Network Management

1 Abstract

DECnet/OSI Phase V incorporates a new network management architecture based on Digital's Enterprise Management Architecture (EMA). The EMA entity model was developed to manage all entities in a consistent manner, structuring any manageable component regardless of its internal complexity. The DNA CMIP management protocol was developed in conjunction with the model to express the basic concepts in the entity model. Phase V network management is extensible; the Phase V management architecture transparently assimilates new devices and technologies. Phase V was designed to be an open architecture. Management of DECnet/OSI Phase V components is effective in a multivendor network.

Network management has been an integral part of DECnet since 1976 when Phase II was developed.[1] Even at that early stage of the DECnet architecture, an effective management capability was recognized as an essential part of an organized approach to networking. Now in DECnet Phase V, the DECnet network management architecture has undergone a major revision based on Digital's Enterprise Management Architecture (EMA). This paper gives an overview of some of the key features and functions of EMA and of DECnet Phase V network management. See the "Overview of Digital's Open Networking" paper in this issue for an overview of the guiding principles, background, and architecture of DECnet Phase V.[2]

Our initial work on Phase V indicated that changes were needed in the network management architecture to support the broad range of networking functions planned for Phase V. First, network managers would have to be able to manage all the Phase V components in a consistent manner. A method was needed to build Phase V management components that would give the same general look and feel and the same modeling approach to all components.

Second, Phase V network management would have to be extensible. The Phase V network architecture was being designed to allow the use of multiple modules that would provide the same or similar services at each layer and to simultaneously support multiple-layer protocols in a network. Therefore, we designed the Phase V management architecture to transparently assimilate new devices and technologies. Our management architecture had to become as extensible as the network architecture.

Finally, since Phase V was designed to be an open architecture, management of Phase V components would have to be effective in a multivendor network. Our design had to ensure that the ability to provide effective management of network components was independent of the vendors supplying them.

The individual management mechanisms used in Phase IV could have been extended to accommodate all the changes planned for Phase V. However, we felt it was time to revisit the basic network management architecture to see if we could find a unified approach that would provide a superior solution.

2 Enterprise Management Architecture

We began our Phase V development project by examining in detail the requirements for a new network management architecture. Our goal was to design an open architecture that allowed for consistent management of an extensible array of network components in a multivendor environment. As we identified the specific requirements that would have to be addressed to meet this goal, we realized that we had the opportunity to develop a architecture that went beyond management of Phase V networks. We realized that we could provide an architecture for the management of both networks and systems. The architecture eventually became known as the Enterprise Management Architecture or EMA.

Early in the project, we recognized that the conceptual separation of manageable components from the software that manages them was a fundamental design principle. EMA therefore distinguished entities, the basic components of the network that had to be managed, from directors, the software systems and accompanying applications used by managers to manage the components, as shown in Figure 1.

Formally, an entity was further split into a service element, a managed object, and an agent. The service element is the portion of the entity that performs the primary function of the entity, e.g., a data link layer protocol module whose primary purpose is communication with a peer protocol module on another machine. The managed object encapsulates the software that implements the functions supported by the entity for its own management. For example, it responds to management requests for the current values of state variables or to requests for the values of certain configuration variables to be set to new values. The agent is the software that provides the interface between the director and the managed object. The agent encodes and decodes protocol messages it exchanges with the director and passes requests to and receives responses from the managed object.

Informally, we generally equate the managed object and the entity because the managed object defines what the manager can monitor and control in the entity.

A director was modeled as a layered software system that provides a management-specific environment to management applications. A director was split into a framework, a management information repository (MIR),

and separate configurable software modules called management modules. The director kernel provides common routines useful for the layered software modules, including services such as dispatch (location-transparent exchange of management requests and responses with entities), encoding

/decoding, data access, data dictionary access, and event management. Taken together, the director kernel and the agent provide a framework for managed objects and management applications to interact. The framework provides an application programming interface (API) to managed object and management module developers. The MIR contains data about particular entities as well as information about the structure and other properties of entity classes, which the director software also knows.

Management modules were distinguished as presentation, function, or access modules. Presentation modules implement user or software access to the director management modules that is device independent and style dependent. Function modules provide value-added management functions that are partially or completely entity independent, such as network fault diagnosis, event or alarm handling, or historical data recording. Access modules provide a consistent interface to the basic management functions performed by entities. In addition, they include one portion that maps operations on entities into the appropriate protocol primitives and another portion that implements the protocol engine for the relevant management protocol. Figure 2 shows the components of a director and an entity.

Although users can conveniently interact with systems through graphical user interfaces (GUIs), sophisticated users wished to preserve a command line interface (CLI) they could use to specify complex management requests quickly. Therefore, we developed a single, extensible command language that would allow human operators or software programs to communicate requests to management modules and (ultimately) entities in a consistent fashion. This work developed into the network control language (NCL). An NCL command specifies an entity, an operation to be performed by the entity, a list of arguments (if any), and a list of qualifiers (for specifying users, passwords, paths, filