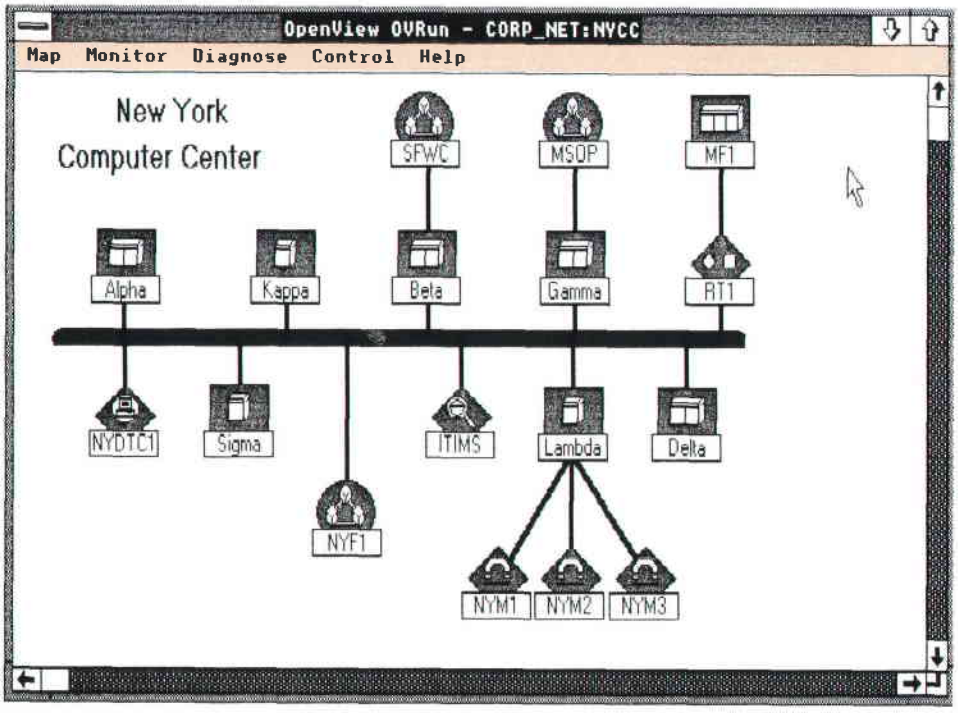


Fig. 1. *An HP OpenView Windows map showing the network configuration for a computer center.*

remote terminal connections and remote system-to-system communications across multiple DTCs on a LAN. The HP OpenView DTC Manager is described on page 76.

- **HP OpenView NS Monitor.** This is an internal HP tool developed to test and refine the HP OpenView architecture and the facilities provided by the HP OpenView Windows software. It is not available as a product. It provides network management services in the form of centralized or distributed network status, diagnostic, and performance monitoring for HP 3000 computers running the MPE V operating system. The user interface software runs on an HP Vectra personal computer under HP OpenView Windows, and the programs that perform the diagnostic and performance monitoring run on an HP 3000 computer that is designated as the management node. The systems being managed are called managed nodes. The user interface software on the Vectra and the programs on the management node communicate with each other via HP OfficeShare on the Vectra and NetIPC on the HP 3000. These products are described on page 85.

Conclusion

A direct result of the HP OpenView design philosophy is that application developers have to learn to split the functionality of their products between the network management functions and the user interface code. This is in alignment with HP's NewWave philosophy and the vision of truly distributed applications. Although network management products address a very specific end user, the architecture and implementations used to develop the products mentioned above represent the future of computer applications. The article on page 54 describes the HP OpenView network management architecture that provides the models to guide the planning, analysis, and design of network management products developed to run in the HP OpenView Windows environment.

Acknowledgments

The development of the initial set of HP OpenView products was the combined efforts of teams at HP's Information Networks Division (IND), Roseville Networks Division (RND), Grenoble Networks Division (GND), Colorado Networks Division, and Queensferry Telecommunications Division (QTD). I thank all the engineers and managers at these organizations who contributed to the HP OpenView product family. Specific thanks go to the following individuals: Larry Goldman, CND section manager, Alan Pare, IND distributed applications lab manager, Catherine Smith, IND project manager for HP OpenView Windows, Atul Garg, IND project manager for the HP OpenView diagnostic and performance monitors, Shnider Youssef, IND project manager for the HP OpenView core services, Phill Magnuson, RND project manager for the HP OpenView BridgeManager, Mike Hurst, QTD project manager for the HP OpenView Data Line Monitor, Bernard Guidon, HP Information Networks Group marketing manager, and Jean-Jacques Ozill, GND project manager for the HP OpenView DTC manager.

HP OpenView Network Management Architecture

This article highlights the principal objectives of the architecture and the reference models used to support the HP OpenView product development.

by Keith S. Klemba, Mark L. Hoerth, Hui-Lin Lim, and Maureen C. Mellon

THE USE OF INFORMATION NETWORKS in commercial, government, and academic organizations has exploded in the 1980s. With wide and local area networks, computing power has migrated from a centralized to a distributed environment. This has reduced costs, enhanced competitiveness, and renewed organizational creativity. The explosion in the number of networks and network devices from a variety of vendors has caused a dramatic increase in network complexity and an acute need to manage these distributed resources more effectively.

The need for network management is apparent today throughout any organization deploying networks. The enterprise network in use by many organizations includes both local and wide area networking technologies and an assortment of network devices from different vendors, and is managed by a team of professionals in different geographic areas. Fig. 1 shows the domain of a typical enterprise network.

Companies are now using information management to gain a competitive advantage by delivering goods and services faster and more efficiently. While the benefits of network management are most apparent during a crisis, they are no less important in day-to-day network operation to control network faults, configurations, performance, or se-

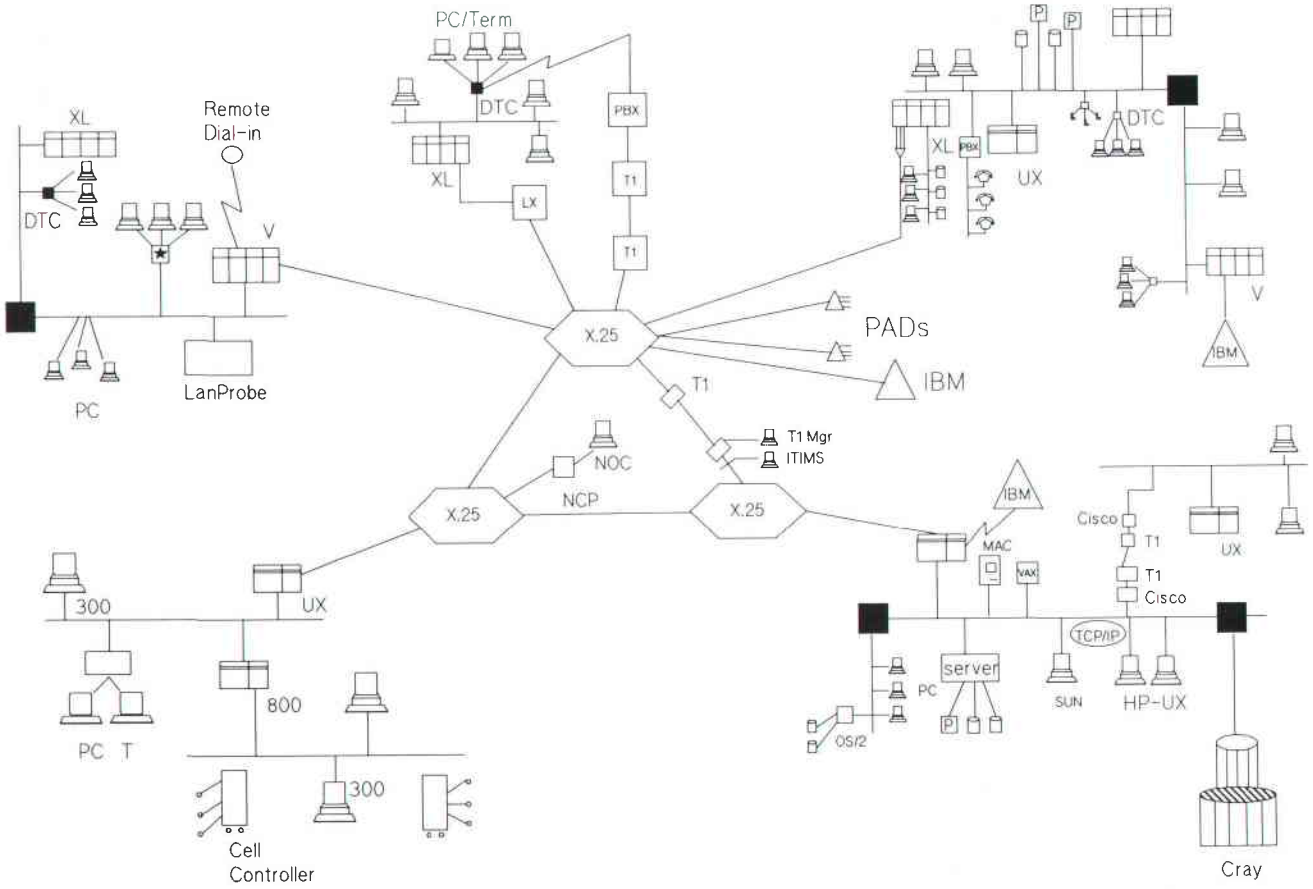


Fig. 1. *Typical enterprise network includes both local and wide area netorking technologies and an assortment of network devices from different vendors.*

curity. An integrated network management system can save time and money by reducing downtime and performance problems.

Problems of Stand-Alone Element Management

During the last decade, most of the computer and networking industry focused on the rapid evolution of network technology to the neglect of network management products. As the need for network management became apparent, the first products made available were stand-alone element managers. These products manage a particular set of network elements, such as bridges, modems, or routers. As more devices are included in the network, the network manager must add more stand-alone element managers. With no correlation, aggregation, or prioritization of information across applications or consoles, it quickly becomes impossible for the network manager to use this large number of element managers effectively. Typically, each element manager uses a different user interface and style of operation, so that different training and expertise are required for each system.

In contrast, an integrated network management system like HP OpenView combines and consolidates the management of network elements from various vendors. It allows the customer to monitor, control, automate, and repair network segments and equipment from a single console and user interface.

Dimensions of Integrated Network Management

The problem of managing the enterprise network can be divided into three components: the users of network management, the objects to be managed, and the functional needs of network administrators. Each component influences network management architecture by imposing requirements for effective management.

Users. There are several users of network management. The corporate network manager works at the executive level and is concerned about network costs, uptime, and strategic planning. The main objectives are to control the network asset and obtain consistent, maximal performance. Telecommunications and MIS directors implement and operate large portions of corporate wide area networks. These managers are concerned about network growth, uptime, and planning. At the local area network level, data communications specialists, system operators, and site telecommunications managers oversee work-group networks in environments ranging from engineering research and development to manufacturing plants to business offices. Users of local area networks rely on their managers to maintain the connection into the local EDP environment, provide assistance with personal computer software management, and keep the network operating during critical periods.

Resources to Be Managed. These fall into four categories or layers, as shown in Fig. 2. The transmission layer, which corresponds to the first layer in the ISO OSI model, includes the physical media, such as T1 multiplexers, modems, broadband cable, fiber optic cable, and the like. The data network layer, which includes transports and services, covers LANs, X.25 and SNA networks, and OSI services. A growing number of customers are adding voice elements to this category. The computation network layer includes networked systems and networked data bases. The networked applications layer consists of distributed applications in areas such as X.400 electronic mail, electronic data interchange, and office automation.

Functional Requirements. HP defines several specific functional areas for network and system management. Fault management provides the ability to identify, diagnose, and resolve network problems quickly. It also includes status monitoring, alarms and alerts, and predictive expert system tools. Configuration management delivers the ability to track network and device configurations from a central control point. Performance management provides the ability to optimize network performance through the collection and analysis of device performance data. Accounting provides the network manager with network use information. Security protects the network and its components from unauthorized intrusion or surveillance. Inventory management addresses the need to track, monitor, and maintain assets over a wide geographic area.

The foundation for integrated network management is vendor support and service. Faced with budget constraints and a shortage of skilled staff, network managers need planning, implementation, and operation support. Consulting, training and support services can speed the implementation of integrated network management and ensure its effective use.

A Standards-Based Architecture

The HP OpenView network management architecture (NMA) is rooted in international and de facto industry standards.[1,2,3] HP OpenView NMA is based on the Open Systems Interconnection (OSI) management framework and models developed by the International Organization for Standardization (ISO). Standards provide HP OpenView NMA with a solid foundation for providing interoperability in heterogenous, multivendor enterprise networks by defining network management protocols and mechanisms for sharing management information. As a result, products based on HP OpenView NMA can be used to manage any type of network that conforms to the OSI standards. HP OpenView NMA derives three essential elements from the OSI standards on which it is based: a network management framework, a well-defined mechanism for describing managed objects, and a set of services and protocols for communication.

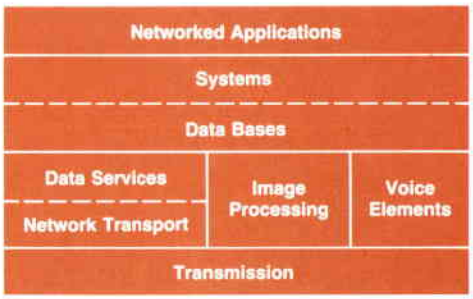


Fig. 2. *Network resources to be managed fall into four layers.*

OSI System Management Model

In 1988, the ISO specified the OSI System Management Model as a basic framework for network management. In the OSI model, a *manager process* is responsible for realizing the specific management functions requested by the user through interactions with *agent processes*. The agent processes represent the management services offered by *managed objects*.[4] Possible management services include:

- Operations such as reading a counter
- Actions such as resetting a network device
- Notifications (known as events) such as an indication that a threshold has been exceeded.

Fig. 3 illustrates the OSI System Management Model. The manager process and the agent process use a Common Management Information Protocol (CMIP) to exchange management information.[5] The interface labeled A is the only open integration point in the OSI System Management Model. The management services offered by managed objects at this interface are defined through the use of a standard specification scheme known as the OSI Structure of Management Information.[6] The interface labeled B is an unspecified integration point that can be implemented using industry-standard or proprietary protocols.

OSI Object Model

The Structure of Management Information is a set of rules or guidelines for defining *classes* of managed objects. The rules are used when defining each class of managed object to ensure specification uniformity Classes of managed objects can be defined for any manageable network resource. For example, there can be a managed object class that describes LAN bridges, another that describes computer systems, and a third that describes network equipment in general.

Object classes are defined by specifying their attributes, operations, actions, and events. Once an object is defined, it is placed within a registration hierarchy and a unique class identifier is assigned by the registration authority. For example, a managed object describing a piece of network equipment might have:

- Attributes detailing its physical location, state, and percentage of utilization
- Actions to request its activation or deactivation
- Events such as alarm and change reporting.

The use of an object model is important because it brings with it the concept of inheritance. In the context of network management, inheritance allows refinement of existing classes while ensuring compatibility with existing software. The inheritance relationship that exists between object classes is important because it allows existing management applications to work with the new object class, and it provides a mechanism for software reuse.

The set of managed objects within a system constitutes that system's Management Information Base.[2] Instances of managed objects exist within a containment hierarchy referred to as a *containment tree*. For example, a real open system (computer) would contain numerous managed objects such as a routing table, which would contain entries, or an n-layer protocol, which would contain n-layer connections. Some of the benefits associated with the containment relationship are:

- If a managed object is deleted, all managed objects contained within it are also deleted.
- Management requests can be directed to a group of objects related by containment using scoping.

Beyond the OSI Standards

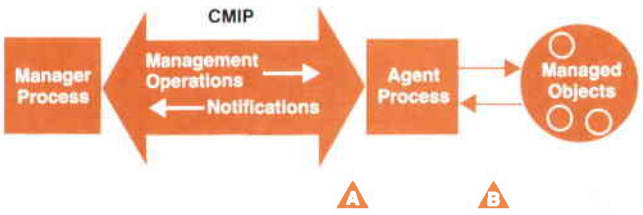
HP OpenView NMA specifies a complete environment of services and facilities available to management applications distributed throughout the network. This vision of HP OpenView NMA requires the addition of architectural components beyond those described in the OSI System Management Model.

The HP OpenView NMA reference model, illustrated in Fig. 4, has nine components. It refines the OSI model with three major extensions, which are essential to the management of complex, multivendor enterprise networks. First, HP OpenView NMA creates a distributed network management communication infrastructure consisting of three components known as *supervisor* (S), *postmaster* (P), and *communication profiles* (C). Second, HP OpenView NMA adds an additional point of integration by dividing the manager process into a *management application* (A) and a *user interface* (U). Third, HP OpenView NMA extends the object-oriented paradigm in two ways. The need for managed object persistence is addressed with the specification of managed object *data stores* (D). The scheme used to define managed objects (O) is also used to specify key application functionality in the manager process as a managed object, making the management services (M) it provides available for reuse. This provides another open point of integration. Finally, *environmental services* (E) are the services provided by the native operating system and these can be used by any of the other components.

HP OpenView Object Model

U	S	O
A	P	M
E	C	D

Object-oriented concepts and technology are fundamental to HP OpenView NMA. The architecture reduces the multidimensional problem illustrated in Fig. 1 to a single dimension by using a common object model for describing all resources to be managed. HP OpenView NMA uses the concepts defined by the OSI standard for describing managed objects. HP OpenView NMA supports the structures of management information used by the OSI model and the Internet Engineering Task Force (IETF). HP OpenView NMA also provides support for managed objects defined in the ISO and IETF management information bases and is expandable to include objects defined by developers for special purposes.



CMIP = Common Management Information Protocol

Fig. 3. *OSI System Management Model.*

Managed objects are an abstraction of the real resources being managed. The management services (M) offered by a managed object consist of the software (programs) necessary to provide the services defined in the managed object specification. This software may contain several parts, such as the infrastructure interface, an object manager, and environmental services (E) necessary for communication with the real resource.

HP OpenView NMA extends the OSI object model by describing managed object data stores. These are identical to managed objects except that they are persistent. In the OSI model, managed objects are volatile. The managed object data stores model a mechanism for maintaining information about managed objects even when networks or systems are powered down.

Distributed Communications Infrastructure

U	S	O
A	P	M
E	C	D

Network management is a distributed activity in that the user interfaces, management applications, and management services can be located in different systems throughout a network. The HP OpenView NMA communications infrastructure (Fig. 5) provides the facilities for establishing and maintaining communication between these components. The communications infrastructure consists of the postmaster (P), the supervisor (S), and communications profiles (C). These components draw heavily upon the environmental services (E) to carry out their functions.

The postmaster (P) provides basic message routing services between the network management components listed above. It operates as a table-based object-oriented message router. The routing table shown in Fig. 5 is the focal point of the postmaster's functionality and is itself a managed object. Given a message (perhaps from an application) addressed to a specific managed object, the postmaster looks up the managed object name in the routing table to find the information necessary to deliver the message to the managed object. The fields in the postmaster's routing table provide the managed object name, the communication profile number, and profile-specific data.

The information required to perform the name-to-address translation can be obtained from a directory service. The postmaster can make use of the directory service through the supervisor, which has the responsibility for creating and maintaining the routing table. However, the postmaster must be able to operate independently in case these supporting services fail. The routing table can be considered a cache of information derived from directory services and other sources.

The supervisor (S) administers the existence of and access to all the components associated with the communications infrastructure. It has the authority and the ability to initiate, cancel, lock and unlock, and control access to management services within its supervisory domain.

The HP OpenView NMA process of exchanging management information is adopted from the OSI framework. An application layer network management protocol supports transaction-oriented exchanges between the distributed processes. HP OpenView NMA supports formal and de facto industry-standard protocols with a common network management communications application program interface based on the OSI Common Management Information Services (CMIS).[7] In an OSI environment, CMIP (as shown in Fig. 5) would be the network management protocol of choice. In an Internet TCP/IP environment, the most widely used protocols are the Simple Network Management Protocol (SNMP, specified in RFC 1098) and CMIP over TCP/IP (CMOT, specified in RFC 1095). HP OpenView NMA supports not only these network management protocols but also the addition of proprietary protocols under the common application program interface for complete integration in a multivendor, heterogenous network. Each protocol stack is modeled as a communications profile managed object; this allows extensibility of the communications infrastructure illustrated in Fig. 5.

The environmental services (E) represent the facilities provided by the environment in which the network management solution must operate (e.g., HP-UX, MPE, MS-DOS). The capabilities provided by these environments can be used by any of the components of HP OpenView NMA. In this context, these services could include the file system, the X.500 naming system, or proprietary network management protocols. It is also possible that some of these services could become part of a managed object. For example, a managed object that is responsible for report generation and delivery service would collate the information to be presented in the report, create a file, and deliver it to any requesting application using a file transfer environmental service such as FTAM (File, Transfer, Access, and Management).

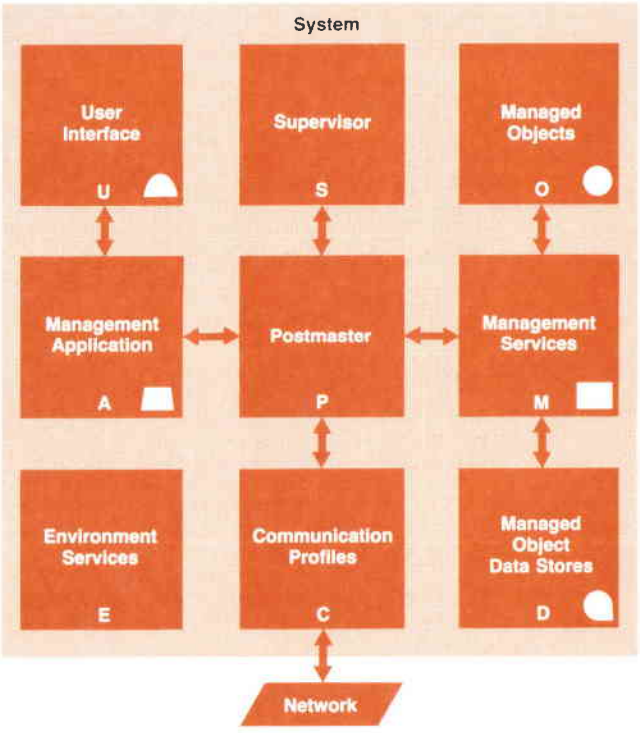


Fig. 4. *Components of the HP OpenView network management architecture reference model.*

Multiple User Interfaces

U	S	O
A	P	M
E	C	D

HP OpenView NMA specifies the user interface as a separate component of the network management solution. This addresses the need to allow users to access network management solutions from a variety of devices, which may include dumb terminals, workstations with graphic displays, voice, or any appropriate devices. It also addresses the need to be able to build solutions that meet individual users' requirements so that only relevant information is provided.

The separation of the user interface from the management application provides an additional point of integration so that several management applications can drive a single display or several displays can be driven by a single management application. User interfaces can be distributed throughout the network using environmental services (E) to access remote management applications (A).

Extension of the Object-Oriented Paradigm

U	S	O
A	P	M
E	C	D

HP OpenView NMA extends the power of the object-oriented paradigm as specified in the OSI model by treating portions of application functionality as managed objects. These managed objects are defined using the same object specification techniques described previously. The result of this approach is that value-added application functionality that was previously only available within a given application now becomes available as a source of information or services to other management applications. This encourages the full integration of products into a hierarchy of management solutions while reducing duplication of functionality and many concerns about consistency. Thus, what was previously a monolithic application is decomposed into a much reduced management application with an accompanying managed object that offers management services identical to the value-added functionality that was previously locked within the application.

The additional point of integration provided by HP OpenView NMA between management applications and these managed objects also opens functionality for wider use by more applications. Consequently, HP OpenView NMA facilitates the development of applications upon a base of existing managed objects. For example, a traditional fault application would include event processing capabilities. When a configuration application is to be installed in the same system, it would not normally have access to event processing in the fault application, and would have to duplicate that functionality. HP OpenView NMA encourages the specification of event managers as managed objects, thereby making event management services available to other management applications.

These managed objects can provide services not only to management applications but also to other managed objects. HP OpenView NMA describes a scheme in which the most significant value-added portions of a network management solution are described as managed objects, so they can be linked into management chains, providing comprehensive services to management applications.

Conclusion

It is important to stress that HP OpenView NMA is intended as a reference model for developers of network management products. It is not intended to mandate the implementation of all the components described above. Neither are the components intended to define program or process boundaries. Products implementing HP OpenView NMA may range in complexity from bridges to complete systems. Bridges, for example, may only implement the management services defined for the bridge managed object class along with a specific protocol stack. The remaining components could be distributed on other systems as needed.

Initial HP OpenView products implement portions of HP OpenView NMA. For example, HP OpenView Windows (see article, page 60) provides a rich environment for developing graphic user interfaces customized for network management. Newer HP OpenView products will implement more features of the architecture.

In summary, HP OpenView NMA uses the OSI standards

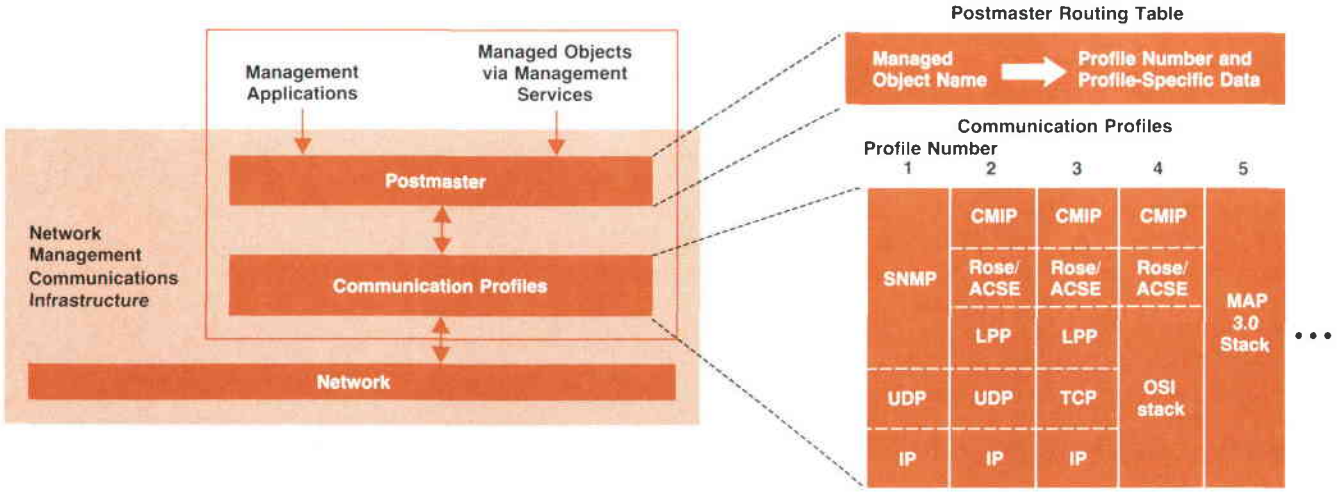


Fig. 5. *Distributed communications infrastructure of the HP OpenView network management architecture.*

to manage multivendor environments. It refines the standards to introduce additional points where integration of products can take place and by doing so reduces duplication of product functionality. It also allows for customization to support specific network management activities by building on an existing base of managed objects.

References

1. *Information Processing Systems - OSI - Basic Reference Model - Part 4: Management Framework*, ISO/IEC 7498-4, 1989.
2. *New Recommendations of the "M" Series*, AP IX-31-E, International Telegraph and Telephone Consultative Committee (CCITT), IXth Plenary Assembly, Document 31, Study Group IV, Report R26, April 1988.
3. *Advanced Networked Systems Architecture (ANSA) Reference Manual*, Rel.00.03, ANSA, June 1987
4. *Information Processing Systems - OSI - Systems Management Overview*, ISO/IEC DP 10040, January 1989.
5. *Information Technology - OSI - Common Management Information Protocol Specification*, ISO/IEC 9596, 1989.
6. *Information Technology - OSI - Structure of Management Information, ISO/IEC DP 10065, 1989.*
7. *Information Technology - OSI Common Management Information Definition*, ISO/IEC 9595, 1989.

HP OpenView Windows: A User Interface for Network Management Solutions

HP Openview Windows provides a consistent graphics-based user interface for users of network management applications, and a set of utilities that enable developers to create network management applications for the HP OpenView Windows environment.

by Catherine J. Smith, Arthur J. Kulakow, and Kathleen L. Gannon

HP OPENVIEW WINDOWS is a graphical user interface based on the Microsoft Windows environment that provides facilities for handling the user interface for network management applications. For application developers, HP OpenView Windows provides programs to carry out tasks such as drawing a network map or handling alarms. From the end user's perspective. HP OpenView Windows combines the functionality of many of the user's network management applications under one easy-to-use interface, simplifying the learning curve.

This article describes the features provided by HP OpenView Windows to developers and users. Some of the other HP OpenView articles in this issue describe the details of interfacing to HP OpenView Windows.

Overview

HP OpenView Windows consists of three programs: OVDraw, OVRun, and OVAdmin. These three programs provide the following functionality:

- OVDraw. This program allows users to create maps made up of one or more pictures that represent a data network.
- OVRun. This program provides the facilities that allow users to monitor, diagnose, and control their networks. OVRun uses the maps created with OVDraw.
- OVAdmin. This program is used to set operating characteristics for HP OpenView Windows and HP OpenView applications. These include functions such as assigning passwords and setting up network management parameters.

The graphical user interface consists of maps, pictures, and symbols used to represent a network. A map depicts the whole network and is made up of pictures, each of which shows a portion of the network. For example, a map may represent a network for a group of buildings, while each picture of the map shows the network for one of the buildings. Pictures are composed of symbols that portray

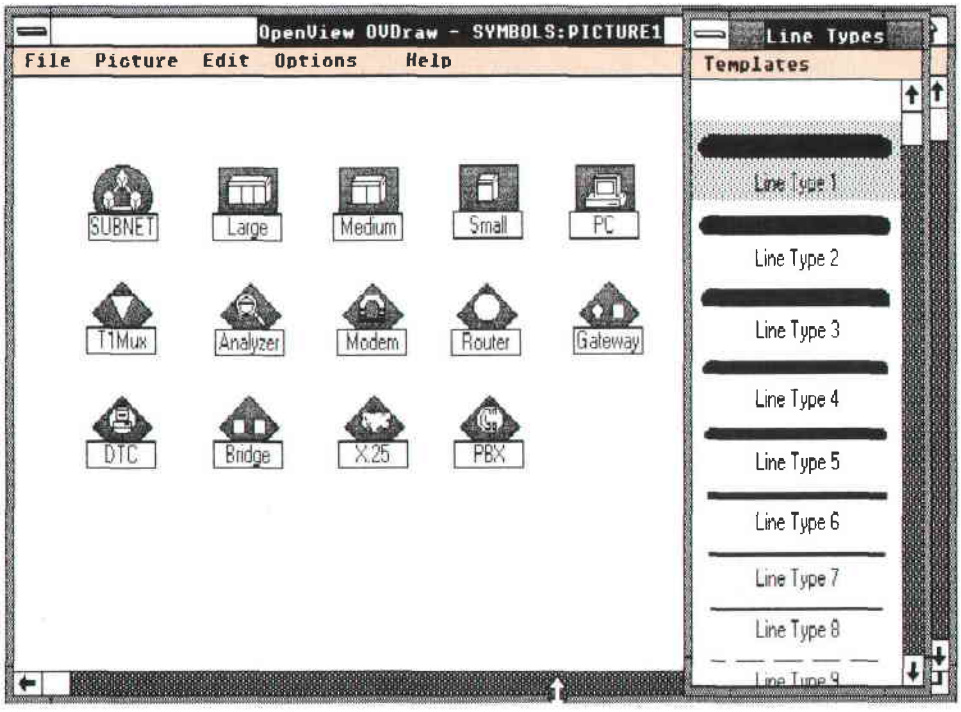


Fig. 1. *Some of the symbols used by HP OpenView Windows to represent network components that are managed by network management applications.*

network components, such as computers, modems, bridges, X.25 switches, lines, or a portion of a network (subnet). The symbols provided by HP OpenView Windows are shown in Fig. 1. Subnet symbols are special symbols that are used to signify what is contained in another picture of the network and to help the user navigate around the network. As will be shown later, subnet symbols can be used to represent the network configurations in different locations (e.g., different buildings).

Symbols also provide configuration and status information. Network topology information is provided by the way in which symbols depict connections between components in a network. Status information is provided by the colors of the symbols, which represent the condition or state of each device. Network status is discussed later in the article.

The user gives instructions to HP OpenView Windows and HP OpenView applications via a combination of symbols, menus, and dialog boxes. Symbols represent the network components described above, and they are used to select the network component the user wishes to work on. Menus and menu items represent the operations the user can select to perform network management functions. Menus and menu items may be standard HP OpenView menus or menus added by an application. Dialog boxes allow the user to give instructions to an application. The overall structure and capabilities of dialog boxes are provided by Microsoft Windows. The content of a particular dialog box is provided by the application.

HP OpenView Windows provides a large part of the user interface for applications, but applications must provide part of the user interface. Applications can add to the HP OpenView Windows user interface by creating new menus and adding new menu items, and by adding dialog boxes. The articles on pages 66, 71, and 85 describe adding menus and dialog boxes to HP OpenView Windows.

The HP OpenView Windows product is divided into two categories: end user products and application developer products. The end user products contain the hardware and/or software components required to use HP OpenView Windows. The software component includes the three programs OVRun, OVAdmin, OVDraw, and one or more HP OpenView applications. The recommended hardware configuration consists of the HP Vectra model ES/12 personal computer with a 40-megabyte hard disk and a 2M-byte expanded memory board. For developers the key product is the HP OpenView Windows Developer's Kit, which contains the HP OpenView Windows end user software and the pieces needed to develop HP OpenView Windows applications.

Application Installation

When the HP OpenView Windows software is installed on the Vectra, it creates two sections in the Microsoft Windows WIN.INI file. The two sections are called OpenView and OpenViewApps. The OpenView section contains information such as the default network map and the name of the file used for logging. The OpenViewApps section contains entries for HP OpenView applications, which are filled in when applications are installed. An entry in the OpenViewApps section is in the following format:

```
[OpenViewApps]
AppName = AppRun.Exe,AppDraw.Exe,AppAdmin.Exe
```

AppName is the application name, AppRun.Exe is an executable file to be started when the end user runs OVRun, AppDraw.Exe is an executable file that is started when OVDraw is run, and AppAdmin.Exe is the executable file started when OVAdmin is run. Applications don't need to have executables for all three HP OpenView programs.

When an HP OpenView Windows application is installed, it is registered with HP OpenView Windows through the entry in the WIN.INI file. HP OpenView Windows applications also register for graphic symbols and menu items. Registration for symbols (or objects in Microsoft Windows terminology) and menus is accomplished by calls to HP OpenView Windows intrinsics.

When one of the HP OpenView Windows programs (OVRun, OVDraw, or OVAdmin) is started, HP OpenView Windows checks the WIN.INI file and invokes all the HP OpenView applications installed there. When an application is invoked, its WinMain loop is entered and the first HP OpenView Windows intrinsic called is OVInit( ), which sets up communications between the application and HP OpenView Windows. The application then calls OVRegister( ) to inform HP OpenView Windows what object types (symbols) the application manages, and OVMenuAdd( ) and OVMenuAddItem( ) to inform HP OpenView Windows which menus and menu items are valid for the given object types. The application informs HP OpenView Windows that the initialization is done by calling OVInitComplete( ).

Four main facilities are provided by HP OpenView Windows: map handling, menu integration, status, and context sensitive help.

Map Handling

When we started considering the requirements for a user interface for network management applications, it was obvious there was a need for a graphics-based user interface. Typically the way network managers and operators use a network management application is to draw a picture of the network on paper, annotating the drawing with identification and other information. The data from the drawing is used to tell the network management application which network components to manage.

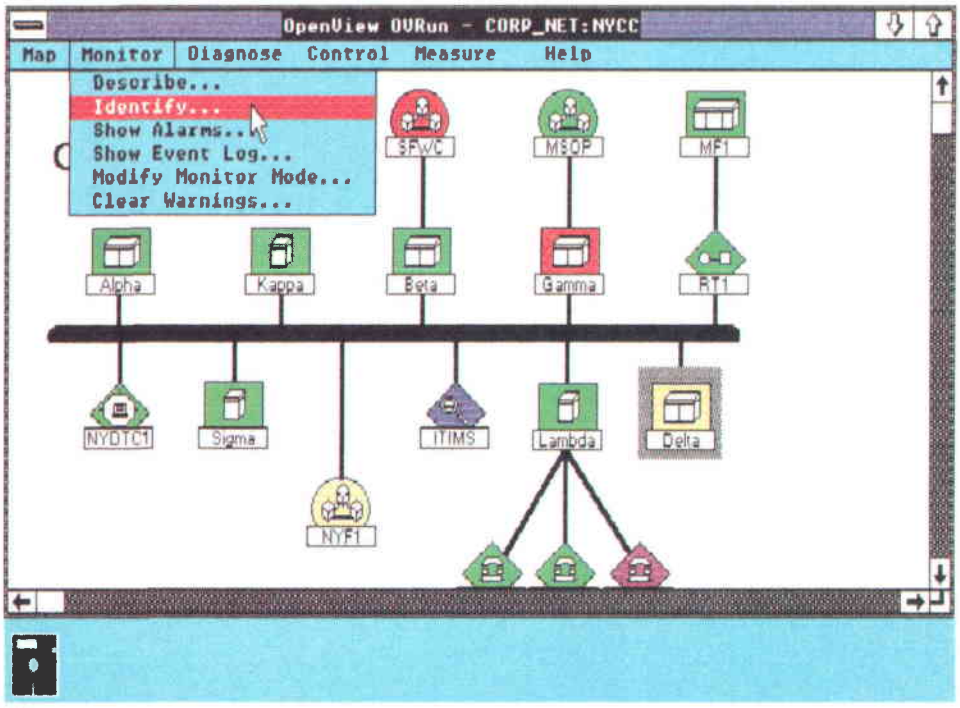
HP OpenView Windows simplifies this task by supplying the network map composed of graphics symbols that represent the network components. Associated with each symbol is a network management application that is invoked when the particular symbol and a menu item are selected. Thus, the network manager no longer needs to run an application and then identify the network component of interest. Also, when a network manager is monitoring a network and the state of some network component changes, it is much faster for the network operator to identify the node having problems by looking at a network topology rather than having to look up a node name or address.

In Fig. 2 the user wants to identify a computer system Delta (i.e., get details on the machine type, version number, etc.), and so selects the computer symbol labeled Delta and the Identify menu item under the monitor menu. After the user selects* the Identify menu item, HP OpenView Windows

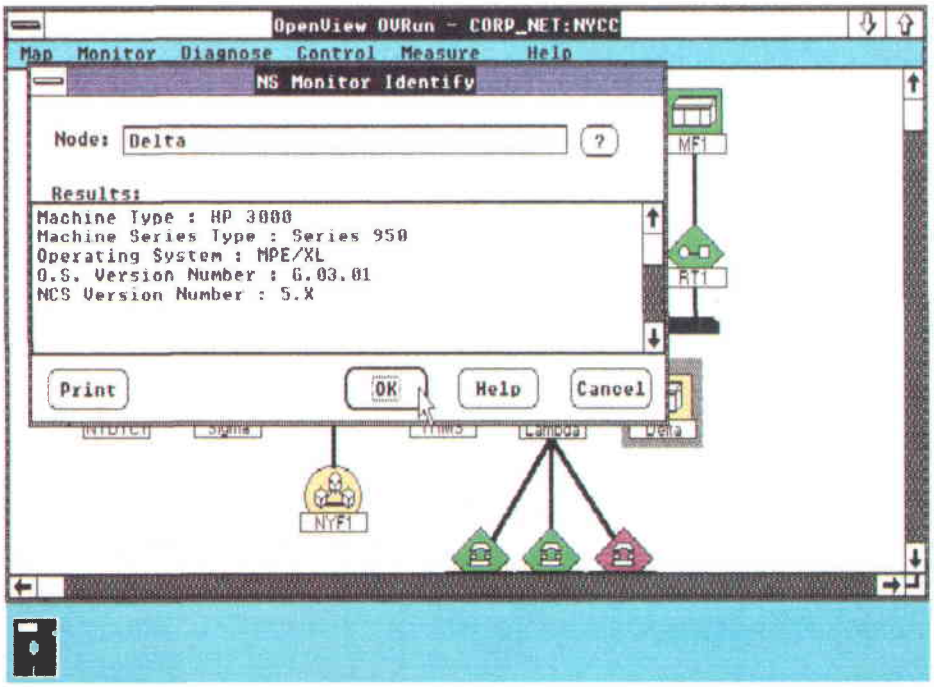
*Selection is accomplished by clicking the left mouse button on a symbol or menu item.

passes the information to the application registered for that symbol type/menu item pair. The application then has control, and can use HP OpenView Windows routines to access the device or stored data to accomplish the actions associated with Identify. In the example in Fig. 2, the application registered for the Delta computer system is NS Monitor, which is listed at the top of the dialog box in Fig. 2b. If the user wanted to identify a bridge, a bridge symbol and then the Identify menu item would be selected. The article on page 66 describes how the HP OpenView BridgeManager implements the Identify function.

The map can be organized in any way the customer wants to view the network: physical, geographical or organizational. For example, suppose a company has four computer systems—two in building 1 and two in building 2, and the two sites are connected via a LAN bridge. The user can draw a hierarchical map (Fig. 3) or a network model map (Fig. 4). These examples also demonstrate that the user is not limited to placing a symbol in only one picture. In Fig. 4 the subnet symbols BLDG1B and BLDG2B and the bridge symbol BRIDGB show up in more than one display. BRIDGB represents the same physical bridge each time it appears.



(a)



(b)

Fig. 2. *(a) The user selects the computer symbol labeled* Delta *and the* Identify *menu item. (b) This symbol/menu item combination provides the user with some identification information about the computer system* Delta.

The map functionality in HP OpenView Windows allows the user to navigate through the map in many different ways. The user can double click on a subnet symbol to display the network that the subnet symbol represents, or the user can navigate through the map by using the following menu items contained in the map menu.

- Go To Top allows the user to view the top picture of any network currently being viewed.
- Go To Previous allows the user to view the previous picture displayed.
- Go To Picture allows the user to view a picture listed in a dialog box. By clicking on one of the items in the box and then clicking OK on one of the the pictures listed, the selected item is displayed.
- Go To allows the user to find pictures containing the symbol for a selected node or device. A dialog box will be displayed listing all the pictures containing the symbol if the symbol is in more than one picture.

Based on feedback from developers, two features were added to the map handling facility: treating lines as symbols and allowing multiple applications to register for the same symbol.

Lines. In the early versions of HP OpenView Windows, many different line types were defined, but they couldn't be managed like other network symbols. We discovered that many network management applications wanted to be able to manage lines. This was especially true of HP's telecommunications divisions. Where the systems divisions thought in terms of boxes, the telecommunications divisions thought in terms of lines.

Since we wanted the user interface to be useful to all network management applications, lines were made equal with other network component symbols. Lines can change color to reflect the status of the line. They can be named, registered for, selected, and managed. Lines can also be labeled and split into different internal types. For example, obj_line can be split into line types obj_line1, obj_line2, and so on. This feature allows multiple applications that register for lines to coexist together. Without this feature, two applications that want to manage lines and use some of the same menu items would not be able to run in the same HP OpenView Windows together. The article on page 71 provides an example of the use of HP OpenView Windows line types for network management.

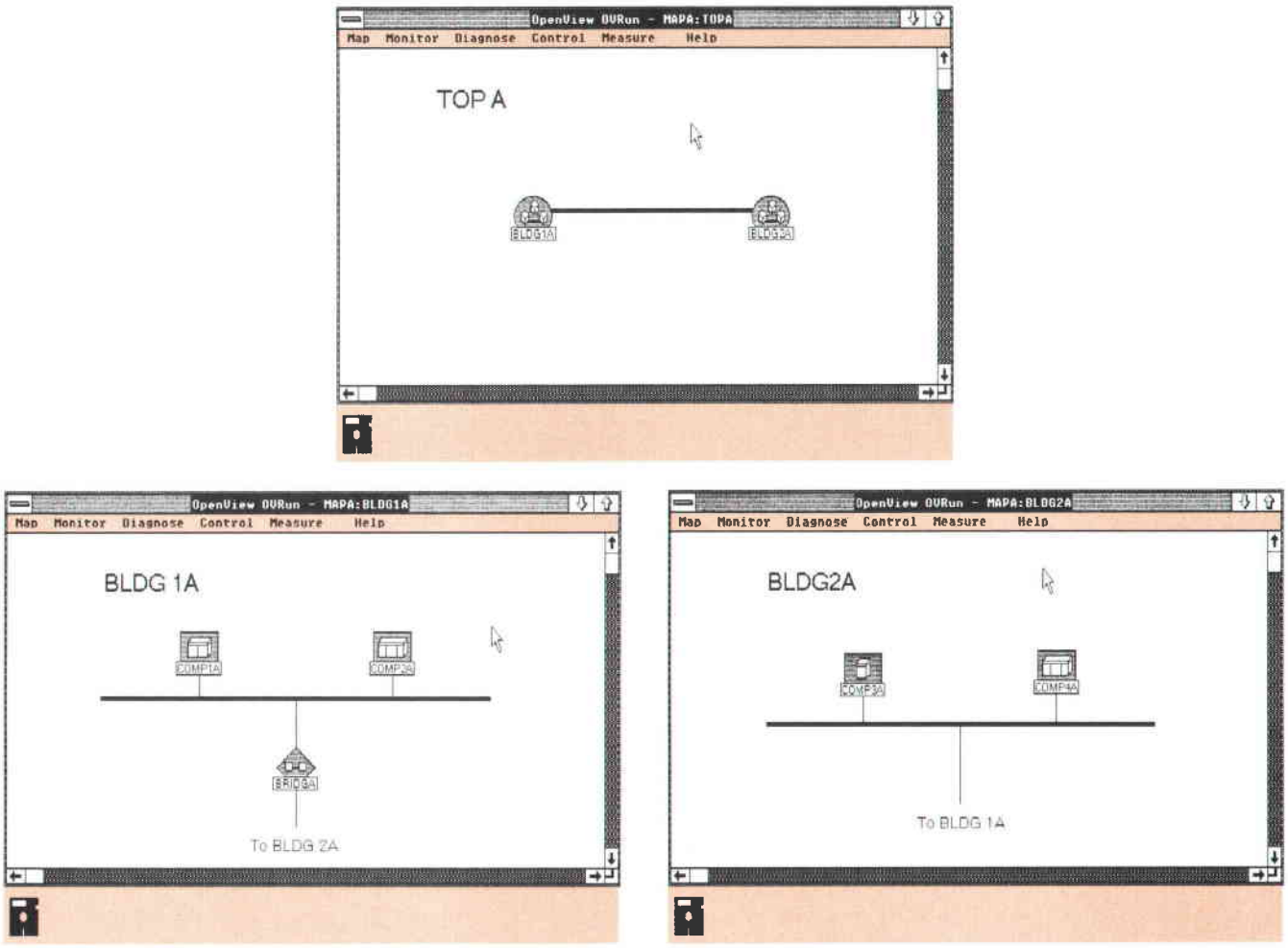


Fig. 3. *Hierarchical network map. Note that the subnet symbol is used to depict the existence of network configurations in different buildings.*

Multiple Symbol Registration. Since more than one application can run at a time, more than one application may want to register for the same symbol. This is especially true for applications that manage system symbols like HP DeskManager, SNA, and TCP/IP. Therefore, we allow multiple applications to register for (and to manage) the same symbol with the limitation that they cannot register for the same menu items. This means that if application A is registered for the menu item/symbol combination Identify/DTC, application B cannot cannot register for this same combination, but it can register for another menu item/DTC combination. This limitation ensures that HP OpenView Windows knows to which application to send the selection message. Allowing multiple applications to register for one symbol also works well for integrating foreign applications that need very different functionality. For example, an application that manages a computer system can run together with a terminal emulator connected to that system. When the user selects the computer system, the menu items for both the system management application and the terminal emulator can be enabled.

Menu Integration

Customers buy network management solutions to gain increased use of network resources and lower the cost of maintaining networks. Lower maintenance costs come through minimizing the time required to train operators and managers. Making the user interface as easy to use as possible is one way of lowering the training costs. Another way of lowering training costs is through the use of a consistent user interface across all network management applications. This means that if a user is tracking down a network problem, there is no need to switch back and forth between different tools with different user interfaces to accomplish the job. There are a wide variety of devices in a network that need to be diagnosed or configured, and if a user has to learn many different user interfaces for each device and type of activity performed, training becomes a significant cost to the customer.

To help ensure a consistent user interface, HP OpenView Windows provides three types of menu items: standard HP OpenView Windows menu items, application dependent menu items, and application-added menu items. Standard

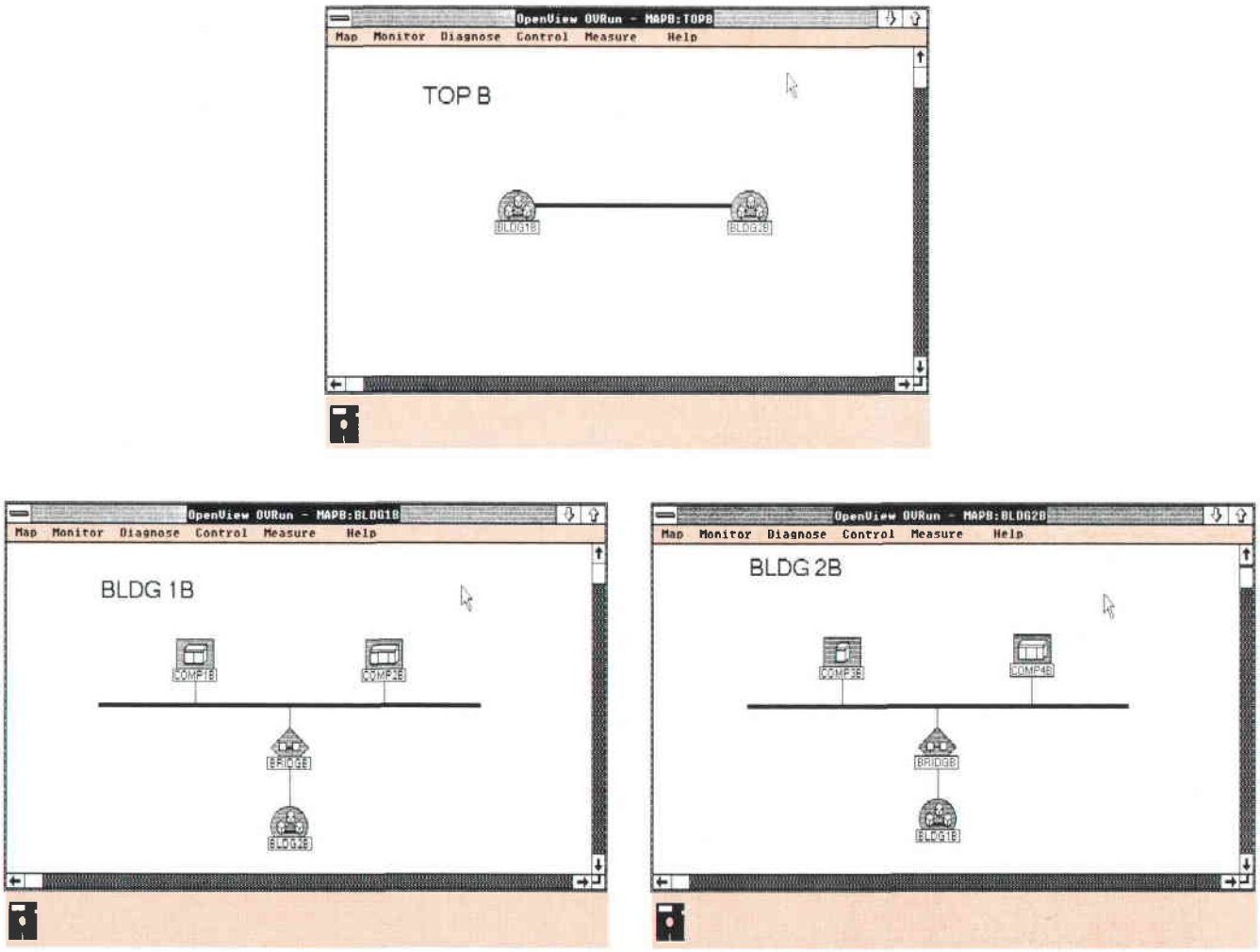


Fig. 4. *Network model map. In this representation the subnet symbol for the buildings is shown in each picture.*

HP OpenView Windows menu items appear regardless of what applications are being used. The functionality of standard HP OpenView Windows menu items is always supplied by HP OpenView Windows, not by the application. The OVRun map menu items such as Go To Top and Go To Previous are examples of standard menu items.

Application dependent menu items are displayed by HP OpenView Windows but rely on applications to provide functionality. They appear on the menu regardless of what applications have been installed. The Indentify menu item described earlier is an example of an application dependent menu item. As shown in Fig. 2 the functionality for Identify in this example is provided by the application NS Monitor.

Application-added menu items are added by an application, and appear only when that application has been installed.

Menu items may be object-specific or nonobject-specific. Object-specific items remain disabled (greyed out) until the user selects a symbol in the map. If there is an application that provides the functionality for that type of symbol, it is enabled. An example of an object-specific menu item is the Identify menu item, which is enabled only if an object to be identified has been selected.

Nonobject-specific items are always enabled and can only be handled by one application. An example of a nonobject-specific menu item is the Show Alarms menu item. It is always enabled since the alarm list doesn't refer to any specific object.

Status

A graphical user interface allows the network operator to gain a large amount of information about a network from the components on the display. The use of color helps the operator absorb this information quickly. Color representing the state of network elements is a key part of HP OpenView Windows.

When HP OpenView Windows is initialized the status of all of the devices is unknown. This state is represented by the color blue. Once the applications start coming up and informing HP OpenView Windows of the states of the devices, the colors are changed to represent the true state of the devices. Red is used for critical, yellow for warning, green for OK/normal, and magenta for an informational state. The informational state might be used to indicate that the device is off-line or under test, or has messages queued that don't represent a warning condition. If a device changes to a critical or warning state, in addition to changing the color of the symbol, a warning message is displayed. Since lines are treated the same as symbols, their colors represent the same state information. The only difference is that the initial color for lines is black.

Since HP OpenView Windows allows the user to have many different pictures representing different parts of a network, it is possible that the user may not be viewing the picture that contains a device whose status has just changed because the device may be at another level of the network hierarchy or on another network. One of the features of the HP OpenView Windows map is status propagation. This means if a symbol in a picture changes state (color), all subnet symbols representing that picture will have their color changed to represent the highest severity contained in the picture. In the map examples shown in Figs. 3 and 4, if COMP1A had a critical error, the symbol COMP1A in the BLDG1A picture would be red. The subnet symbol BLDG1A in picture TOPA would also turn red, indicating to the user that at least one symbol in picture BLDG1A had a critical severity. Because the user may have multiple levels of subnets, status information must be propagated up through multiple levels. If the user had drawn the network in a network model map like the one in Fig. 4, this would mean that every subnet would be red. If COMP1B were critical (red), BLDG1B would be red because it contains COMP1B, and BLDG2B would be red because it contains subnet symbol BLDG1B. In HP OpenView Windows, the user can configure the map to propagate status up one level or all levels.

Help Facility

The last area where HP OpenView Windows provides integration of applications is in the area of context sensitive help. The NewWave help facility[1] is used to allow applications to format and integrate help text. The user is able to access the help facility by either pulling down the Help menu or by clicking on the Help button within a dialog box. The Help pull-down menu has menu items for HP OpenView Windows and for each HP OpenView Windows application installed. Selecting a menu item under the Help menu brings up the index for HP OpenView Windows or the HP OpenView Windows application. Clicking on the Help button within a dialog box brings up the help screen for that dialog box.

Conclusion

HP OpenView Windows is a tool for both network management application developers and end users. For developers, HP OpenView Windows simplifies the task of developing a graphical user interface for network management applications by providing functionality in the areas of map handling, menu integration, status, and help. End users also benefit from HP OpenView Windows, since the resulting applications have a consistent user interface that is easy to learn and use.

References

1. V. Spilman and E.J. Wong, "The HP NewWave Environment Help Facility," *Hewlett-Packard Journal*, Vol. 40, no. 4, August 1989, pp. 43-47.

HP OpenView BridgeManager: Network Management for HP LAN Bridges

Since LAN bridges receive all the data packets transmitted on the LAN segments they interconnect, they are an ideal focal point for monitoring packet integrity and the number of packets forwarded and filtered.

by Andrew S. Fraley and Tamra I. Perez

AS LOCAL AREA NETWORKS (LANs) grow to the limits of their physical and electrical specifications, traffic levels can reach a point where throughput is significantly impaired. A LAN bridge logically separates segments of a very large LAN so that optimum throughput is maintained. A bridge allows intersegment communication only as required. For example, if two nodes on the same LAN segment are communicating, there is no need for their packets to be forwarded to other LAN segments (see Fig. 1). Thus, a bridge restricts traffic to only the necessary LAN segments.

To separate LAN segments logically, a bridge monitors traffic on the network and builds an internal address table. In essence, the bridge learns the MAC addresses and port locations of nodes communicating on the LAN, and only forwards traffic to other LAN segments if source and destination nodes are on different bridge ports.

Recently, an IEEE 802.1 committee defined a standard spanning tree algorithm[1] that allows redundant bridging to increase LAN reliability (see Fig. 2). A spanning tree algorithm is typically used to determine the shortest path between any two LAN nodes. The spanning tree standard adds the capability for two bridges to interconnect the same LAN segments and protect LAN operation in the event of a bridge failure. The spanning tree standard places one bridge in active bridging mode for forwarding packets and the other bridge in backup state for monitoring traffic and maintaining its address table. It is important to note that connecting two bridges that do not support the spanning tree standard could result in an infinite cycle of packets between redundant bridges. This would create an increase in the traffic levels on adjacent LAN segments, bringing down both segments.

Hewlett-Packard builds a number of two-port LAN bridges. These bridges interconnect IEEE 802.3 10-Mbit/s networks, IEEE 802.3 1-Mbit/s networks, and IEEE 802.5 token ring networks. The HP products and the networks they support are given in Table I. The 10-to-10 and 10-to-1 bridges that bridge IEEE 802.3 10-Mbit/s and 1-Mbit/s networks support the spanning tree algorithm and network management.

Table I
HP LAN Bridges

HP LAN Bridge	Network Connection	Network Management
HP 28647B Star-LAN Bridge	802.3 10-Mbit/s to 802.3 1-Mbit/s	Yes
HP 28648B LAN Bridge	802.3 10-Mbit/s to 802.3 10 Mbit/s	Yes
HP 28649A Token Ring Bridge	802.3 10-Mbit/s to 802.5 4-Mbit/s	No

Why Manage a Bridge?

There are two reasons to manage a bridge: to monitor the network and to configure the bridge. Because bridges receive all the data packets on the segments they connect, they are an ideal focal point for monitoring packet integrity and the number of packets forwarded and filtered. This

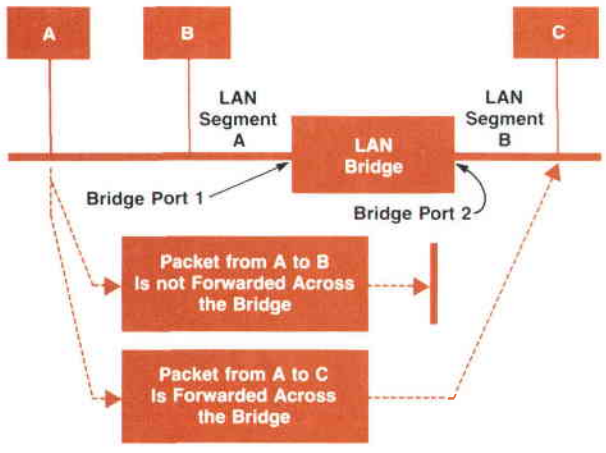


Fig. 1. *How a LAN bridge isolates traffic.*

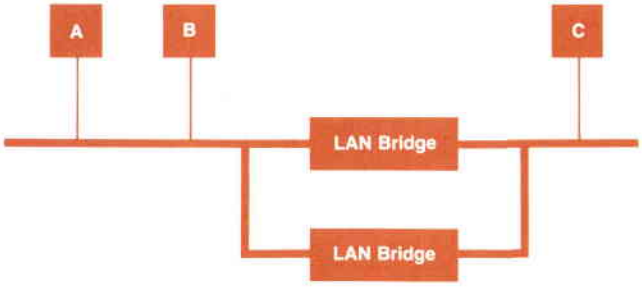


Fig. 2. *Redundant bridge configuration.*

information can be used to tune the network and isolate problems. For example, if a bridge port sees sustained traffic levels over twenty-five percent, this indicates that the segment may have too many nodes. Any subset in which the nodes generate a lot of traffic communicating among themselves should be broken into a separate segments and isolated by a bridge. For another example, if a new node was added yesterday, and today the bridge port to which it is attached is reporting a high level of CRC errors, this indicates a problem with the transmitter on the new node.

Bridge management can also be used to customize the configuration of the bridge's operating parameters. For example, if there is a set of nodes containing sensitive data that should be accessed only by a handful of privileged users, these sensitive nodes can be placed on a private segment and isolated by a bridge. The bridge's address table can be configured to forward only packets from the sensitive nodes and from the handful of privileged nodes. Consider a LAN that has grown so large that the worst-case forwarding time between two bridges exceeds a second. In this case, it might be necessary to adjust the spanning tree algorithm time-outs upward for optimal spanning tree performance.

HP OpenView BridgeManager

The HP OpenView BridgeManager is an HP Vectra computer-based HP OpenView application that manages HP's 10-to-10 and 10-to-1 LAN bridges. The BridgeManager provides the ability to poll bridges, read parameters, set parameters, upload and download complete configurations, log on and log off, log counters, and monitor alarms. The BridgeManager also supports the HP NewWave help system, which has been integrated into the HP OpenView product.

The BridgeManager is divided into two parts: the user interface and the network interface. The user interface interacts with the HP OpenView system and Microsoft Windows. The network interface manages the communication with the LAN bridges.

User Interface

The BridgeManager user interface provides a graphical interface based on a network map consistent with other HP OpenView applications. About half of the BridgeManager user interface code provides links to Microsoft Windows, HP OpenView Windows, and the BridgeManager network interface (see Fig. 3). The other half of the user interface code implements a table-driven formatter that formats packets received from a bridge and parses user input into packets sent to a bridge.

The parsing and formatting functions were implemented in a central formatting mechanism for two main reasons: to reduce code size and to ensure flexibility. In Microsoft Windows, segments can be discarded or swapped from system memory and new segments loaded from disk. When Microsoft Windows runs low on system memory, this swapping or discarding begins to occur and performance degrades. Therefore, it is imperative to keep the size and number of code segments small. The formatting mechanism meets the code size goal by trading off code size for data size. Instead of hard-coding formatting statements for each

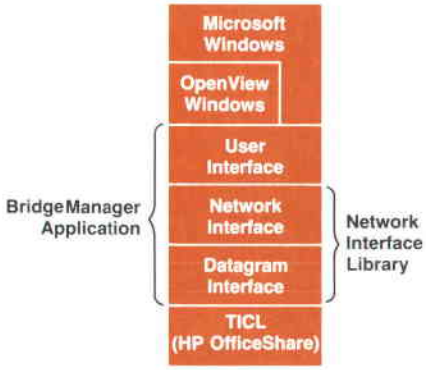


Fig. 3. *Layers of software involved in the HP OpenView BridgeManager application.*

packet type, the formatting directives are described in a data table and a shared formatter interprets the table.

The network management protocol used by the BridgeManager is loosely based on an HP proprietary network management protocol that was targeted for use in HP's network management products until standard network protocols became available.

Throughout the project, the BridgeManager protocol changed to add or delete functionality. The BridgeManager user interface was designed to minimize the time required to adapt to a changing protocol. The formatter achieves this flexibility goal because if the packet format is changed, no code is affected. Only the table must be modified to describe how to format and parse the modified packet.

The formatting table is an array of field descriptors. A field descriptor contains information describing the field type (word, string, MAC address, etc.), the default, minimum, and maximum values for the field, the offset of the field in the result string, a pointer to the location of the field in the incoming packet buffer, and a flag that is TRUE if the field is the last field in the string being created (each string may have multiple fields). The following fragment of a real formatting table describes the formatting of the two lines:

IEEE802 Link Address	080009-0033C4
Product Number	HP 28647B

contained in the list box of the BridgeManager Identify dialog box shown in Fig. 4.

```
struct    {

        int field_type;              /* Flag that determines how the   */
                                     /* formatter handles the data     */
                                     /* identified by the packet       */
                                     /* location field                 */
        char *min_max_default;       /* Pointer to a string containing */
                                     /* information describing the     */
                                     /* bounds of the data field       */
        int field_offset;            /* The character position of the  */
                                     /* formatted field in the output  */
                                     /* line                           */
        char *packet_location;       /* A pointer to data in an incoming */
                                     /* packet or an identifier        */
                                     /* of a resource string           */
        int last_field;              /* A Boolean variable that is true */
                                     /* if the current field is the last */
                                     /* field in the output line       */
}
        counters = {

        FORM_RESSTRING,              /* String from string table       */
        NULL,                        /* No min/max information         */
        0,                           /* Field at beginning of line     */
        (char*) STRING_IEEE_802_LINK_ADDRESS,        /*ID of */
                                     /*string table string            */
        FALSE,                       /* Not the last string in line    */

        FORM_ADDRESS,                /* MAC address                    */
        NULL,                        /* No min/max information         */
        23,                          /* Position 23 in line            */
        (char *)packet.ieee_802_source_addr,     /* Data position */
                                     /*in packet                       */
        TRUE,                        /* Last field in line             */

        FORM_RESSTRING,              /* String from string table       */
        NULL,                        /* No min/max information         */
        0,                           /* Field at beginning of line     */
        (char *)STRING_PRODUCT_NUMBER,       /* ID of string */
                                     /*table string                    */
        FALSE,                       /* Not last field in line         */

        FORM_STRING,                 /* String from packet             */
        "20,"                        /* Max length of string is 20     */
        23,                          /* Position 23 in line            */

        (char *)packet.product_number,   /* Data position in packet */
        TRUE                         /* Last field in line             */
};
```

Consider how this table is used to interpret an identify packet. The first field is of type FORM_RESSTRING. Knowing the field is a resource string, the formatter interprets the packet_location pointer not as a pointer into a packet, but as a constant identifying which string is to be loaded from a Microsoft Windows resource string table. This string is loaded at offset 0 in the result string. Because the last_field flag is FALSE, the formatter realizes that the resulting string is not yet complete. The next field is of type FORM_ADDRESS. The formatter interprets the packet_location pointer as a pointer to a MAC address and formats the packet data into the string at offset 23 in the result string. Because the last_field flag is TRUE, trailing spaces in the result string are trimmed and the string is output by the formatter to a list box or file (as directed by the code that invoked the formatter). The next field is another field of type FORM_RESSTRING. It is handled exactly like the first resource string field. The last field is of type FORM_STRING. The formatter treats the packet_location pointer as a pointer to a string. This is the only field in the example whose min_max_default string pointer is not NULL. The "20" is scanned from the string and the formatter ensures that the string in the incoming packet is 20 characters or less. If longer than 20 characters, the string is truncated. The last_field flag is TRUE so the second result string is trimmed and sent to a list box or file. The two result strings produce the two lines shown above.

When the user wants to change parameters on a bridge, the formatter also takes strings that have been modified through interaction with the user and parses them into outbound packets. The rest of the user interface deals with the network interface to send and receive packets, to interact with Microsoft Windows objects such as list boxes, edit fields, and pushbuttons, and to interface to the network map and the HP OpenView C-tree data base.

To illustrate how the BridgeManager user interface interacts with these three entities to implement a function, consider the Identify function. When the user wishes to read identity information from a bridge, the user selects the bridge from which to read in the network map and then selects the Identify menu item in the Monitor menu. When the HP OpenView system detects that a bridge is selected, it sends a message to the BridgeManager's message queue indicating that the Identify menu item was activated. The BridgeManager responds to this message by calling Microsoft Windows to create and display the Identify dialog box.

Each BridgeManager dialog box has an associated dialog box function that handles messages for that type of dialog box. When the Identify dialog box is displayed, Microsoft Windows sends an initialization message to the dialog box function. When the Identify dialog box function receives the initialization message, it loads the selected bridge's text label and MAC address into the window's Bridge Label and Bridge Address fields shown in Fig. 4. To do this, the BridgeManager uses HP OpenView function calls to get an object identifier specifying which bridge was selected. The BridgeManager then uses this identifier to look up the bridge's label and MAC address by calling an HP OpenView function. The dialog box function then retrieves an identify packet from the selected bridge using the network interface. When the packet is returned, the formatter formats the information into the list box for the user to view.

Network Interface

The most important function of the BridgeManager's network interface is to manage the communications exchange between bridges and the management node. A management node is the system from which a network manager uses the BridgeManager application to monitor a network. A large part of this functionality is devoted to the maintenance of incoming packets.

Before beginning a discussion of internal packet manage-

ment, it is helpful to understand the architecture and general operation of the BridgeManager network communication process. As shown in Fig. 3, the BridgeManager network interface resides in several layers of code. HP OfficeShare is HP's network transport software. The transport interface compatibility layer (TICL) in the HP OfficeShare software provides access to the network hardware via interrupts. The datagram interface running on top of the HP OfficeShare software limits access to TICL as a protection mechanism. To send a packet across the network, the BridgeManager network interface makes a call to the datagram interface, which filters down to a hardware transmission request. When the network hardware receives a packet, it interrupts TICL. TICL then interrupts the upper layers of software until the packet reaches the destination application, which in this case is the BridgeManager.

Recall from the discussion of the user interface that the Microsoft Windows memory manager dynamically swaps code and data segments. Interrupts from the network hardware would fail if the network interface code and data segments were allowed to move dynamically in memory. Therefore, the network interface is implemented as a static Microsoft Windows library that is separate from the user interface. The user interface must poll the network interface to receive any packets that may have arrived asynchronously. When the network interface receives a poll request and has a packet available for the user interface, it copies the packet from its buffer structure into the pointer provided by the user interface. The network interface then frees the packet buffer to accept another packet.

Two types of packets are received at the BridgeManager network interface: a response to a previously posted request packet (hereafter known as a response packet) or an event notification packet. Event notification packets arrive asynchronously. They are sent when the bridge detects a change in the network state that might be of concern to the user. Event notification and response packets coexist in the network interface buffer structure. Therefore, an array is used to maintain four maximum-size IEEE 802.3 packets. Two of the array elements are used for response packets and the remainder for event notifications. The following is a code fragment containing the packet buffer declaration.

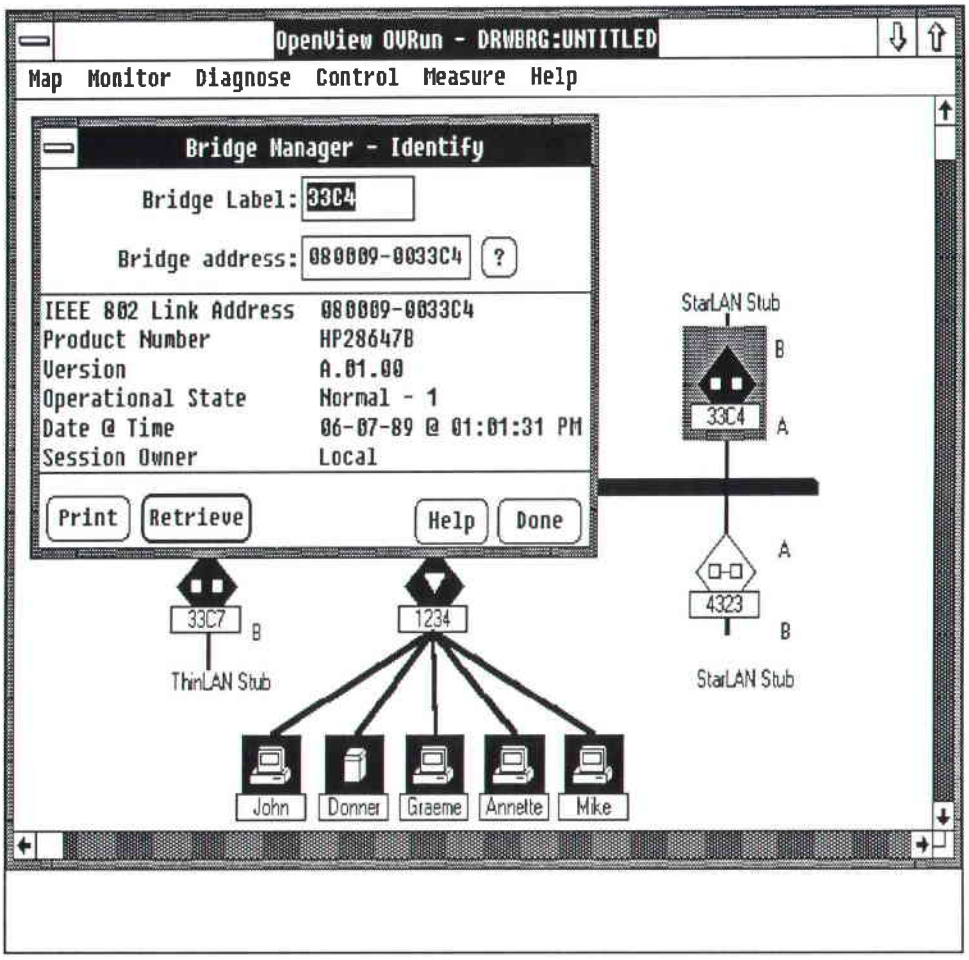


Fig. 4. *BridgeManager* Identify *dialog box.*

```
struct {
        int command_id;          /* TICL identifier of receive          */
                                 /* command posted to this buffer       */
        int buffer_state;        /* Per buffer flag which can take on   */
                                 /* values WAITING, EVENT, or           */
                                    RESPONSE                            */

        char packet_buffer       /* Receive packet buffer               */
          [1518]
}       receive_buffers[4];      /* Allocate four receive buffer        */
                                 /* structures (allowing applications   */
                                 /* to buffer up to four back-to-back   */
                                 /* packets)                            */
```

The network interface uses the buffer_state and command_id elements to control the packet buffer structure. The command_id element is returned from TICL and is used in the event that the receive request must be canceled. The buffer_state element is used to determine whether a buffer is waiting to receive a packet and also to maintain the balance of buffer use between event notification and response packets.

It is critical that the incoming packet buffers not be filled with only response packets or only event notification packets. If the network interface packet buffers were filled with response packets, event notification packets would be locked out of the management node, thus crippling the management feedback system. On the other hand, if the packet buffers were filled with event notification packets, the user would not receive responses to management commands issued on the network. To avoid this problem, the interrupt service routine responsible for validating and accepting incoming packets is designed to monitor the state of the packet buffer with every packet arrival. Currently, two packets of either response or event notification type are allowed to coexist in the buffer structure. If two packets of the same type exist in the buffer and a third arrives, it will be thrown out to maintain the buffer balance.

A major design goal of the BridgeManager was that it coexist with other management nodes and be a well-behaved application with respect to CPU and memory use. The network interface library resides in static nonswappable memory. Therefore, a major design goal was to keep the library as small as possible. The code size was not a problem, but the data structures, specifically the packet buffer, were our major concern. During development, we determined that two buffers per packet type were more than sufficient to ensure that the network interface did not become a a performance bottleneck. However, if this condition changes in the future, the number of buffers for each type of packet is easily altered. A recompilation is required to implement the change.

Conclusion

Bridges are important network components that segment local area networks into logical subnets, filter traffic between subnets and the network segments, and add reliability to the network through redundant paths. Since a bridge observes network traffic during operation, adding management to bridges provides valuable network information to the user.

Acknowledgments

Special thanks to project manager Phill Magnuson, teammates Ralph Bean and John Reilly, and the bridge hardware team.

References

1. *Local Area Network Standard - MAC Bridges*, ANSI/IEEE Standard P802.1D, Rev. 6, September 1988.

HP OpenView Data Line Monitor

Monitoring large and complex network configurations is crucial to maintaining the integrity and performance of data communication lines. The HP OpenView Data Line Monitor is a hardware and software solution for monitoring these data communication lines.

by Michael S. Hurst

INFORMATION NETWORKS are becoming larger and more complex and efficient management of these networks is crucial as organizations become more dependent on them. At the heart of any network are the physical data communication links that connect the computers together. For wide area networks these links often consist of point-to-point leased analog lines and it is problems with these lines that are the most common cause of trouble in data communications. Network managers and datacom managers in places such as corporate data centers are therefore increasingly concerned with the performance and integrity of their data communication lines.

Point-to-point leased lines connect data equipment in separate locations. Normally four-wire lines with two circuit pairs are used, one line for each direction of transmission. Modems interface the data equipment to these lines. Imperfections that cause data errors on analog lines fall roughly into two groups: steady-state impairments (e.g., noise or amplitude modulation) and transient impairments (e.g., impulses or signal dropouts). In addition, telephone companies may condition lines to meet the attenuation distortion (frequency response) and delay distortion (envelope or group delay) characteristics required by modern high-speed modems.

The HP OpenView Data Line Monitor (OVDLM) is an analog leased-line monitoring system for multivendor networks, based on the HP 4948A in-service transmission impairment measuring set (ITIMS).[1] The HP 4948A permits the testing of lines while they are still in use. Conventional testing of analog lines requires the lines to be taken out of service while test signals, such as test tones, are applied. Alternatively, modem-based line monitoring and management systems are available from manufacturers of datacom equipment. However, these systems are usually proprietary and may require specialized and expensive smart modems. The HP 4948A works with ordinary modems in the range of 2.4 to 14.4 kbits/s that are compatible with AT&T or CCITT standards. HP OpenView Data Line Monitor is compatible with the other HP OpenView network management applications and can run concurrently with them.

The HP OpenView Data Line Monitor software controls a single HP 4948A and one or more HP 3777A channel selector access switches to monitor or troubleshoot analog datacom circuits at one location. Each HP 3777A switch can access up to 30 four-wire circuits and can be cascaded to achieve the required access capacity to a maximum of 31 switches. The functionality of OVDLM is provided entirely by the software running in the HP OpenView workstation. Unlike some of the other HP OpenView applications, OVDLM is not split into two parts, with one part running on a remote computer. Instead, the ITIMS and switches are controlled directly from the HP OpenView workstation via the HP-IB (IEEE 488, IEC 625). The HP-IB has the advantage that its high speed enables the ITIMS and up to 13 access switches to be controlled from just one interface card in the HP Vectra PC. (Because of the electrical limit of 15 devices on one HP-IB, two interface cards are required for up to 27 switches, and three for the maximum of 31 switches). In many installations it is possible that the HP OpenView workstation might be situated at some distance from the communications rack where the ITIMS and switches are located. The distance limit of 20 meters with HP-IB cable can be overcome by using HP-IB extenders. A pair of HP 37204A multipoint HP-IB extenders using fiber optic cable allows HP-IB extension up to 1250 meters with negligible loss in performance. For greater distances, such as when the HP OpenView workstation and the ITIMS are at different sites, a pair of HP 37201A HP-IB extenders can extend the HP-IB to unlimited distances over ordinary tele-

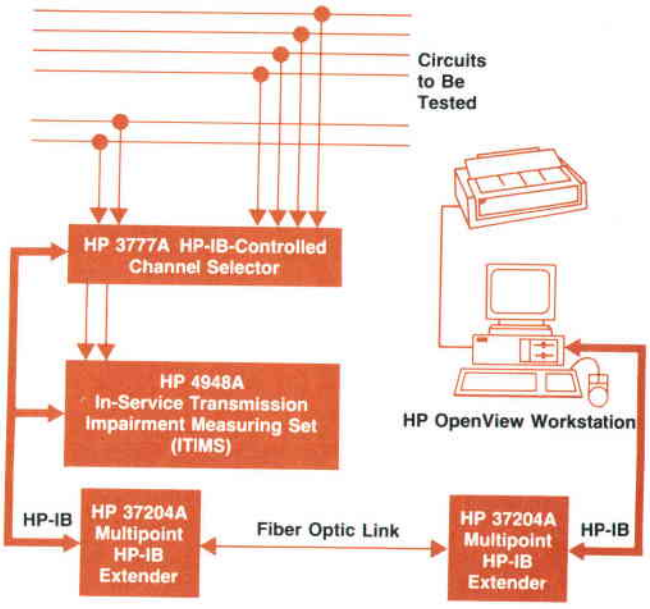


Fig. 1. *A typical equipment configuration for using the HP OpenView Data Line Monitor (only one access switch, the HP 3777A channel selector, is shown).*

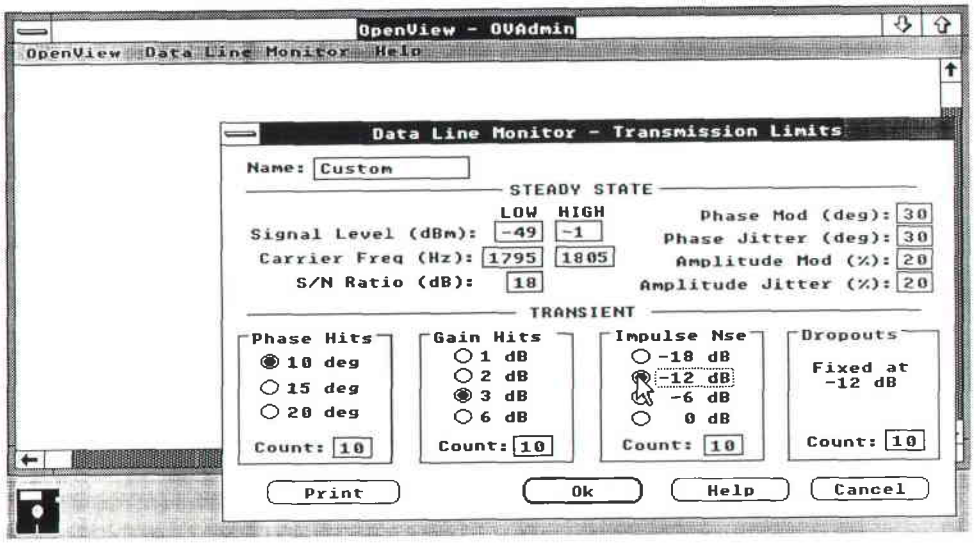


Fig. 2. *Screen used to enter transmission limits.*

phone lines, albeit with a loss in performance. Fig. 1 shows a typical equipment configuration with one access switch.

For routine alarm monitoring, OVDLM can be set to test all lines automatically in sequence. For troubleshooting, a single line can be selected and monitored continuously. This is particularly powerful for trapping intermittent faults. A description of all lines to be monitored is stored in the HP OpenView Windows data base, including details of the modem type and transmission performance limits. When OVDLM detects a problem with a particular line, the color of that line is changed to red and a message is displayed in the HP OpenView alarm window. OVDLM can report line performance in two ways. First, routine day-to-day monitoring typically uses an alarm-only mode, which indicates when any of the key analog parameters for each line go outside their predefined limits. Secondly, when data is required for line performance benchmarks or trend analysis, OVDLM is able to store the maximum, minimum, and average values of all line characteristics during a selected measurement period. The results for each line are stored in a common log file which can be viewed at any time.

Like all HP OpenView applications, OVDLM is divided into three programs: OVAdmin, OVDraw, and OVRun. The article on page 60 describes these programs.

OVDLM and OVAdmin

In OVAdmin the user (a network or datacom manager) can define sets of transmission and conditioning limits. Transmission limits include specifications such as the minimum and maximum signal level, minimum signal-to-noise ratio, maximum number of dropouts allowed in a 15-minute interval, and so on. Conditioning limits define attenuation distortion and delay distortion masks. Typically these sets of limit values are based on AT&T or CCITT specifications for analog leased lines. Each set of limits is given a name, which is then used to specify the limits for a particular line. Many lines can thus share common sets of limits. The sets of limits are stored in a special OVDLM data file rather than the HP OpenView Windows data base. Where the data is stored is not apparent to the user. Fig. 2 shows the screen used to enter the transmission limits and Fig. 3 shows the

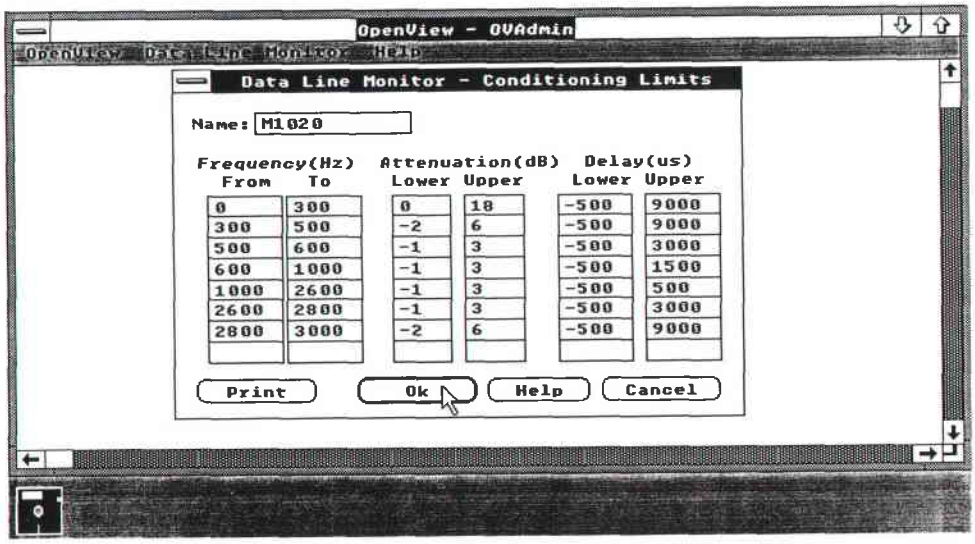


Fig. 3. *Screen used to enter conditioning limits.*

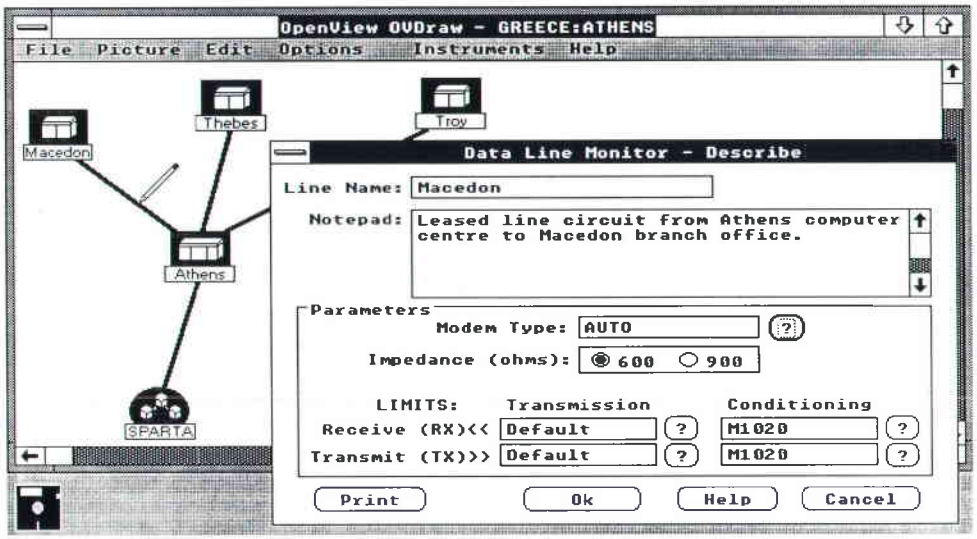


Fig. 4. *The* Describe *dialog box is used to describe datacom line characteristics. The pencil is used to select the desired line.*

screen used to enter the conditioning limits. The data shown entered in Fig. 3 is the attenuation and group delay mask from CCITT recommendation M.1020 for international leased analog circuits.

OVDLM and OVDraw

In HP OpenView Windows, lines drawn on the network map can be managed in the same way as computer or component icons. OVDLM registers with HP OpenView Windows to manage these lines. In OVDraw the user draws the datacom lines on the network map, describes their details, and enters configuration information about the ITIMS and switches. Lines are described in exactly the same way as computer or component icons are described in other HP OpenView applications, that is, the user selects the Describe menu item from the Edit menu, moves the pencil cursor over the line and clicks the mouse button. This brings up the line Describe dialog box. OVDLM extends the default line description of HP OpenView Windows to include details of the line impedance, the modem type, and the names of the line transmission and conditioning limits (as set up in OVAdmin). Fig. 4 shows a sample network map with a line Describe dialog box. Note that the modem type is set to AUTO. When this line is monitored in OVRun, the ITIMS will automatically search for the correct modem type by examining the live line signal.

A difference between lines and most other managed objects is that lines are passive because they need separate tools (the instruments) to do the management. The ITIMS and switches therefore have to be described in OVDraw. To accomplish this an Instruments menu was added to the OVDraw menu bar containing the Data Line Monitor menu item. Selecting this menu item will bring up an ITIMS Instruments dialog box on which the HP-IB interface select code and the address of the ITIMS and first-level switch can be entered (see Fig. 5). From this dialog box a Connect Line dialog box can be accessed for connecting lines to the ports of the first-level switch. A Connect Switch dialog box can also be selected for connecting a second level of switches to the first-level switch. Subsequently lines can be connected to the second-level switches.

OVDLM and OVRun

The OVRun part of OVDLM controls the program's

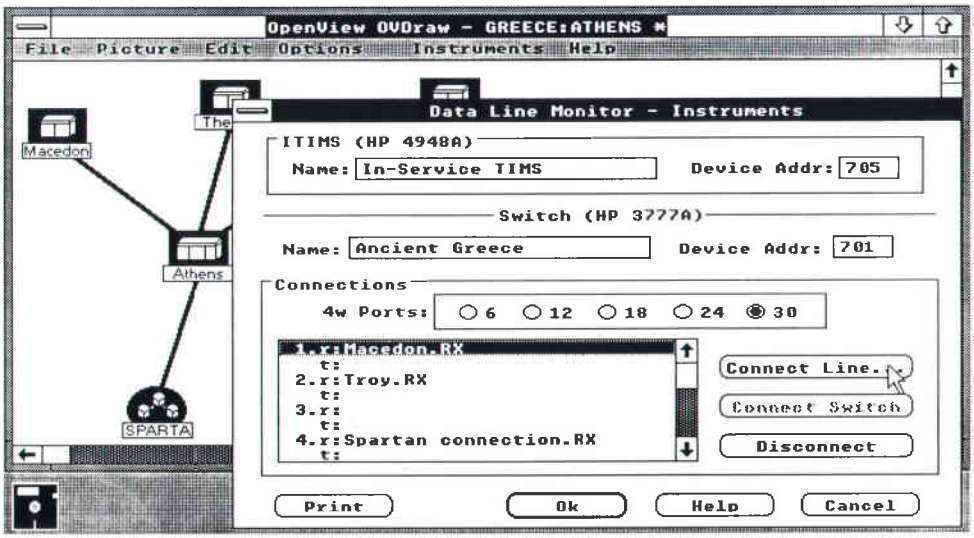


Fig. 5. Instruments *dialog box used to describe the HP-IB interface codes and the various connections between the line and switch boxes.*

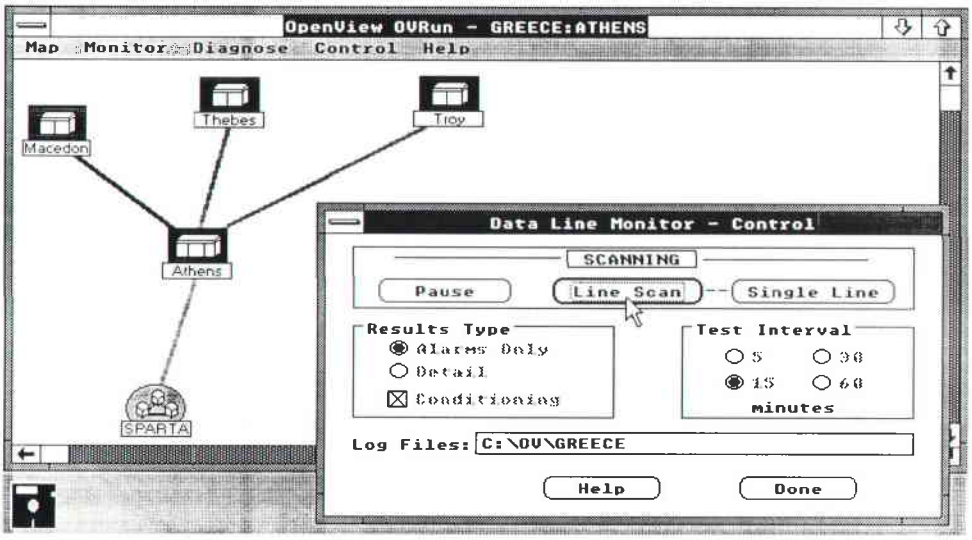


Fig. 6. Control *dialog box with* Line Scan *menu item selected,* Alarms Only *results, and 15-minute test intervals.*

monitoring activities. The user sets up global monitoring parameters and starts and stops the monitoring via a Control dialog box accessed through a data line monitor menu item on the Control menu. The Set Parameters menu, which is a default menu item supplied by HP OpenView Windows, is not used for this function because it requires a network line name to be selected first. Entering a line name was found not to be intuitive for setting the global monitoring state because the global status information is not associated with any particular network line.

The Control dialog box allows the monitoring state to be set to line scan mode or single-line mode, along with the required results and a test interval. The name of a disk file to hold the results can also be entered. Fig. 6 shows a Control dialog box with the Line Scan menu item selected, Alarms Only results, and a 15-minute test interval selected. The Results Type and Test Interval controls are greyed (dimmed) because they can only be altered when monitoring is paused.

In line scan mode OVDLM will monitor each line in turn for a duration set by the test interval (5, 15, 30 or 60 minutes). For example, if there are 30 4-wire circuits and a five-minute test interval is selected, a complete scan of all the circuits will take about six hours (assuming both the transmit and receive circuit pairs of each line are to be tested and allowing a minute per test for ITIMS training). If alarms-only results are selected, a summary of the impairment violations is written to the results log file at the end of the test interval (C:\OV\GREECE in Fig. 6). If conditioning is selected as well as alarms only, the attenuation and delay results will be checked against the conditioning limits (see Fig. 3) and a pass/fail result will be written to the log file. If detail results are selected, the maximum, minimum, and average values of the steady-state line impairments, along with the number of transient events, are recorded at the end of the test interval. If conditioning and detail results are selected together, the complete attenuation and delay data is recorded as well.

In single-line mode OVDLM will continuously test only one line that has been selected on the network map. The Results Type selection works similarly to line scan mode, except that in Alarms Only, each individual violation is re-

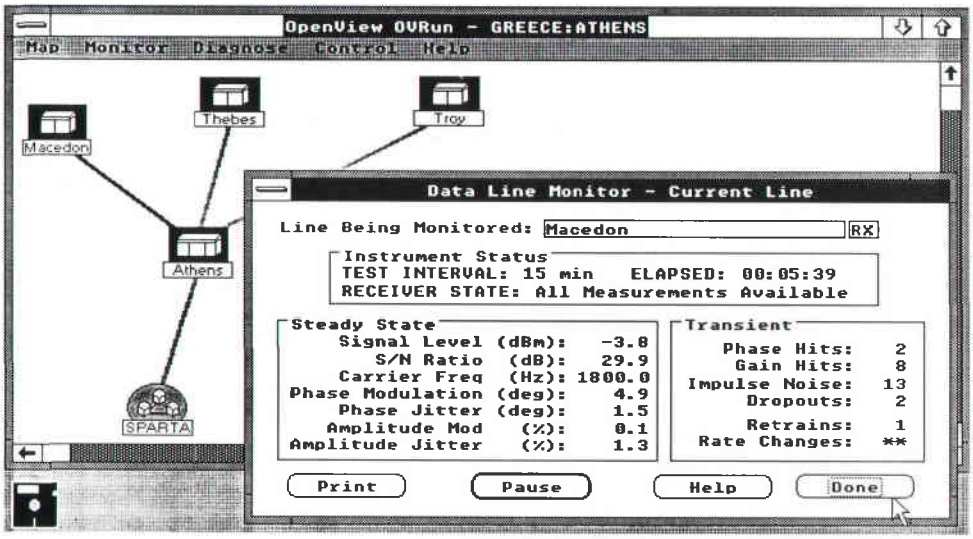


Fig. 7. *An example of a current line window.*

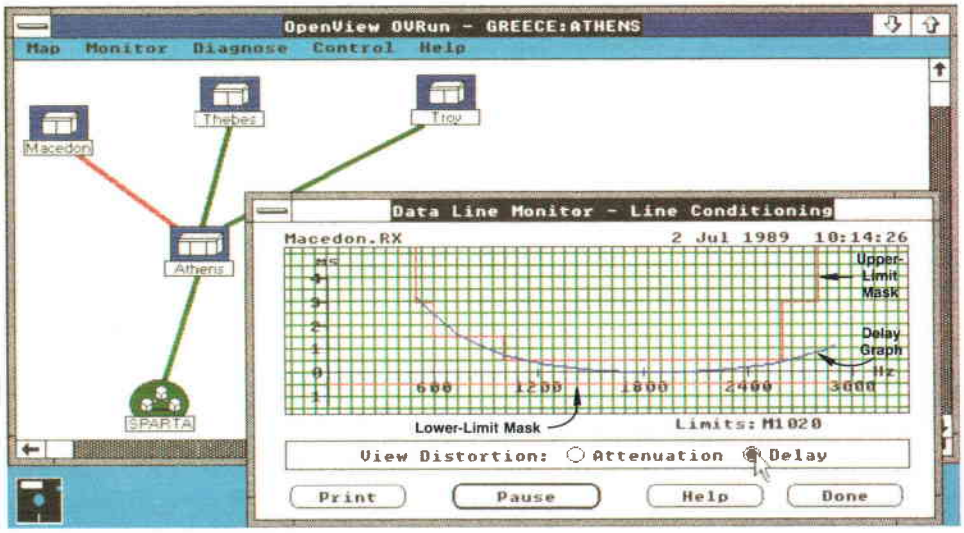


Fig. 8. *An example of a line conditioning window. The data for the upper-limit mask comes from the delay upper column in Fig. 3 and the lower-limit mask comes from the delay lower column in Fig. 3.*

corded in the log file. This is useful for tracking down intermittents on a particular circuit.

Once started, monitoring runs in the background even if all the OVDLM dialog boxes have been closed and the user is working in another HP OpenView Windows application. Thus the user is not normally aware of the monitoring process. However, during HP-IB input/output, the Vectra can appear busy for a few seconds, especially when reading back the line conditioning data from the ITIMS. This is because asynchronous HP-IB input/output is not supported in the Microsoft Windows environment. To inform the user when HP-IB input/output is in progress, OVDLM changes the cursor icon to an HP-IB busy icon consisting of an hourglass and an HP-IB logo. At the end of the test interval the color of the line on the network map is changed to green if the measured performance is within limits and to red if it is outside one or more limits. In the latter case an alarm message is sent to the HP OpenView Windows alarm window.

During monitoring the user can choose to view either real-time results for the line currently being tested or historical results for a line taken from the log file. These results are accessed through Current Line and Line History menu items on the Diagnose menu.

Selecting the Current Line menu item will display a current line window and a line conditioning window. The current line window shows the steady-state and transient results and is refreshed from the ITIMS every twenty seconds. Any results that are outside the limits for the line are shown in red. The conditioning window displays a graph of attenuation versus frequency or delay versus frequency, using the ITIMS data-spectrum results and the conditioning mask limits for the line. This window is only refreshed every minute, the period with which the ITIMS remeasures the data spectrum. Fig. 7 shows an example of a current line window. Here the impulse noise and retrains counts are above limit and hence shown in red. As a result, the line on the map is also turned red. Fig. 8 shows a sample line conditioning window. Note that the delay graph moves outside of the mask below about 1000 Hz.

The Line History menu item allows browsing through the OVDLM log files for historical results of a particular line. Filters are available to select the types of results required. For example, there are filters that can be used to narrow the selection to alarms results or details results for a particular transmit or receive pair.

Using Log Data

It is important for datacom managers to track the long-term performance of their lines. The OVDLM log files are in ordinary ASCII text and organized so that they can be easily read by other application programs. For example, the data can be imported into a spreadsheet program, such as Microsoft Excel, and trend analysis done on the detail results for each line. This trend analysis allows lines with deteriorating performance to be spotted before they become critical. Demonstration Excel macros to do this are supplied with the OVDLM software.

Acknowledgments

Acknowledgments are due John Duff, who designed and coded the OVDLM software, and Mark Dykes, Garry Irvine, and Elaine Butterwith for advice on many aspects of the definition and implementation of OVDLM.

Reference

1. N. Carder, et al., "In-Service Transmission Impairment Testing of Voice-Frequency Data Circuits," *Hewlett-Packard Journal*, Vol. 38, no. 10, October 1987.

Network Management for the HP 3000 Datacom and Terminal Controller

The HP OpenView DTC Manager software is responsible for controlling, monitoring, and diagnosing the DTCs on a local area network. Its functions can be exercised either from a local workstation on the LAN or from an HP Response Center or other remote workstation.

by Serge Y. Amar and Michele A. Prieur

A NEW GENERATION OF HP 3000 COMPUTERS was born in 1986. The distinctive features of this generation are HP Precision Architecture,[1] the MPE XL operating system,[2] and the input/output structure.[3]

One of the peculiarities of the input/output structure is the way terminals and printers are connected to a host HP 3000 computer. They are connected through a controller (originally called the distributed terminal controller or DTC), which is connected to a local area network. Originally, because the DTC code and configuration file were too big to fit in nonvolatile memory, the host was in charge of downloading them to the DTC at power-up. The host was also responsible for building the DTC configuration file and for managing the DTC (reset, upload, self-test, etc.).

For the first release of the MPE XL software, the LAN was used more as an I/O bus than as a network, in the sense that a terminal plugged into a DTC could establish a connection to one and only one host computer even if the LAN was shared by more than one host. Moreover, some important features were missing, such as wide area network access (X.25).

With the release of the MPE XL 2.0 operating system in October 1989, major new functionalities are implemented in the DTC: X.25 access, PAD (packet assembler/disassembler) support, terminal I/O switched connections, back-to-back connections, and others. All of these services can be shared by multiple MPE XL systems connected to the LAN. In keeping with its expanded capabilities, the DTC has been renamed the *datacom and terminal controller*.

The DTC now offers LAN-accessible shared services, and is no longer tied to one host system. With the back-to-back feature, it can even work without any hosts on the LAN. For these reasons, a new way had to be found to manage the DTC and its services. This is the function of the HP OpenView DTC Manager workstation.

The HP OpenView DTC Manager

The HP OpenView DTC Manager is the network management software for the new services of the DTC. It is responsible for controlling, monitoring, and diagnosing the DTC.

The HP Openview DTC Manager software runs on an HP Vectra ES/12 personal computer with a VGA display, a 40-megabyte hard disk, an HP LAN access card, and a 2M-byte expansion board. It runs under the Microsoft Windows environment and the HP OpenView Windows umbrella. It implements a graphical user interface, allowing the user

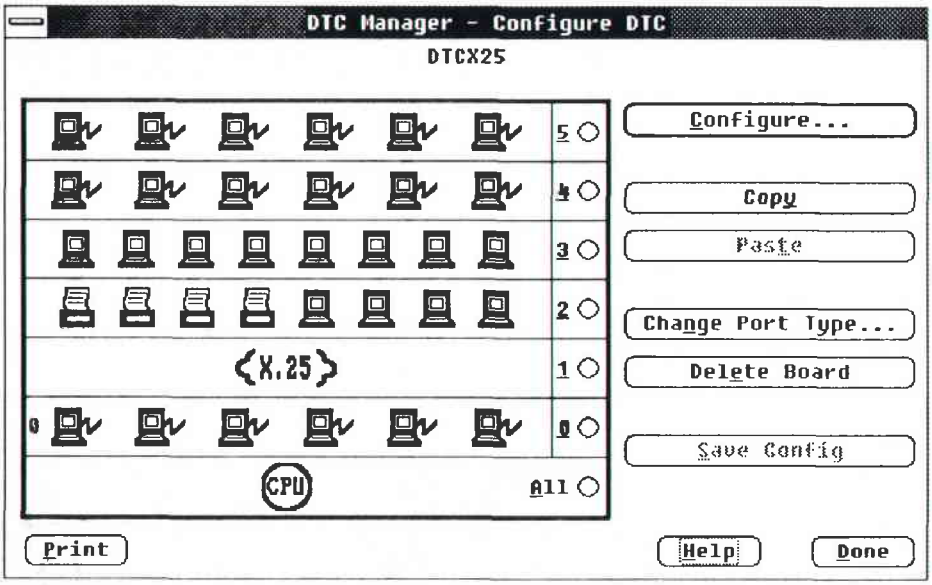


Fig. 1. *Most HP OpenView DTC Manager functions begin with the display of the DTC rear panel, which shows the types of cards and devices that are plugged into the DTC. DTC stands for datacom and terminal controller.*

to trigger management functions on the local area network DTC components.

The contributions of the HP OpenView DTC Manager include:

- The use of HP OpenView Windows to allow the user to "touch" the network.
- Easy access to the DTC components through a graphic, logical view of the DTC rear panel.
- Remote access to all of the DTC management functions through a modem connection using the same graphic user interface.
- The ability to add a support tool at any time without modifing the HP OpenView DTC Manager code.

HP OpenView Windows is a network shell that mainly displays and manages graphic maps of networks. At initialization, it spawns the applications that have an entry in the OpenViewApps section of the WIN.INI file (see article, page 60). It provides services such as map object management, menu and menu items addition, alarms, single-key data base access, and others.

Almost all of the HP OpenView DTC Manager's functions, triggered by the user from the HP OpenView network map, begin with the display of the DTC rear panel, which allows the user to see at a glance the types of cards and devices that are plugged into the DTC (Fig. 1). Using the mouse, the user can easily select the part of the DTC on which to execute the management function. The selectable components are: the CPU board, a card (X.25, direct connect card, modem connect card), a terminal, or a printer. Each of these is represented by a specific icon.

All features of the HP OpenView DTC Manager can be accessed remotely through a 1200-baud or 2400-baud modem. The user interface is completely separate from the input/output functions so that it can be run in a remote PC without the need to pass graphic data through the serial modem link. This structure was chosen to optimize the exchange of information in the case of remote access—only relevant binary data is transferred. A major problem of such a structure is that it is not possible to take advantage of some Microsoft Windows features. For example, a list box can contain more information than can be transferred all at once over the serial link to load the list box. Therefore, the HP OpenView DTC Manager handles the scroll bars of list boxes to navigate through the lists, and only list data to be displayed is transferred.

Hooks have been implemented in the HP OpenView DTC Manager so that a DTC support tool can be added at initialization time, using the DtcManagerApp section in the WIN.INI file. For example, if a dump formatter were implemented following the guidelines, it would be able to take full advantage of the remote access capability of the HP OpenView DTC Manager.

Software Structure

The HP OpenView DTC Manager consists of four Microsoft Windows components (Fig. 2):

- PCMPRMP. This module is responsible for interfacing with the HP OfficeShare driver and implementing the two management protocols MP and RMP.[4,5]
- DTCMGRIO. This module is responsible for maintaining the data base associated with DTC management and for translating requests coming from the user interface to requests that can be understood by the PCMPRMP module. It is also responsible for deciding whether a download request coming from a DTC must be serviced or rejected.
- DTCMGRUI. This module is responsible for handling the user interface—for example, interpreting requests coming from the user, requesting additional information when needed, and displaying the results of the requests.
- RLS/CA. This module has two parts. RL Switch is responsible for managing the local and remote modes and for switching messages between DTCMGRIO and DTCMGRUI either on the same PC or through a serial modem link. Data is never exchanged directly between DTCMGRUI and DTCMGRIO. CA stands for connect application. It is responsible for the logon and logoff process in both local and remote modes.

Two of these modules, DTCMGRUI and RLS/CA, are HP OpenView Windows applications. An HP OpenView Windows application is, first of all, a Microsoft Windows application, that is, it has a WinMain, which performs initialization and implements the GetMessage/DispatchMessage loop. However, during the initialization process, it must call some HP OpenView intrinsics, and it does not create its own main window. The application's main window is created by HP OpenView Windows and acts as a communication window between HP OpenView Windows and the application. The application must provide the window procedure for this communication window.

Because an HP OpenView Windows application has no

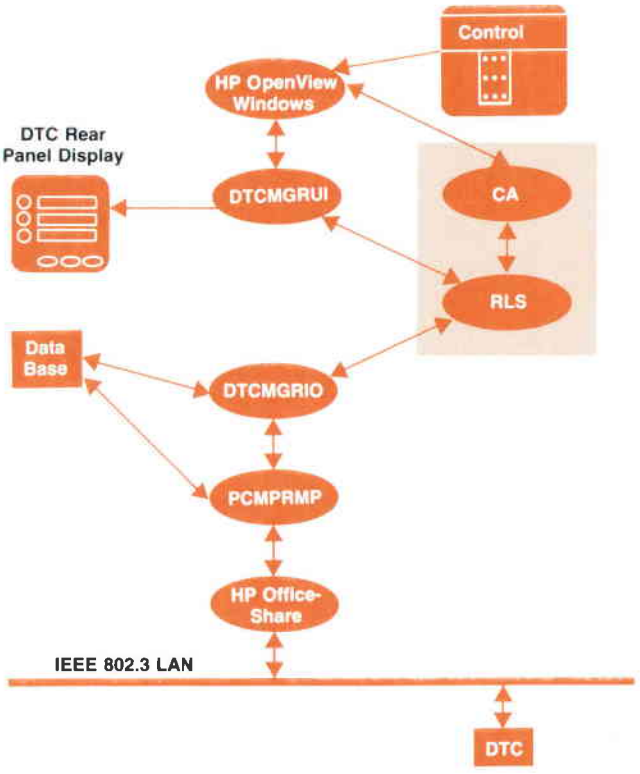


Fig. 2. *The HP OpenView DTC Manager consists of four Microsoft Windows components: PCMPRMP (protocol controller), DTCMGRIO (data base manager and translator), DTCMGRUI (user interface), and RLS/CA (communication and logon).*

main window of its own, it has no menu bars. Instead, it requests HP OpenView Windows to add menu items to the main HP OpenView Windows menu bar. An HP OpenView application has to provide for the processing of all the menu items it has registered, and for the processing of some HP OpenView general messages.

User Interface Structure

The structure of the user interface module (DTCMGRUI) is highly modular (see Fig. 3). It is based on the property of Microsoft Windows that once created, a window is an independent entity able to receive and process its own messages transparently to the process that has created it.

DTCMGRUI consists of a main window and a series of child windows. The main window is the communication window created by HP OpenView Windows during initialization of the application. It is an invisible window and it is mainly dedicated to receiving messages coming from HP OpenView Windows. When the user selects a function managed by the application, HP OpenView Windows sends a message to the communication window.

The window procedure of the communication window is just a dispatcher that creates child windows of the requested class. These are invisible child windows, shown as level 1 windows in Fig. 3. Once it has created the right level 1 child window, the communication window procedure sends it a trigger. Then, it no longer worries about the requested function since it will be processed by the created child window. The communication window simply waits for the Done message sent back by the child window, and then destroys the child window, since this means that the requested function is completed. Every child window is a function of a certain type. For example, there are child windows of type Configure, Set Parameters, Upload DTC, and so on.

Level 1 child windows are only created and destroyed by the main level. Although a child window could destroy itself, for consistency this is never done in DTCMGRUI. The main level is always aware of what is active below it.

In a similar manner, level 1 child windows can create invisible level 2 child windows, which are independent entities for independent subfunctions. Level 1 child windows can also create visible level 2 child windows, which are dialog boxes. Dialog boxes can also be created by invisible level 2 child windows.

One of the main differences between an invisible child window and a dialog box is that an invisible child window is destroyed by its parent, whereas a dialog box destroys itself. Moreover, an invisible child window, when it has completed its function, sends a predefined message to its parent. A dialog box child window, when it is created, receives as a parameter the message it must send back to its parent once it has completed its function. Dialog boxes do not have predefined completion messages.

Each of the child windows, either invisible or visible, is able to communicate with DTCMGRIO through the RL Switch module without the help of the communication window by giving its own handle in any DTCMGRIO request.

Remote Access Structure

One of the objectives for the HP OpenView DTC Manager was remote access to network management applications. Two PCs with modems running the HP OpenView DTC Manager can communicate and the same DTC management capability is available at both (Fig. 4). For example, from an HP Response Center PC, an HP engineer can manage a customer's DTCs. In fact, only the user interface (DTCMGRUI) and RL Switch run on the Response Center PC. Commands are sent to the DTCMGRIO module of the customer's PC

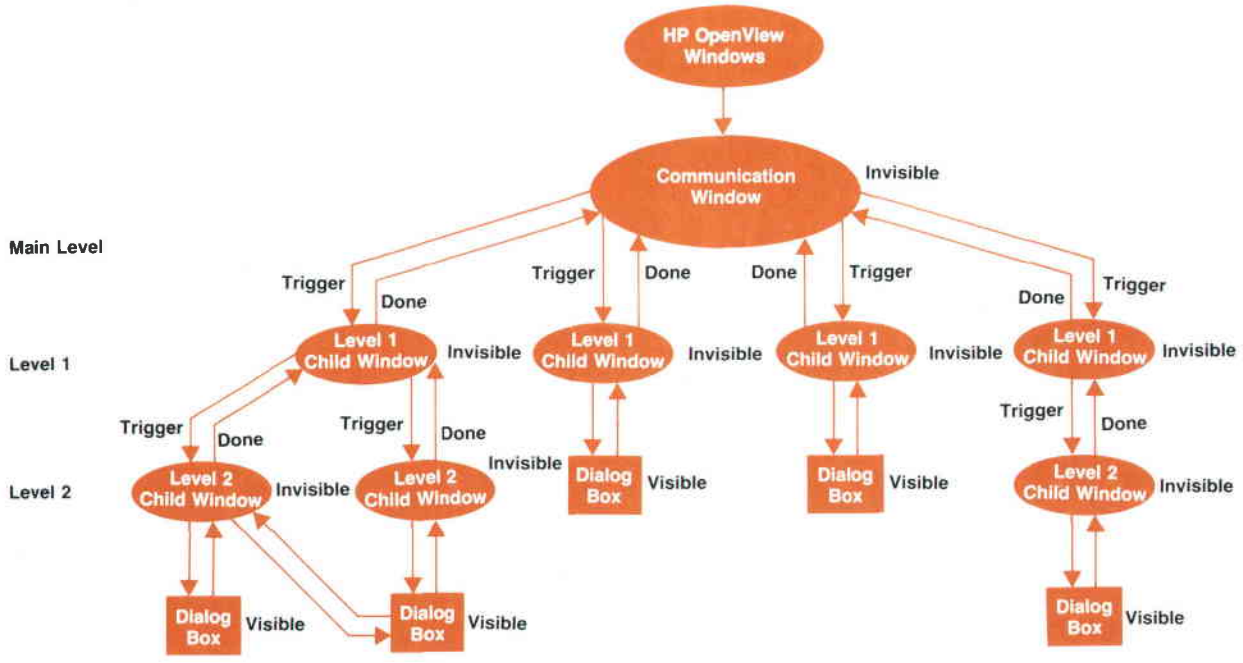


Fig. 3. *Structure of the user interface module DTCMGRUI.*

through the serial link.

This functionality is implemented in RL Switch, which is able to switch from local to remote mode and to send orders and replies between DTCMGRUI and DTCMGRIO either locally within the same PC or remotely over the modem line (Fig. 5).

The user is able to log on locally or remotely using the connect application, CA. At initialization, this module adds four menu items to the control menu of HP OpenView Windows: Logon, Logoff, Remote Connect, and Remote Disconnect.

When Logon is activated, Remote Connect and Remote Disconnect become inactive. When Remote Connect is activated, both Logon and Logoff are inactive. Finally, when the user is neither locally logged on nor remotely connected, both Logon and Remote Connect are enabled.

When a user at the Response Center PC chooses the Remote Connect menu item, the DTCMGRUI module in the Response Center PC receives the following sequence of messages:

```
RL_PREPAREFORCONNECT with param =
 RESPONSE_CENTER_PC

RL_LOGONSTATUS
```

On the first message, DTCMGRUI just stores the fact that it is no longer in local mode but not yet in remote mode. On the second message, DTCMGRUI stores the fact that remote mode is now active. All HP OpenView DTC Manager menu items are enabled and the inactivity timer is started.

The DTCMGRUI module located in the customer PC receives only the message RL_PREPAREFORCONNECT with param = CUSTOMER_PC. This message is ignored.

Thereafter, except for some very rare cases, whether the HP OpenView DTC Manager is working in remote or local mode is absolutely transparent to DTCMGRUI. There are only two exceptions. First, the display of the Function In Progress dialog box depends on the mode for some requests to DTCMGRIO. Second, in the dialog boxes of type OpenView DTC Manager List, when the items are read one after another and the HP OpenView DTC Manager is in local mode, the items are displayed only when the list box is full to avoid

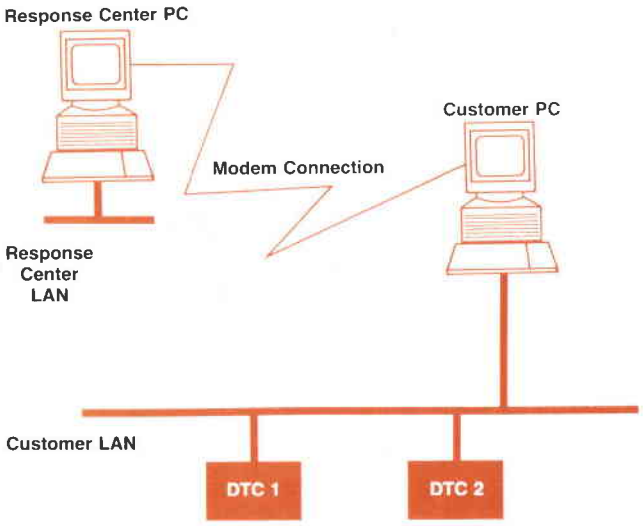


Fig. 4. *The HP OpenView DTC manager allows remote access to all DTC functions. For example, from a PC in the HP Response Center an HP support engineer can access a customer's DTCs.*

flickering effects. In remote mode they are displayed immediately.

A remote connection is closed if (1) the Response Center engineer completes the job and activates the Remote Disconnect function, (2) the customer requests that the connection be aborted, or (3) the connection between the two modems breaks. In case (1), DTCMGRUI receives RL_LOGOFFREQUEST from RL Switch, and then RL_REMOTECONNECTCLOSED. In cases (2) and (3), DTCMGRUI receives only RL_REMOTECONNECTCLOSED. On RL_LOGOFFREQUEST, DTCMGRUI is free to accept or reject the logoff. If no function is active, the logoff is accepted. If any function is active, the logoff is rejected and the remote disconnect procedure is interrupted. On RL_REMOTECONNECTCLOSED, any active function is aborted and DTCMGRUI assumes that its new status is local and logged off.

Figs. 6, 7, and 8 show the sequence of messages exchanged between CA, RL Switch, and DTCMGRUI in the cases of a remote logon, a remote logoff, and a customer-

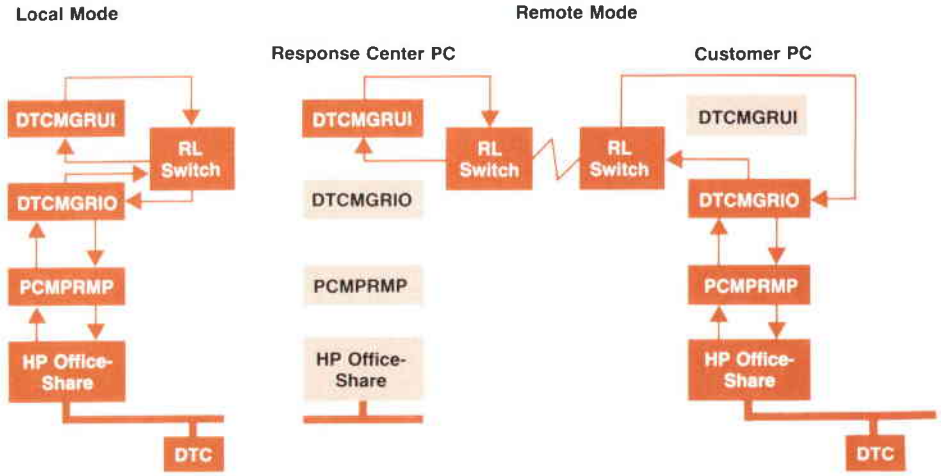


Fig. 5. *The RL Switch module has the ability to switch between local and remote modes and route messages accordingly.*

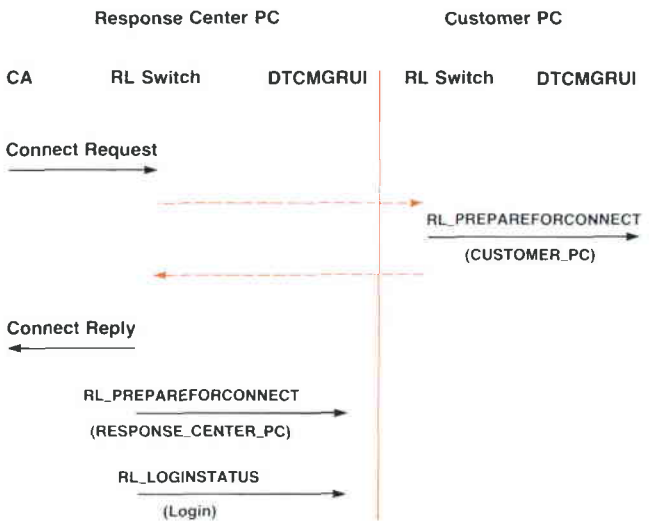


Fig. 6. *Messages exchanged for a remote logon.*

forced abort or a modem line failure.

Dialog Boxes

Microsoft Windows provides two types of dialog boxes: modal and modeless. Modal dialog boxes do not allow the application (here DTCMGRUI) to do any processing except for handling the dialog box, while modeless dialog boxes allow the application to process both the dialog box and other application windows concurrently.

In compliance with the *HP OpenView Application Style Guide*, all dialog boxes in DTCMGRUI are modeless dialog boxes. Modeless dialog boxes consume less stack space than modal boxes. They behave as standard child windows and are more bug-free than modal boxes. However, the HP OpenView DTC Manager often needs modal behavior. For example, when a dialog box is displayed, the user is not allowed to do anything except enter information in the box. To meet this need, modal behavior is simulated by creating a modeless dialog box and disabling the window that has created it.

The modeless dialog boxes inside DTCMGRUI can be classified into three categories: classical, HP OpenView DTC Manager lists, and backplane.

The classical dialog boxes are normal Microsoft Windows dialog boxes. They implement list boxes, radio and check buttons, edit fields, and so on. All of these controls are handled by Microsoft Windows after the dialog box is created and initialized. Most of the dialog boxes fall into this category.

HP OpenView DTC Manager lists are dialog boxes that contain list boxes handled by DTCMRGUI instead of Microsoft Windows. These list boxes are initialized with 13 items and are always reset and reinitialized with 13 items maximum. The scroll bar on the right side of the list box is not part of it, but is a scroll bar control handled by the dialog box as an independent scroll bar. The items in these lists are not stored in memory by DTCMGRUI but are requested one by one from DTCMGRIO through RL Switch. Every time the user wants to add, delete, or modify an item or scroll up or down the list, the whole list is reset and orders are sent to DTCMGRIO to reread the displayed part item by item (Fig. 9). This mechanism is used because each record in the list may be quite long and lists may have many records, so the amount of memory required to load all the items at once in memory and to have Microsoft Windows handle the list would be too large. Moreover, in remote mode, if the amount of information requested at one time is too large, the time between the start of a function and the function's actually becoming active would be too long. With this principle, in remote mode, each item is displayed as soon as it arrives, so it seems to be faster, and in any event, a maximum of 13 items will be transferred even if the list is 512 items long.

Support Tool Integration

Support engineers sometimes need to run special appli-

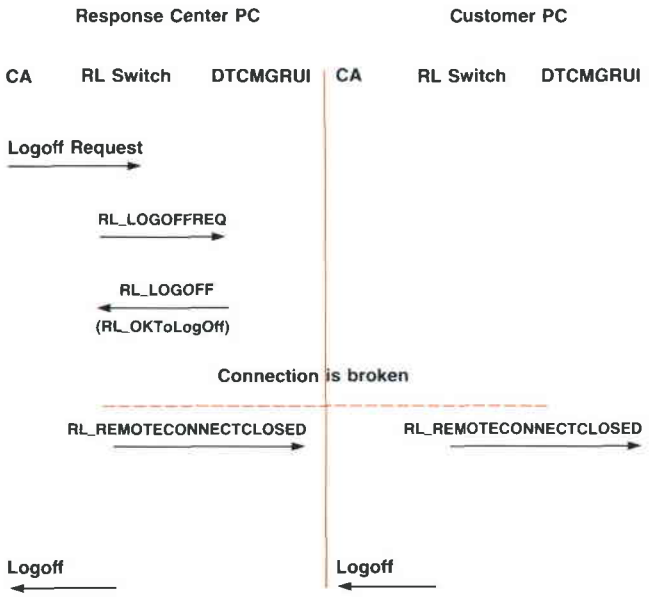


Fig. 7. *Messages exchanged for a remote logoff.*

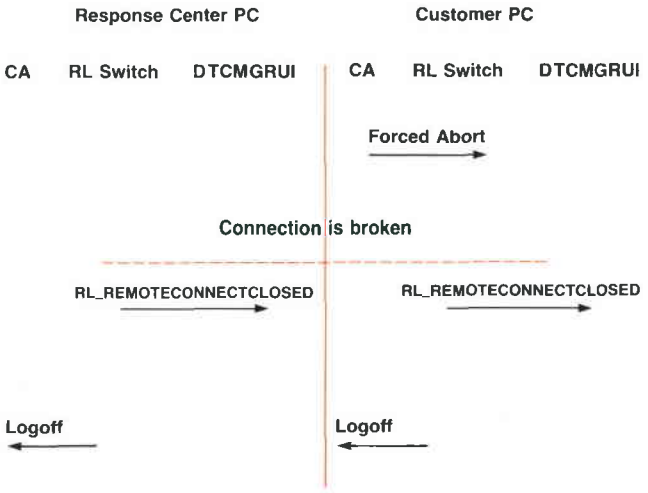


Fig. 8. *Messages exchanged for a customer-forced abort or a modem line failure.*

cations—for example, a light formatter for the dump files coming from the DTC. These applications must be integrated into the HP OpenView DTC Manager, even though they were not defined at the time the HP OpenView DTC Manager code was written. To accommodate this need, hooks were implemented in DTCMGRUI and DTCMGRIO so that these applications can be added dynamically and integrated into the HP OpenView DTC Manager.

Except for some special procedures during initialization and shutdown, a DTC Manager application is a normal Microsoft Windows application that doesn't need to deal with HP OpenView Windows. Like the DTC Manager, DTC Manager applications are structured in two parts, one for the user interface and one for everything dealing with the files and the DTC (the second part is referred to here as the I/O part of the application). This allows them to be used in remote mode as well.

The structure of the solution is shown in Fig. 10. The DTC Manager application is accessed through a menu in the Tools subsection of the HP OpenView main menu. The Tools menu and its submenus are added by DTCMGRUI during the initialization phase when the DtcManagerApp section is found in the WIN.INI file.

When the DTC Manager tool (or application) needs to be used, the user selects the application from the Tools menu. DTCMGRUI receives the request from HP OpenView Windows and sends a message to DTCMGRIO requesting it to spawn the I/O part of the application. All HP OpenView DTC Manager menu items are grayed. DTCMGRIO spawns the I/O part of the application, passes the RL Switch window handle to it, and replies to DTCMGRUI that the spawning was successful. With the reply, it sends the handle of the I/O part of the application.

DTCMGRUI spawns the user interface part of the application and passes to it the handle of the I/O part of the application and the handle of RL Switch. The modules of the application can now exchange messages directly through RL Switch using the RL_SENDDATA message.

When the application is no longer needed, the user selects a close function in the user interface part of the application, which notifies DTCMGRUI that it wants to quit. It does not quit yet. DTCMGRUI sends a request to DTCMGRIO to kill the I/O part of the application. DTCMGRIO kills the I/O part of the application and notifies DTCMGRUI. DTCMGRUI then kills the user interface part of the application and reenables the HP OpenView DTC Manager menu items.

The advantages of this solution are:

- The DTC Manager application modules are Microsoft Windows modules, not HP OpenView Windows modules. The DTC Manager application menu item will appear only if the WIN.INI file contains a DtcManagerApp section.
- The DTC Manager application code is loaded only when needed, so no memory is allocated to it while the HP OpenView DTC Manager is processing a management function.
- The DTC Manager applications don't have to deal with HP OpenView Windows, so they are independent of HP OpenView Windows and HP OpenView DTC Manager releases.

The DtcManagerApp section of the WIN.INI file has the following syntax:

```
[DtcManagerApp]
App1 = Appli1 , APPUI.EXE, APPIO.EXE
```

where App1 is the name of the DTC Manager application, Appli1 is the name of the menu item to be added to the Tools menu, and APPUI.EXE and APPIO.EXE are the names of the application's EXE modules. It is mandatory for these files to be in the subdirectory EXE of the DTC Manager data base. During the initialization process, DTCMGRUI looks in the WIN.INI file for the DtcManagerApp section. If it exists, the Tools menu is added to the HP OpenView menu bar. For each valid entry in this section, DTCMGRUI adds the

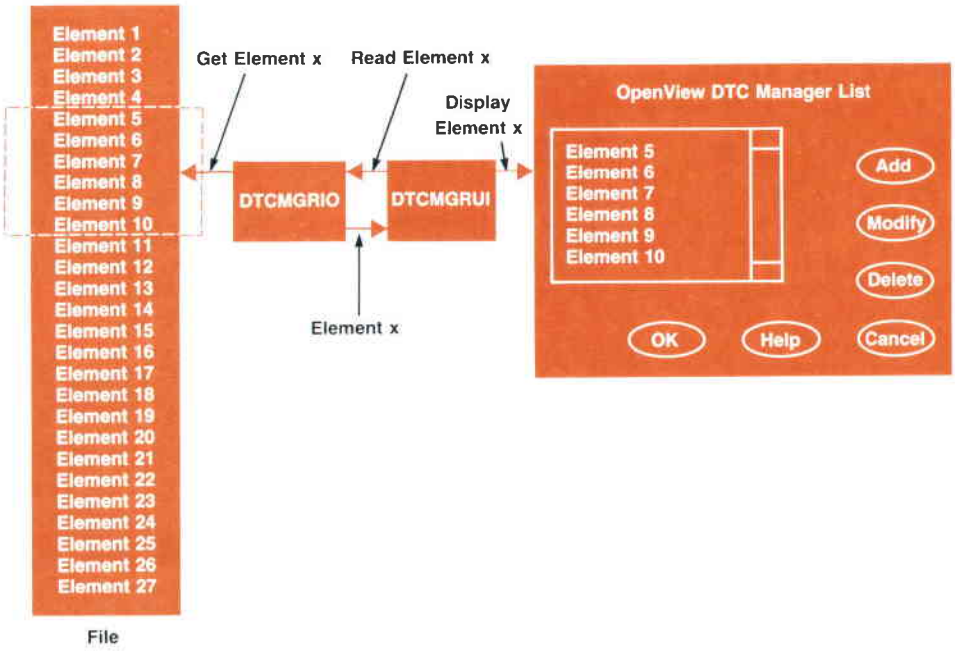


Fig. 9. *HP OpenView DTC Manager lists are dialog boxes that contain list boxes that are managed by DTCMGRUI instead of Microsoft Windows. The items in the list are requested one by one from DTCMGRIO (through RL Switch).*

corresponding menu item in the Tools menu.

HP OpenView DTC Manager Data Base

The HP OpenView DTC Manager does not use the file access facility provided by HP OpenView Windows for three reasons. First of all, the file access facility provides single-key indexed file access and the HP OpenView DTC Manager would have needed multikey access. Second, the configuration data for a DTC can be very large and retrieving it via HP OpenView file access would have drastically slowed the initialization of a remote access if the user chooses to transfer the HP OpenView network topology map. Finally, not using HP OpenView file access means that the DTCMGRIO and PCMPRMP modules can be Microsoft Windows applications and not HP OpenView Windows applications, so that even if HP OpenView Windows is inactive, they are always active and ready to receive DTC events and service DTC download and upload requests. This allows the user to free memory by closing HP OpenView Windows to run another Microsoft Windows application while the PC continues to serve DTC-triggered management functions.

The HP OpenView DTC Manager data base takes advantage of the MS-DOS® directory hierarchical structure and stores data within a tree of subdirectories of the DTCMGR directory. The DTCMGR directory with its subdirectory tree is created at installation time in the directory specified by the user; the default is the root directory. The user may not specify more than one level of directory—for example, C:\MYDIR. In this case the data base will be installed in C:\MYDIR\DTCMGR.

The following list illustrates a typical HP OpenView DTC Manager data base containing a DTC named DTCNAME on the HP OpenView map, loaded with two serial interface cards in slots 0 and 1, one X.25 card in slot 4, and slots 2, 3, and 5 empty. The DTC code and configuration have been downloaded and X.25 protocol has been started on the X.25 card but the PAD support protocol has not. In this list, lowercase is used for files and uppercase for directories:

```
...\DTCMGR\map802
          \acclist
          \copyhdr
          \DTCNAME.DTC\$CONF$\global
                                \globhdr
                                \backplan
                                \SLOT0\tioconf
                                \SLOT1\tioconf
                                \SLOT4\l123
                                      \switinfo
                                      \stsinout
                                      \padinsec
                                      \padacc
                                      \padswit
          \DTCNAME.DTC\$DWLD$\global
                                \globhdr
                                \backplan
                                \SLOT0\tiocont
                                \SLOT1\tiocont
                                \SLOT4\l123
                                      \switinfo
                                      \stsinout
          \DEFAULT\cpu.def
                  \term.def
                  \printer.def
                  \host.def
                  \globhdr.def
                  \acclist.def
                  \DC\struct.def
                  \MODEM\struct.def
                  \X25\l123.def
                       \stsswit.def
                       \stsinout.def
                       \padswit.def
                       \padacc.def
                       \padinsec.def
          \COPY
          \MONITOR
          \UPLOAD
          \CODE
          \EXE
```

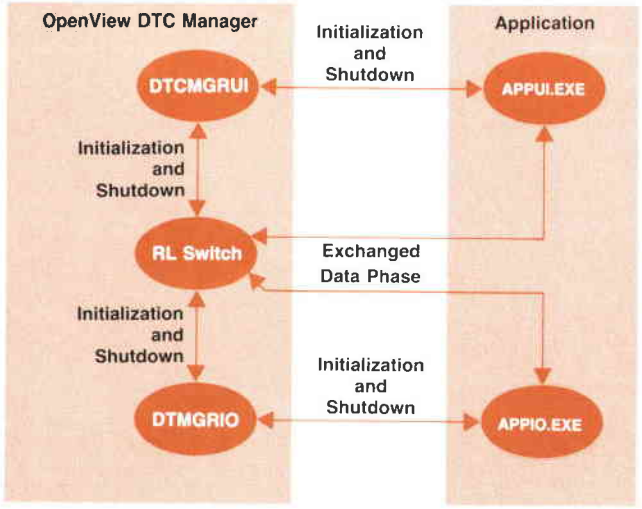


Fig. 10. *Integration of applications such as support tools into the HP OpenView DTC Manager.*

The file map802 contains one entry per configured DTC containing: the DTC name (e.g., DTCNAME), the DTC's current LAN (IEEE 802.3) address, the downloaded LAN address, and the DTC's LAN node name. It is updated and used by the DTCMGRIO module. Each time a DTC is created, its HP OpenView map name is written in this file with the two LAN addresses equal to 0. When the user saves a configuration, DTCMGRIO gets the LAN address contained in the DTC CPU configuration and puts it in the current LAN address in the map802 file. When the DTC requests a download, DTCMGRIO copies the current LAN address into the downloaded LAN address.

The file acclist is the security list file, which is downloaded to all the configured DTCs.

The DTCNAME.DTC subdirectories contain the DTC configuration data. They describe a DTC and consist of up to three subdirectories: $CONF$, $CONF$.TP, and $DWLD$.

The $CONF$ subdirectory contains the DTC off-line con-

figuration files. This configuration will be downloaded into the DTC on the next download operation. The contents of this directory and its associated subdirectories are modified when the user selects the Configure menu item. The $CONF$ subdirectory consists of three files and several subdirectories, one for each configured card of the DTC. The file backplan stores the information needed to display the backplane of a DTC. It contains 56 bytes:

- Byte 0: DTC Type. This is for future use, if we need to distinguish between multiple types of DTC.
- Byte 1: Verify Flag. Indicates whether the configuration associated with this backplane has been verified.
- Six records of 9 bytes each, one record per card. Record 0 is for card 0, and so on. Each record contains the card type (empty, direct connect, modem connect, or X.25), and the port types (terminal, printer, or host) for ports 0 to 7.

The file global contains the configuration of the CPU board (node name, IP address, logging classes, user prompt, welcome message). The file globhdr contains the LAN address of the DTC.

Each configured card has a SLOTx subdirectory, where x is the card number (from 0 to 5). The files contained in each SLOTx subdirectory completely describe the card. The number of files and their contents depend on the card type. If the card is an 8-port direct connect card, its SLOTx subdirectory contains the file tioconf, which stores the configurations of ports 0 to 7. If it is a 6-port modem connect card, its SLOTx subdirectory contains the file tioconf, which stores the configurations of ports 0 to 5 and two dummy configuration blocks. If the card is an X.25 card, its SLOTx subdirectory contains six files:

- l123 stores level 1, level 2, and level 3 configuration data and the permanent virtual circuit list used by X.25.
- switinfo is a list that stores the host resolution table (system-to-system switching information) used by X.25.
- stsinout is a list that stores the LUG table (system-to-system local user group) used by X.25.
- padinsec is a list that stores the PAD security table (PAD incoming security) used by PAD support.
- padacc is a list that stores the PAD device table (PAD access) used by PAD support.
- padswit is a list that stores the PAD switching table (DTC PAD switching information) used by PAD support.

The $CONF$.TP subdirectory contains the temporary configuration files. This directory is used for temporary storage of the modifications when the user is in the off-line configuration function. The $CONF$.TP subdirectory is created by DTCMGRIO when the user starts modifying a DTC configuration. It has the same structure as the $CONF$ subdirectory.

The $DWLD$ subdirectory contains the downloaded configuration files. This is the image of the DTC configuration data. This set of files is updated during dynamic configuration functions. The $DWLD$ subdirectory is created by DTCMGRIO when the user creates a new DTC. Upon successful completion of a download operation DTCMGRIO will copy files from the $CONF$ directory into the $DWLD$ directory, which keeps an exact image of the configuration data downloaded to the DTC. When dynamic changes are done using the Set Parameters menu item, only the data contained in this directory is updated unless the user requests that the modifications be copied to the off-line configuration as well.

The COPY subdirectory is created by DTCMGRIO when the user selects the COPY function (in the configuration menu) for the first time. Thereafter, it is never removed, allowing the user to make several copies of the same item using the PASTE function. The file ..\DTCMGR\copyhdr contains the type of item copied. The type can be DTC (the entire DTC configuration), SIC-DC (direct connect card), SIC-Modem (modem connect card), SSIC (X.25 configuration), Term (terminal configuration), Printer, or Host. This file is checked when the PASTE function is executed, since the COPY action and the PASTE action must be consistent. Then, according to the type of item, the subdirectory COPY is filled with all or part of a $CONF$ subdirectory.

The DEFAULT subdirectory contains all the default values for the configuration of a DTC. It has three subdirectories—DC, MODEM, and X25—and six default files, which contain the default configurations of each type of card in the DTC. The DC and MODEM subdirectories both have a file called struct.def (9 bytes), which contains the default structure for each card type and port type. The X25 subdirectory contains the default files to be used for an X.25 card.

The UPLOAD subdirectory receives the files containing upload data. These files are named depending on what kind of upload data they contain. DTCNAME and HOSTNAME are the names used by the user on the HP OpenView map.

The MONITOR subdirectory contains event logging files and trace files.

The CODE subdirectory contains the code files to be downloaded to the DTC.

The EXE subdirectory contains all the executable files of the HP OpenView DTC Manager and some support tools.

Memory Organization

The PC memory organization was one of the most difficult challenges of this project. The goal was to have the HP OpenView DTC Manager run on an HP Vectra ES/12 personal computer equipped with a 2M-byte additional memory board (the HP 45944A Vectra ES expanded memory card[6]).

Microsoft Windows can run in two different modes: large frame or small frame. In small frame mode, the memory area from C8000 to EFFFF is used to map expanded memory. The data segments of Windows applications, Windows libraries, and applications' dynamic libraries are loaded into conventional memory, that is, below 640K. This optimizes the use of expanded memory, but it uses a large amount of the shareable memory and minimizes the parallelism of applications.

In large frame mode, Microsoft Windows sets what is called a bank line, specifying how much space will be used in the 256K-to-640K slot for expanded memory mapping. Once this line is chosen, everything below the line (from 0 to the line) is considered shareable memory (called non-bankable memory, below-the-line memory, or global memory). Everything above the line (from the line to 640K) is used for page banking (called bankable memory or above-the-line memory). In this model the applications' data segments and dynamic libraries and the Windows libraries

are banked, meaning that they are loaded in expanded memory.

The advantage of running in large frame mode is that it frees a lot of the conventional memory, thereby increasing the parallelism of applications. The disadvantage is that it uses a lot of expanded memory because Microsoft Windows duplicates some Windows library segments in every bank and the smallest application will take no less than 112K bytes of expanded memory. Moreover, when an application needs more pages, Microsoft Windows will allocate pages to this application until it reaches the maximum number of pages that can be banked simultaneously. Thereafter, Microsoft Windows is not able to free these pages for other applications.

Code and resources (dialog box descriptions, text, etc.) are always banked in expanded memory whatever the mode.

The HP OpenView DTC Manager needs Microsoft Windows to start in large frame mode, because in small frame mode it runs out of global memory. Depending on the memory setting of the PC, Microsoft Windows/286 2.1 decides at start time in which mode it is going to run. Some of the criteria are:

- The amount of conventional memory left when Microsoft Windows is started. This depends on how many drivers are present and whether HIMEM is in use (the HIMEM driver saves 64K bytes of conventional memory).
- The version of the expansion memory driver available. This should be EMS 4.0 to be in large frame mode.
- The type of expansion board used. It should support EMS 4:0 to be in large frame mode.
- The amount of expanded memory available.
- Whether backfilling is in use. It should be in use to be in large frame mode. Backfilling is the ability to replace the upper 256 or 512K bytes of the CPU memory by an equivalent amount on the expansion memory board. This is mandatory to enable windows to set the bank line (also called the EMS line) below 640K bytes. In this situation, the memory from 0 to the EMS line is considered to be shared, and the memory from the EMS line to 640K is used to bank expanded memory.

Taking all these parameters into account, the memory organization shown in Fig. 11 allows the HP OpenView DTC Manager to run correctly.

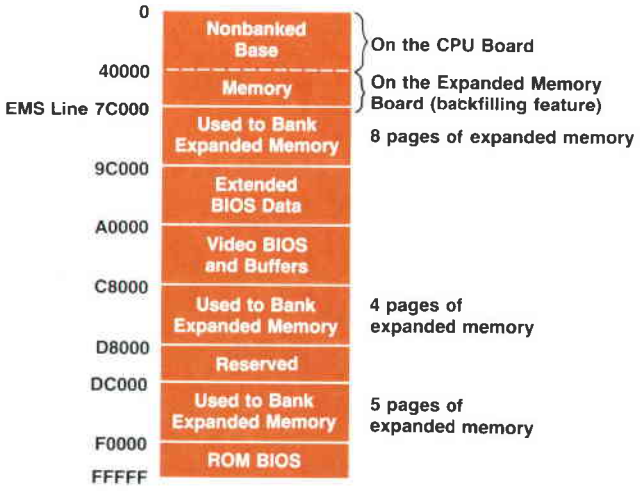


Fig. 11. *HP Vectra ES personal computer memory map for the HP OpenView DTC Manager.*

Summary

A new generation of HP 3000 computers needed a new generation of network management. The HP OpenView DTC Manager provides easy-to-use graphical interfaces, allowing the user to have comprehensive knowledge of the network and its components.

The important new shared services provided by the DTC also needed a management station that was independent of the host. The HP OpenView DTC Manager workstation provides this. However, a host may continue to manage a DTC when it is not necessary to share its services.

Acknowledgments

The HP OpenView DTC Manager was a joint effort of HP's Information Networks Division (HP OpenView Windows and RL Switch/CA), Business Networks Division (PCMPRMP), and Grenoble Networks Division (DTCMGRUI and DTCMGRIO). We would like to take this opportunity to thank all the teams involved for their contributions. It was an outstanding example of teamwork among all of the marketing, QA, production, and lab departments.

References

1. M.J. Mahon, et al, "Hewlett-Packard Precision Architecture: The Processor," *Hewlett-Packard Journal*, Vol. 37, no. 8, August 1986, pp. 4-21.

2. J.R. Busch, et al, "MPE XL: The Operating System for HP's Next Generation of Commercial Computer Systems," *Hewlett-Packard Journal*, Vol. 38, no. 11, December 1987, pp. 68-86.

3. D.V. James, et al, "HP Precision Architecture: The Input/Output System," *Hewlett-Packard Journal*, Vol. 37, no. 8, August 1986, pp. 23-30.

4. S. Youssef-Digaleh and A. Garg, *Network Management Architecture and Protocol*, Hewlett-Packard internal publication, Revision 2.2, September 1986.

5. C. Brunet, et al, *DTC Network Management Internode Specifications*, Hewlett-Packard, October 1988.

6. G.W. Lum, et al, "Expanded Memory for the HP Vectra ES Personal Computer," *Hewlett-Packard Journal*, Vol. 39, no. 6, December 1988, pp. 57-63.

Developing a Distributed Network Management Application Using HP OpenView Windows

Using concepts from the HP OpenView architecture and the facilities provided by HP Openview Windows, network management services and distributed applications were developed for user feedback and validation of the architecture.

by Atul R. Garg and Lisa M. Cole

THE HP OPENVIEW NETWORK SERVICES MONITOR (OV/NS Monitor) provides network management functions for distributed HP 3000 computers. OV/NS Monitor is divided into two parts: the main application and the user interface. The main application resides on an HP 3000 computer that is designated as a management node and the user interface resides on an HP Vectra personal computer. The main application performs network management functions via the software residing on the HP 3000 computers designated as managed nodes. OV/NS Monitor is for internal use only and is not available as a product.

This article describes the approach used to develop the OV/NS Monitor network management application using some of the concepts from the HP OpenView architecture and the facilities provided by the HP OpenView Windows software. Many of the ideas and concepts used in developing this application are being incorporated in the develop-

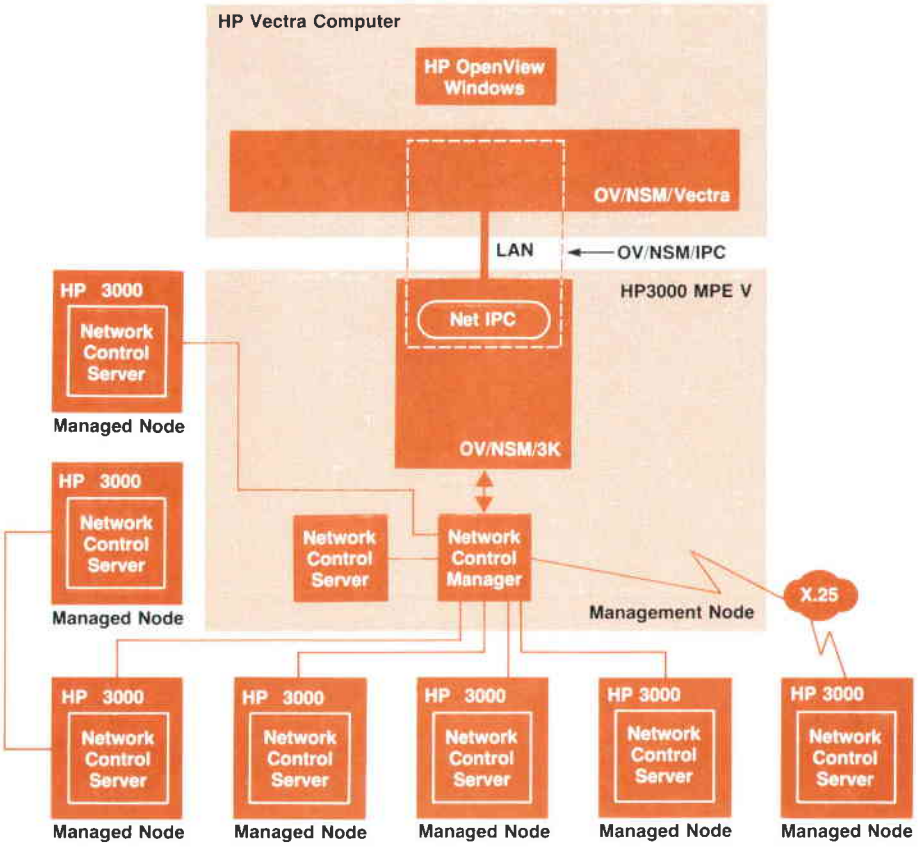


Fig. 1. *The main components of the HP OpenView Network Services Monitor (OV/NS Monitor).*

ment of other integrated and distributed network management products.

System Overview

A typical network consists of many nodes or computer systems. One (or more) of the nodes in the network is designated as the management node, and the other nodes in the network are referred to as managed nodes. The set of managed nodes can be connected to the network management node by a LAN, a set of point-to-point links, an X.25 link, or a gateway. Fig. 1 shows this configuration and some of the major OV/NS Monitor modules.

HP OpenView NS Monitor/Vectra (OV/NSM/Vectra). This portion of the OV/NS Monitor runs on the HP Vectra personal computer and uses the HP OpenView Windows utilities to interact with the user and IPC software to communicate with the portion of the OV/NS Monitor running on the HP 3000. OV/NSM/Vectra's function is to provide the user interface.

HP OpenView NS Monitor/3000 (OV/NSM/3K). This portion of the OV/NS Monitor runs on the HP 3000 computer that is designated as a management node. This portion of the OV/NS Monitor interfaces with OV/NSM/Vectra to service user requests, and it interacts with a module called the network control manager to retrieve network information from the managed nodes.

HP OpenView NS Monitor/IPC (OV/NSM/IPC). This software provides transparent interprocess communication between OV/NSM/Vectra and OV/NSM/3K. The IPC mechanism uses HP OfficeShare on the Vectra, and NetIPC[1] on the HP 3000 to provide reliable communication between the HP Vectra personal computer and the HP 3000.

HP OpenView Windows. HP OpenView Windows runs on the Vectra and provides a set of utilities and functions to display application menus and interact with the user. It provides a common user interface for all network management functions.

Network Control Manager (NCM). The network control manager runs on the management node and handles communication with the network control server on the managed nodes.

Network Control Server. The network control server runs on the managed nodes and performs functions on the managed node in response to requests from the management node.

HP OfficeShare. This application provides communication services between HP Vectra personal computers and HP 3000 systems.

The architecture of the OV/NS Monitor permits multiple management nodes to control overlapping areas of the network—that is, two or more management nodes can receive data from the same node. The design also permits several Vectras to connect to the same management node and multiplex the communications in an orderly manner. Each Vectra can connect to a management node via a LAN, a direct connection, or a modem.

OV/NSM/Vectra requires a connection to the management node. If the user tries to perform a function that requires the use of the management node and the node is not already connected, a logon dialog box is displayed and the user is given an opportunity to connect to the management node. HP OpenView Windows does not provide any security of its own and expects each application to implement the required level of protection. In OV/NSM/Vectra, users are authenticated as they try to log on to the management node. As part of authenticating a user, each user is given a level of security clearance at the management node. Three capability levels are recognized: reader, writer, or super user. If a user does not have sufficient capability for a particular menu item it is not selectable. This is done by a Microsoft Windows concept called graying out (dimming) the menu item.

The following sections describe the overall design of OV/NSM/3K and OV/NSM/Vectra in more detail and provide some insight into the operation of these applications.

HP OpenView NS Monitor/3000

To provide an extensible architecture that enables new distributed management applications to integrate easily with HP OpenView NS Monitor, a modular approach was employed to design the OV/NSM/3K modules. For the high-level design, HP Teamwork/SA was used to create data flow diagrams and to validate the consistency of these diagrams. For low-level design, finite state machines were used to design each individual module, and in some cases, state tables were used to settle several design issues. Since OV/NSM/3K is modular, a message-based interprocess communication (IPC) mechanism was adopted because it proved to be more flexible and easier to use than other alternatives, such as procedural IPC. Pseudocode and text specifications were also used to help promote and explain the design and interfaces between the modules.

Operation of OV/NSM/3K

OV/NSM/3K is started by running a stream job after the network control manager and network control server software have been successfully started. The stream job sets up some file equations and logging options, and then starts the Monitor process, which in turn starts other OV/NSM/3K processes. Once initialization is done, a message is printed on the operator console indicating success or failure of the start-up process.

OV/NSM/3K is stopped by executing a UDC (user-defined command) script. When this is done a HALT message is sent from the Monitor process to all of its child processes. Each process, after receiving the HALT message, is responsible for cleaning up before exiting.

Before any messages can flow between the Vectra and the management node, a connection must be established between OV/NSM/Vectra and OV/NSM/3K. A connection is initiated when the user logs on to OV/NSM/Vectra. OV/NSM/Vectra formats the logon request and sends it to OV/NSM/3K. After validating the request, OV/NSM/3K sends a confirmation logon back to OV/NSM/Vectra that indicates the connection is established and more requests can be sent from the Vectra to the management node.

When the user invokes a network management service (e.g., Try Connection), the request is sent to OV/NSM/3K, which tries to establish a connection with the network control server on the targeted (managed) node. If the connection is successfully established, the request is sent to

the network control server on the target node. After processing the request, the target node will send back a response which is forwarded to the user on the Vectra via OV/NSM/3K.

As shown in Fig. 2, OV/NSM/3K consists of five processes. The Monitor, Status, and Opt Logger processes are required to be running at all times. The Diag and Config processes are started when the user logs in and are terminated when the user logs off. All of these processes use a message-based IPC mechanism to communicate with each other.

Monitor Process

The Monitor process is the parent process of the other processes in OV/NSM/3K. It is started by the stream job described earlier. During initialization it starts the Status and the Opt Logger processes. The Monitor's primary function is to receive requests from any OV/NSM/Vectra trying to establish a dialogue with the Config or Diag processes. After verification of the user password, the Monitor process spawns the Config or Diag processes and passes the information to them so that they are able to communicate with OV/NSM/Vectra. Fig. 3 shows the data flows for setting up a connection to a Config process. If there is a status or configuration change made, the Monitor process will broadcast the change to all the active Diag processes. Regardless of the number of Vectras running OV/NSM/Vectra, there is only one Monitor process running at all times.

Status Process

The Status process has two principal tasks: maintaining the state of each object that can be displayed on the network topology map and acting as a centralized collector of remote messages. The Status process changes the state of an object based on events received from local or remote nodes.

Anytime a node changes state, the Status process sends a message to OV/NSM/Vectra (via the Monitor process) and writes a message to the event log. The Status process also conveys the state of all nodes to a Diag process on request.

The Status process's second task of acting as a centralized collector of remote logging messages involves logging only those messages that it considers critical enough to be of interest. The Status process writes these messages to log files, which are protected from being written to by any other process. Some messages that are sent to these files are generated by the Status process itself in response to time-outs or other local events.

Opt Logger Process

The Opt Logger is the process that collects network performance data. Its basic functions are to request the network control server on a target node to start a data collection run, stop a running data collection run, and collect the data produced by a data collection run. A data collection run is both the process of collecting network performance data and the storing of that data in a TurboImage data base.

Diag Process

The Diag process provides the diagnostic functions for troubleshooting a specific network problem. The Diag process is started by the Monitor process when a valid logon

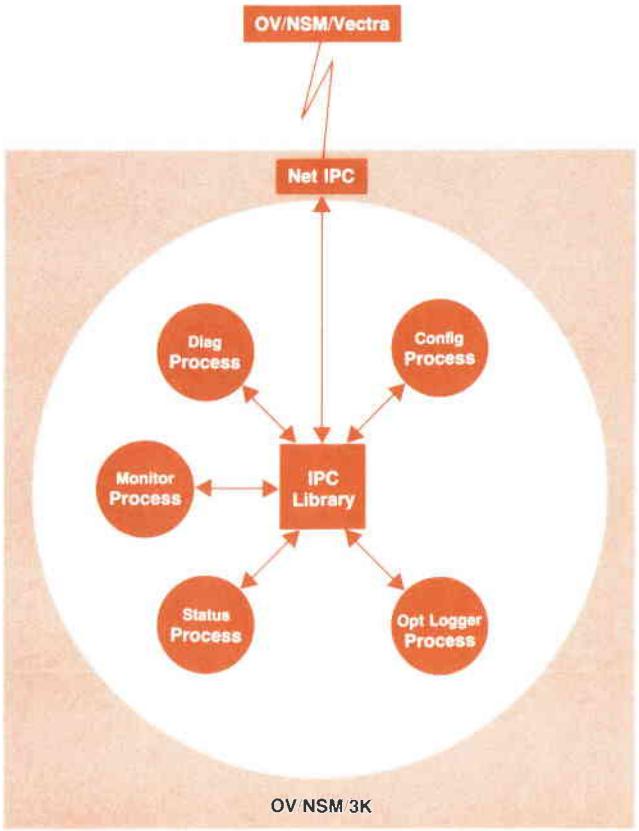
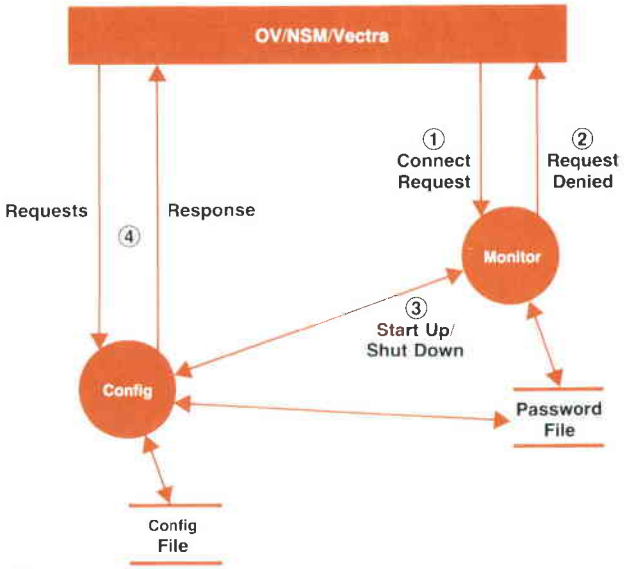


Fig. 2. *The main components of the HP OpenView Network Services Monitor/3000 (OV/NSM/3K).*



① Logon
② Invalid User
③ Successful Logon
④ Direct Communication with the User and OV/NSM/3K after Logon

Fig. 3. *OV/NSM/3K* Monitor *process interactions with the* Config *process during initialization and shut down.*

request is received from OV/NSM/Vectra. If the Diag process does not encounter an error, it will send a logon confirm message to OV/NSM/Vectra, thus establishing a connection. After this, all messages to or from OV/NSM/Vectra will go through the Diag process.

There can be multiple instances of the Diag process running at one time—one per Vectra connection. Since the Diag process is started by the Monitor process, it can also be brought down by the Monitor process via the HALT message. The Diag process can also terminate if it detects its connection with OV/NSM/Vectra is broken or it receives an EXIT message from OV/NSM/Vectra when the user logs off.

Config Process

The Config process implements OV/NS Monitor configuration functions that include user management and network topology file management.

The Config process is started up by the Monitor process when it receives a valid logon with a request to start the Config process. Only one Config process can be active on the management node at a time. The Config process can be shut down in the same manner as the Diag process.

Interprocess Communication for OV/NSM/3K

All OV/NSM/3K processes use a set of IPC library routines to send and receive messages to and from each other, to set up a timer that causes a process to be notified when the timer expires, and to interface to NetIPC for communication between QV/NSM/3K and OV/NSM/Vectra and the communication ports on the HP 3000. The IPC library provides a uniform IPC interface and centralizes the IPC related code. Collecting all the IPC in one place enables the underlying IPC mechanisms to be changed without affecting the other processes and having the same IPC interface in all modules made it possible to integrate the OV/NSM/3K modules with little difficulty. The IPC library also minimizes the processing overhead that takes place on the Vectra side of OV/NS Monitor. This objective is achieved by using a fixed-size header in every message. Thus, the address of where function dependent data begins in a message can be easily determined by adding the fixed number of bytes representing the size of the message header to the address of the first byte of a message.

HP OpenView NS Monitor/Vectra

The OV/NSM/Vectra software is an integral part of OV/NS Monitor, yet it is distinctly different from the OV/NSM/3K portion of OV/NS Monitor in a few key respects. First, OV/NSM/Vectra runs on a single-user personal computer running MS-DOS®, Microsoft® Windows, and HP OpenView Windows. OV/NSM/3K operates and was developed on a multiuser HP 3000 business computer running the MPE V operating system. Second, OV/NSM/Vectra is the user interface portion of the OV/NS Monitor, while the OV/NSM/3K software is used to monitor and control the nodes being managed by the user on the Vectra. Because of these differences, distinct requirements were developed for the OV/NSM/Vectra software.

The basic strategy used to provide for extensibility and changes resulting from user feedback was to modularize the architecture based on the natural divisions present in OV/NSM/Vectra applications. While doing this, the constraints imposed by both Microsoft Windows and HP OpenView Windows had to be accommodated.

OV/NSM/Vectra is required to handle two main interfaces: the user interface and the interface to HP 3000 software. The interface between the Vectra software and the HP 3000 software is message-based and the interface between OV/NSM/Vectra and the user is arranged in terms of dialog boxes. Thus, the basic design clusters the parts for a particular activity into one module to create a specific interface to the user. This includes the dialog box and its associated resources, the dialog box procedure, and the message interface to the HP 3000. This arrangement is shown in Fig. 4. Modules are as independent as possible, and each module assumes very little of its environment. Global state data is reduced to a minimum and, where applicable, is manipulated by access procedures rather than directly by the various modules.

To help meet the goal of timeliness, code reuse was employed and implemented in two ways: linkable libraries and extensive use of code templates. Code that is commonly used by many modules was put into linkable libraries to reduce code space use. For modules that were similar in structure but differed slightly in specifics, templates were created. Dialog box procedures are the best examples of this. All the OV/NSM/Vectra dialog box procedures have the same basic structure, differing only where required to perform the unique task or service provided by that box. This increased code size but leveraged large amounts of design and coding effort.

Wherever possible, an attempt was made to exploit the features provided by Microsoft Windows. For example, Microsoft Windows provides natural and easy ways of creating object classes called subclasses. OV/NSM/Vectra used this feature extensively to leverage the amount of original coding that needed to be done for the user interface.

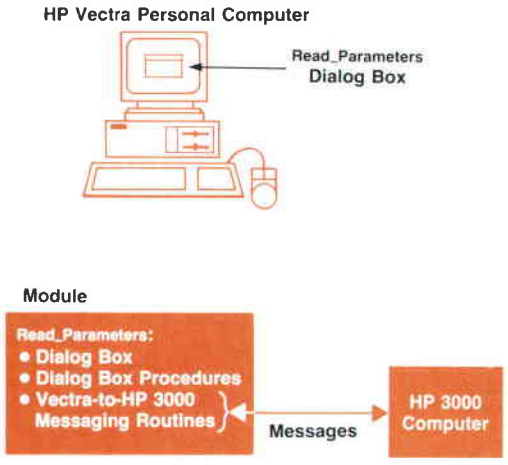


Fig. 4. *The design strategy for modularization in OV/NSM/Vectra for a particular user activity. The dialog box, the dialog box procedures, and the Vectra-to-HP 3000 routines are combined into one module.*

Microsoft and MS-DOS are U.S. registered trademarks of Microsoft Corporation.

OV/NSM/Vectra Structure

Each of the OV/NSM/Vectra applications consists of three separate processes: OV/NSM/VectraRun, OV/NSM/VectraDraw, and OV/NSM/VectraAdmin. This set of processes, together with the OV/NSM/3K processes, provides the OV/NS Monitor network management facilities.

The division of functionality between the three processes is based on the guidelines recommended in the *HP OpenView Windows Developer's Kit Writer's Style Guide*. A common application structure for developing the three processes was created without regard for the functional differences between them. There are three areas of interaction in this common application structure with which OV/NSM/Vectra is concerned: interacting with the user, interacting with the management node, and interacting with the network.

Interacting with Users

Interacting with the user requires OV/NSM/Vectra to interface with both Microsoft Windows and HP OpenView Windows, with the majority of the interaction taking place directly with Microsoft Windows.

Microsoft Windows Interface. Most of the OV/NSM/Vectra tasks are simple Microsoft Windows applications. They use the intrinsics provided by the *Microsoft Windows 2.0 Software Developer's Kit*, and the standard Microsoft Windows objects such as dialog boxes, menus, and icons.

The most common operation performed by OV/NSM/Vectra is to solicit user input and display user output using dialog boxes. The dialog box is used as a unit of modularization in OV/NSM/Vectra. That is, a dialog box, along with its dialog box procedure, IPC routines, and printing routines, forms a dialog box module. Each dialog box module is responsible for handling the requests and responses between the Vectra and the management node. It does this using the services of the transaction manager module and the IPC interface, which are described later. Fig. 5 shows this structure.

Sometimes there was a need to modify the default behavior of Microsoft Windows as in the case of the predefined window classes called controls. An example of a control is a single-line edit field that allows an application to display information to the user and receive input of any type from the user. The properties of controls can be changed by a process known as subclassing, which allows the application to inherit the current set of properties associated with a control and modify those properties to create a new type of control. An example of a subclassed control is the password subclass, which is a single-line edit control that allows the user to input only alphanumeric characters, the first of which must be an alphabetic character. This subclass does not echo the keyboard input to the user.

OV/NSM/Vectra uses subclasses frequently. The benefits of subclassing to OV/NSM/Vectra are twofold. First, OV/NSM/Vectra adds syntax checking to all of its new controls. Thus, syntax errors are detected as soon as an incorrect character is entered rather than resorting to primitive, forms-based methods of reporting errors. Second, the use of subclasses allows OV/NSM/Vectra to add or remove functionality without having to create new controls from scratch, reducing the amount of time required for design, coding, and testing of new code.

Some of the features of Microsoft Windows did present large design and implementation problems during the development of OV/NSM/Vectra. One of these problems is that Microsoft Windows provides no true communication capabilities of its own, nor does it provide an interface to any of the commonly accepted methods of communication for personal computers (e.g., NetIPC or NetBIOS). Integrating one of these communication mechanisms into a Microsoft Windows application proved to be quite a challenge and is described in more detail in the section on interacting with the network.

HP OpenView Windows Interface. To operate in the HP OpenView Windows environment with other applications, there are certain standard Microsoft Windows functions that applications ask HP OpenView Windows to perform on their behalf, such as adding menus to the menu bar. This gives HP OpenView Windows some measure of control to present a consistent look and feel to the HP OpenView environment. Applications use the intrinsics provided by the HP OpenView Windows developer's kit to access these normal Microsoft Windows features, as well as some administrative tasks required to operate as an HP OpenView Windows application.

HP OpenView Windows applications are unlike normal Microsoft applications in some respects. One difference is that the main window for an application (the window most applications create when they come alive) is invisible. This window has zero coordinates and is used for taking advantage of the messaging facilities of Microsoft Windows. It is known as a communication window and is created by making a call to HP OpenView Windows. The actual window displayed on the screen with the map and menus is wholly owned and operated by HP OpenView Windows. Since Microsoft Windows associates a message queue with an

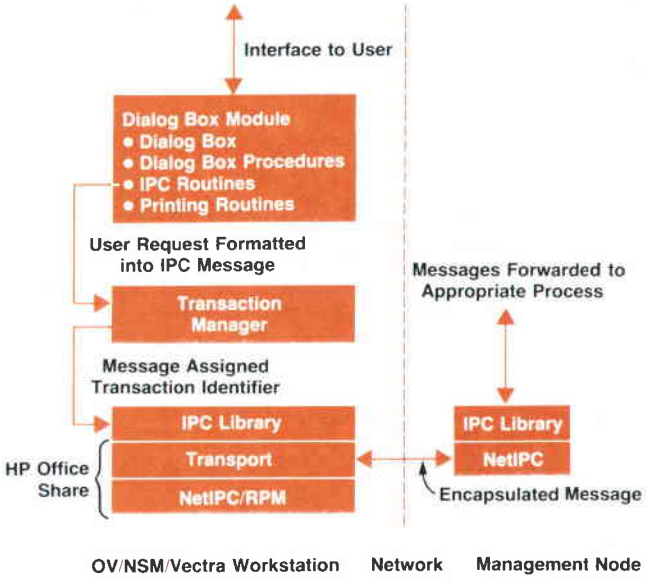


Fig. 5. *The modules involved in providing interaction with the user and interaction with the network in the OV/NSM/Vectra software.*

application's main window and posts all messages destined for an application to this queue, all messages resulting from user interaction with the map and menu items are received by HP OpenView Windows. HP OpenView Windows then forwards messages that arrive in its queue to the appropriate application's communication window for subsequent processing.

Interacting with the Management Node

The bulk of the OV/NS Monitor resides on the HP 3000 management node. The OV/NSM/Vectra applications establish communication with the management node using TCP/IP via the NetIPC interface. All communications to and from the management node are in one of the predefined OV/NS Monitor message formats. The normal mode of operation is request/response oriented, with all requests being initiated from the OV/NSM/Vectra side. These requests are received and processed by OV/NSM/3K and the results are formatted and returned to OV/NSM/Vectra.

This appears to be straightforward. However, a fundamental design feature of OV/NSM/Vectra is that multiple requests can be outstanding on the management node. OV/NSM/Vectra cannot, therefore, block on any individual reply. A method of pairing requests with responses and associating these request/response pairs with their originating dialog boxes had to be developed. To complicate matters further, there are a few asynchronous event messages that originate on the management node, such as the notification of the shutdown of the OV/NSM/3K software. These asynchronous messages have no associated dialog box, but need to be handled in as efficient a manner as any other response. Satisfying these requirements placed some additional constraints on the design of the network interface.

Interacting with the Network

OV/NSM/Vectra handles the receipt and sending of packets across the network by encapsulating network packets as Microsoft Windows messages to a dialog box procedure. This interface is defined and supported by two modules: the IPC library and the transaction manager shown in Fig. 5.

The IPC library is a thin layer of software that interacts with the HP OfficeShare communications software. HP OfficeShare is divided into two layers: the transport layer and the NetIPC/RPM layer. The transport layer supports either an HP ThinLan link or a serial link. Based on the physical connection between the Vectra and the management node, one of these transports must be loaded into memory by the user before running Microsoft Windows. Only one type of link can be loaded at a time, and it resides in Vectra EMS memory.[2] OV/NSM/Vectra assumes nothing about the transport except that it expects it to be present. Since the transport software is sitting outside of Microsoft Windows, there has to be a way of requesting data to be sent on the link. This is provided by a dynamic library included in the NetIPC/RPM development package. This library provides a standard NetIPC interface for use with Microsoft Windows-based applications. This is the only communications interface OV/NSM/Vectra deals with.

Like the NetIPC interface library used by OV/NSM/3K, OV/NSM/Vectra's IPC library provides a set of procedures for accessing the NetIPC interface. These access procedures hide the complexities of connection establishment, termination, and other network operation from the rest of the application, thereby allowing them to make simple open, close, read, and write calls to the network.

The transaction manager is responsible for creating and destroying transactions associated with dialog boxes as well as distributing packets from the network along with a transaction identifier to the correct dialog box. Each dialog box procedure requests a transaction from the transaction manager when it has a request to send to the management node. The transaction manager associates this transaction, identified by a unique transaction identifier, with the dialog box handle. The packet is then sent to the management node. The response to this request is also identified by the transaction identifier. At some point the response packet is retrieved from the network by the IPC library module. It is given to the transaction manager module, which then looks at the transaction identifier and posts the message to the dialog box registered for this transaction. The dialog box procedure then processes the message as it processes all Microsoft Windows messages, completely unaware that the message actually came from the network.

Combining the Interfaces

When HP OpenView Windows is started from Microsoft Windows, the user is actually executing one of the HP OpenView Windows processes OVRun, OVDraw, or OVAdmin (see article on page 60). These processes will spawn all other Run, Draw, and Admin applications, respectively. For the OV/NS Monitor these include OV/NSM/VectraRun, OV/NSM/VectraDraw, or OV/NSM/VectraAdmin. The list of applications that are spawned is kept in the Microsoft Windows initialization file WIN.INI along with the location of the user's default map, the NewWave help files[3] used by all HP OpenView Windows applications, and the time and date localization information.

When actions performed by the user on the HP OpenView Windows map are of interest to OV/NSM/Vectra, HP OpenView Windows forwards a message to OV/NSM/Vectra's communication window. In the order received, these messages are appended to the queue and are processed, along with any other Microsoft Windows messages posted directly to the queue.

Once the user's selection has been passed to OV/NSM/Vectra, HP OpenView Windows fades into the background and OV/NSM/Vectra makes calls directly to Microsoft Windows to display dialog boxes, solicit user input, and display results. OV/NSM/Vectra requests are packaged into IPC packets and sent to the management node. When the responses are returned from the management node, they are received by the HP OfficeShare transport. When the network polling mechanism in OV/NSM/Vectra discovers that a packet has been received from OV/NSM/3K, it copies the packet contents into its own data space. The transaction manager then reads the packet header to locate the transaction identifier to determine which dialog box should receive the packet. Once it knows where to send it, the transaction manager encapsulates the packet into a Microsoft Windows message and posts it to the dialog box that originated the request. Once this message is received by the procedure associated with the dialog box, the message is

disassembled and its results are displayed in the dialog box display area.

If the transaction manager discovers, after reading the packet header, that the packet contains an asynchronous message rather than a response to an earlier dialog box request, the packet is encapsulated in a Microsoft Windows message and passed to the OV/NSM Vectra asynchronous message handler for further processing.

Conclusion

Developing the HP OpenView NS Monitor applications helped to identify the common functions that are required to provide a framework for the development of distributed and integrated network management functions. We found that it was essential to provide a common interprocess communication facility between the user interface, the management node, and the managed nodes. In addition, validation of users and their access rights, handling of events and status monitoring, and data storage and retrieval must be designed and developed uniformly to support the needs of all applications that are to be integrated.

Acknowledgments

We wish to acknowledge the technical accomplishments and contributions of the other members of the HP OpenView NS Monitor project team: Tin Phan, Chandramohan Thekkath, Kumar Vora, Richard Bogen, John Hardin, Wilson Ang, Lisa Gullicksen, and Jim Schnitter. Special recognition should also be given to Tony Ridolfo, the original project manager for the HP OpenView NS Monitor project. In addition, because of the integrated nature of the prototypes that were being developed, we would like to thank all the members of the HP OpenView Windows and HP OpenView core software development teams, as well as all the members of the network management product team for their valuable contributions, without which we could not have completed this project.

References

1. K.J. Faulkner, et al., "Network Services and Transport for the HP 3000 Computer," *Hewlett-Packard Journal*, Vol. 37, no. 10, October 1986, p. 14.

2. G.W. Lum, et al., "Expanded Memory for the HP Vectra ES Personal Computer," *Hewlett-Packard Journal*, Vol. 39, no. 6, December 1988, pp. 57-63.

3. V. Spilman and E.J. Wong, "The HP NewWave Environment Help Facility," *Hewlett-Packard Journal*, Vol. 40, no. 4, August 1989, pp. 43-47.

Authors

April 1990

6 Modular Liquid Chromatograph

Herbert Wiederoder

Herbert Wiederoder served as both section manager and project manager for the HP 1050 liquid chromatograph project at the Waldbronn Analytical Division. Previously, he was a project manager for the HP 1090L liquid chromatograph and developed hardware and software for the HP 1090M and HP 1084B. Herbert joined HP in 1977, the year he graduated from the Technical University of Berlin, earning a diploma in electronics and computer science. He also is a graduate of Fachhochschule Ulm (1974) with a degree in engineering. A member of Gesellschaft für Informatik, Herbert's professional interests are centered around control systems and applications for analytical instruments. Born in Stuttgart, he resides in Spielberg with his wife and two children. He enjoys tennis, skiing, and traveling.

11 LC Quality Engineering

Helge Schrenker

Helge Schrenker was HP's Waldbronn Analytical Division quality manager during development of the HP 1050 liquid chromatograph system. He joined HP in 1973 as a product marketing manager. In 1979, he became an R&D group leader investigating flow control systems and doing systems integration for the HP 1090 high-performance liquid chromatograph. In 1983, he became a quality assurance manager. Helge earned an applied physics diploma (1967) from Kiel University. He has written several technical articles on pH measurement and control, concepts of fast-liquid chromatography, flow control in high-performance liquid chromatography, and the effect of mobile phase preheating on liquid chromatography performance. His work has resulted in patents on a flow-controlled high-pressure pump and a column thermostat for high-performance liquid chromatography. Before joining HP, Helge worked in technical marketing support for Philips Electronic Industries, and served in the German Air Force. Born in Marbach, West Germany, he and his wife live in Karlsruhe. His hobbies and interests include photography, bicycling, environmental protection, and alternative traffic concepts.

Wolfgang Wilde

Wolfgang Wilde performed reliability engineering tests during development of the HP 1050 liquid chromatograph system. He is currently the quality engineering manager for HP's Waldbronn Analytical Division. He earned a diploma (1986) in physics from the University of Mainz in West Germany, and joined HP in 1987 as a reliability engineer.

17 LC Sample Injector/Autosampler

Gerhard Ple

Gerhard Ple served as project leader for the autosampler for the HP 1050 liquid chromatograph system and was responsible for HP 1050 firmware development. In a prior HP position, he developed electronic hardware and firmware for the HP 1090 autoinjector, autosampler, and pump configuration. Gerhard joined HP's Waldbronn Analytical Division in 1979, after receiving his electrical engineering diploma from the University of Karlsruhe that year. He was born in Neustadt, West Germany, and currently resides in Karlsruhe.

Wolfgang Kretz

R&D project engineer Wolfgang Kretz served as project leader for the mechanical design of the HP 1050 liquid chromatograph autosampler. He also was project leader for development of the autoinjector for the HP 1090 liquid chromatograph, developed improvements for the HP 79841A injector system, and served as a production engineer for the HP 1090. Wolfgang received his mechanical engineering degree (1972) from the Fachhochschule Konstanz, and an engineering diploma from the University of Stuttgart in 1978, the year he joined HP. Born in Buchheim, West Germany, Wolfgang now lives in Waldbronn with his wife and two children. His hobbies include photography, reading, hiking, and astronomy.

24 LC Solvent Delivery System

Klaus Witt

Klaus Witt contributed to the design of the pump driver and control system for the HP 1050 liquid chromatograph system. He is now a group manager within R&D. He joined HP in 1979 as an R&D engineer in the Waldbronn Analytical Division, shortly after graduating from the Fachhochschule Osnabrueck with an electronics diploma. At HP, he helped design the digital servo chip for the high-performance liquid chromatograph. He is named an inventor in a patent application for a variable-stroke pump. Born in Nedlin, Germany (now Poland), Klaus served 15 months in the West German Army, and now resides in Keltern, Germany, with his wife and two children. His hobbies include raising horses and horseback riding.

Fred Strohmeier

Mechanical engineering and hydrodynamics are the professional interests of Fred Strohmeier, section manager with HP's Waldbronn Analytical Division. He helped design the HP 1050 and HP 1090 liquid chromatograph systems, and is currently the project leader for the HP 1050 pumping system. Fred, who joined HP in 1979, is named an inventor in patent applications for a pumping apparatus to deliver liquid at high pressure and for a sample injector for liquid chromatography. He received his engineering diploma in 1979 from the Fachhochschule Karlsruhe. Born in Baden-Baden, Fred resides in Rheinmuenster, West Germany, with his wife and child. He enjoys jogging, canoeing, and reading.

36 LC Absorbance Detectors

Axel Wiese

A year after earning his PhD degree in physics and physical chemistry in 1977, Axel Wiese joined HP's Waldbronn Analytical Division. He developed the HP 79854A multiwavelength detector for the HP 1050 liquid chromatograph system. Before that, he designed the HP 79881A filter photometric detector and the HP 1046A fluorescence detector. Now a program manager, Axel has authored several technical papers in the field of microwave instrumentation. He resides in Karlsruhe, West Germany, and his hobbies include traveling and archeology.

Konrad Teitz

Project manager Konrad Teitz helped develop the HP 79853A variable wavelength detector, which is part of the HP 1050 liquid chromatograph system. In the past, he has developed other detection systems for the liquid chromatograph instruments produced at the Waldbronn Analytical Division. He joined HP in 1973 in Böblingen, West Germany, and is named an inventor in a patent concerning

a new type of ultraviolet absorbance detector. His professional interests include electronic circuit simulation and EMC. Konrad, who received an engineering diploma in 1973 from the University of Karlsruhe, West Germany, was born near Lippstadt, and now resides in Karlsbad with his wife and son. He enjoys backpacking, photography, and amateur radio.

Günter Höschele

Günter Höschele developed data processing firmware for the HP 1050 liquid chromatograph system. He also helped develop the HP 1040A detector and HP 79881A detector data acquisition hardware and firmware. Günter joined HP in 1979, two years after graduating from Stuttgart University with an electronics diploma. An R&D engineer with HP's Waldbronn Analytical Division, his professional interests are fast data acquisition and processing. Born in Stuttgart, Günter currently resides in Langensteinbach, West Germany. He enjoys reading, high-fidelity music, and bicycle riding.

Volker Brombacher

Volker Brombacher contributed to the development of the analog-to-digital converter for the multiwavelength detector and the I/O and diagnostic firmware for the HP 1050 liquid chromatograph system. He is currently an R&D project leader in the spectroscopy section of the Waldbronn Analytical Division. Volker earned an engineering diploma in electronics from the Technical University of Karlsruhe in 1985, joining HP shortly after graduation. He was born in Pforzheim, West Germany, and resides in Pfinztal with his wife and two children. Volker's hobbies include photography, an aquarium, motorbiking, meditation, and yoga.

Hubert Kuderer

Hubert Kuderer developed the multiwavelength detector for the HP 1050 liquid chromatograph system. He has also designed front-end electronics for the HP 1040A detector, and worked in quality assurance and manufacturing engineering. After receiving an engineering diploma from the Fachhochschule Offenburg, West Germany, in 1978, he joined HP. He has authored technical journal articles on spectograph sensitivity, photodiode array spectrometers, and photodiode architecture, and he is named an inventor in two patents in PDA readout technology. Born in Offenburg, Hubert resides in Waldbronn with his wife and two children. He enjoys tennis, skiing, and woodworking.

44 LC Firmware Development

Christian Büttner

Soon after he received his engineering diploma in electronics (1981) from Fachhochschule Esslingen, Christian Büttner joined HP's Waldbronn Analytical Division in 1981. He designed and implemented firmware for the HP 1050 liquid chromatograph system. He has also designed firmware for the HP 1090 local user interface, and for the HP 1040 detector. As a project manager, he currently is working on HP 1050 communications and firmware enhancements. Christian was born in Stuttgart, West Germany, and now lives with his wife and two children in Waldbronn. He enjoys table tennis, bicycling, and amateur theater.

Fromut Fritze

After studies at the University of Karlsruhe in computer science, and graduation from the Fachhochschule Karlsruhe (1982) in electronic engineering, Fromut Fritze joined HP in 1982. He designed operating system software and tools for the HP 1050 liquid chromatograph system. Before that, he designed hardware and software for the HP 1090 liquid chromatograph system, a remote control standard, and Pascal ChemStation software. Now a project leader, he is working on a revision control system and software quality. His professional interests center around operating systems and object-oriented techniques. Born in Heidelberg, West Germany, he and his wife live in Karlsruhe. His hobbies include photography, traveling, and desktop publishing.

Gerhard Ple

Author's biography appears elsewhere in this section.

51 HP OpenView Network Management

Anthony S. Ridolfo

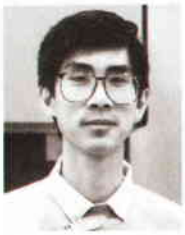
Born in Great Falls, Montana, Tony Ridolfo received his mathematics education at Wabash College (BA degree in 1966), Ohio University (MS in 1968), and Iowa State University (PhD in 1975). After teaching mathematics and computer science as a professor at Murray State University in Kentucky, he joined HP in 1976 and developed firmware for a number of HP's calculators. He also served as a project manager for the HP 75C computer's operating system. Tony coauthored an HP Journal article on the HP 75C in 1983. As a project manager on the HP OpenView project, he started development of HP OpenView Windows and later led the development of the OpenView NS performance monitor. He is a member of the Society for Industrial and Applied Mathematics, and the American Mathematical Society. Tony lives in Saratoga, California, with his wife and three teenage children, is active in the Boy Scouts and as a referee in youth soccer. His leisure activities include golf, skiing, and bridge.

54 HP OpenView Architecture

Keith S. Klemba

Information networks are the primary professional interest of Keith Klemba, a member of the Information Networks Group technical staff who helped develop the HP OpenView network management architecture. His other professional interests include broadcast and space information networks. He joined HP's Network Architecture Lab in 1986. Before joining HP, he was a product manager with Vitalink Communications Corporation in Fremont, California, and a systems engineer and technical director with SRI International in Menlo Park, California. He received an AA degree (1970) in computer science from DeAnza College. Keith authored a technical paper on HP OpenView architecture, published in the 1989 proceedings of the IFIP. In 1967, he served a tour of duty in Vietnam with the U.S. Army. He is a member of the IEEE, and a commissioner for the police athletic league's girls' softball for the city of Santa Clara. Married and the father of two children, Keith was born in Chicago, Illinois, and now lives in Santa Clara, California. His hobbies include raising 4-H guide dogs for the blind.

Hui-Lin Lim



As a member of the technical staff in the Network Architecture Laboratory in HP's Information Network Group, Hui-Lin Lim helped to develop and refine the HP OpenView network management architecture. He joined HP's Singapore Network Operations group in 1988. Before that, he developed office systems and wrote programs for the National Computer Board in Singapore. His professional interests include the UNIX operating system and personal computers. Hui-Lin earned a BSc degree (1983) in computer science from Queen Mary College at the University of London in the United Kingdom. Born in the Republic of Singapore, he lives in Santa Clara, California, with his wife and child. He enjoys photography and science fiction.

Maureen C. Mellon

Maureen Mellon is the project manager of a group responsible for developing HP OpenView network management architecture. Maureen joined HP in 1989 and managed four network architecture specialists on this project. She received a BSEE degree (1975) from Villanova University in Villanova, Pennsylvania, an MSEE degree (1976) from Drexel University in Philadelphia, Pennsylvania, and performed additional graduate work towards a PhD (1989) at the University of California at Los Angeles. Before joining HP, Maureen worked on ISDN and broadband ISDN for AT&T Bell Laboratories in Holmdel, New Jersey. An editor of CCITT recommendation Q.714 on signaling system No. 7, and coauthor of a book on economic characterizations of large telephony networks, Maureen is a member of the IEEE, and Eta Kappa Nu and Tau Beta Pi societies. She was born in Darby, Pennsylvania, and lives in Cupertino, California, with her husband. She enjoys skiing, tennis, running, and hiking.

Mark L. Hoerth

Mark Hoerth is the product manager for the HP OpenView network manager server on the HP-UX operating system. After joining HP in 1987, he served as the product manager for instrument support at HP's Product Support Division. He received a BS degree (1985) in electrical engineering from the University of Nebraska, and an MBA degree (1987) from Stanford University. Mark was born in Rapid City, South Dakota, and now resides with his wife in Stanford, California. His hobbies include photography, swimming, racquetball, and running.

60 HP OpenView Windows

Arthur J. Kulakow

Arthur Kulakow developed the graphics library and the map routines used in HP OpenView Windows. He is now researching methods that will enable software develpers to use C++ in the development of network management platforms. Before joining HP in 1985, Art was a software engineer with Bell Laboratories in Naperville, Illinois, and at Digital Research Company in Monterey, California. He earned a BS degree (1980) and MS degree (1982) in computer science from the University of Wisconsin in Milwaukee. He is a member of ACM and his professional interests include user interface design, object-oriented programming, and computer graphics. Born in Milwaukee, Wisconsin, Art lives in San Jose, California. He enjoys scuba diving, hiking, camping, and aerobics.

Kathleen L. Gannon

Kathleen Gannon worked as an R&D software engineer in the development of the HP OpenView Windows/MS-DOS system. Before that, Kathy was a systems engineer at HP Labs. She began her HP career as a product marketing engineer for cooperative support products in 1980. She earned a BS degree (1980) in industrial engineering from Northwestern University in Evanston, Illinois, and an MS degree (1985) in electrical engineering from Stanford University. Born in Pittsburgh, Pennsylvania, Kathy is married and lives in Santa Clara, California. She has been an active member of the board of directors of the Santa Clara Valley Science and Engineering Fair for the past five years. Her hobbies and interests include needlework and science fiction.

Catherine J. Smith

Catherine Smith is the R&D project manager for the HP OpenView Windows/MS-DOS product. Prior to that, she served as project manager for the HP MPE XL terminal subsystem. Catherine joined HP in 1978, soon after she graduated from the University of California at Berkeley with a BS degree in electrical engineering and computer science. Born in Yakima, Washington, she resides in San Jose, California. Catherine is married and has one son in college. Her hobbies include downhill skiing and hiking.

66 HP OpenView BridgeManager

Tamra I. Perez

Developing firmware and network software are Tamra Perez' major professional interests. She developed the network interface for the HP Bridgemanager project, and is now designing a user interface to manage HP's new high-speed and remote bridges. After joining HP in 1986 in the Roseville Networks Division, she codeveloped a wire test tool for HP StarLAN-1 products. Before that, Tamra worked in IBM's disk technology division as a summer intern for two years. She earned her BS degree (1986) in computer engineering from San Jose State University, and will receive her MS degree in computer engineering from UC Davis in June 1990. Tamra was born in San Jose, California, and lives in Roseville. She is a member of the Sweet Adelines women's singing quartet, a flute player, and a member of the Tau Beta Pi Engineering Honor Society.

Andrew S. Fraley

After designing and implementing LAN hubs, PC LAN cards, and drivers, Andrew Fraley worked on the design and development of the HP BridgeManager's user interface. Andy joined HP's Roseville Networks Division as a design engineer in 1986, soon after earning a BS degree (1986) in computer science from the Massachusetts Institute of Technology. Born in Columbus, Ohio, he and his wife reside in Citrus Heights, California. Andy enjoys soccer, skiing, tennis, and guitar.

71 HP OpenView Data Line Monitor

Michael S. Hurst

Michael Hurst joined HP in the Queensferry Telecom Division in Scotland shortly after he received his Bachelor of Science degree with honors (1978) in computer science from Edinburgh University in Scotland. Mike was the project manager for the HP OpenView data line monitor software. In the past, he designed microprocessor hardware and firmware for the HP 3779A/B and HP 3776A/B primary multiplex test sets, and HP 1000 application software for the HP 37100S remote access and test system. He is currently directing new developments for the HP 37100S system. Mike is a member of the ACM and the British Computer Society, where he serves on the Edinburgh Branch Committee. Born in Wiltshire, England, he resides in Edinburgh, Scotland, where he enjoys skiing and running. One of Mike's ambitions is to spend two weeks skiing in Colorado.

76 HP OpenView DTC Manager

Serge Y. Amar

As the technical leader in the development of the HP OpenView Datacommunications and Terminal Controller (DTC) manager, Serge Amar was responsible for its overall design and implementation. He is now a project manager for the OpenView DTC manager. Before that, Serge was a foreign service employee with HP's Information Networks Division in Cupertino, California. He developed the DTC management module for the first release of MPE XL. Prior to joining HP in 1983, he designed electronic credit cards for home banking for Honeywell Bull Co. Serge received his diploma in electronics (1979) from Orsay University south of Paris. Born in Colomb-Bechar, Algeria, he now lives in Grenoble, France, with his wife and two children. His professional interests include personal computers and network management, and his leisure time activities include skiing and diving.

Michele A. Prieur

Michele Prieur helped develop the HP OpenView Datacommunications and Terminal Controller (DTC) manager. Joining HP's Grenoble Terminal Division in Grenoble, France, in 1980, she worked as a development engineer for the HP 2392A terminal and as a technical leader for HP AdvanceLink. She is now a project manager in the development of network management systems. She received an engineering diploma (1980) from the Ecole Supérieure d' Electricité. Born in Lyon, France, Michele and her husband and child now reside in Grenoble. She enjoys horseback riding, reading, and sports.

85 Network Management Application

Lisa M. Cole

A graduate of Merrimack College in North Andover, Maine, Lisa Cole earned a BS degree (1983) in computer science. After joining HP in 1987 as a design engineer, she worked on the Vectra portion of the HP OpenView NS Monitor, and is now developing security software for the communications portion of OpenView. Previously, she worked on network management with Digital Equipment Corporation, on communications applications and SNA with Wang Laboratories, Inc., and on compilers with Honeywell Information Systems. Lisa's professional interests include network management and standards-based communications. A native of Salem, Massachusetts, she lives in Scotts Valley, California. She and her husband are expecting their first child in June. Lisa enjoys skiing, travel, and handicrafts.

Atul R. Garg

A project manager for the HP OpenView distributed network management services, Atul Garg has worked in the area of network protocols and architecture since he joined HP in 1981. He worked on HP AdvanceNet and on HP's design of the internet architecture and IEEE Standard 802.3. He also contributed to the development of the distributed NS monitor applications under OpenView. Atul coauthored an article on HP AdvanceNet in the HP Journal in October, 1986, and published another article on LAN internet protocols for a Midcon conference in 1982. He earned a bachelor's degree (1979) in electrical engineering from the Indian Institute of Technology in Kanpur, India, and an MS degree (1981) in electrical engineering from the University of Hawaii in Honolulu. Born in New Delhi, India, Atul lives in Santa Clara, California, with his wife and daughter. His hobbies include badminton and bridge.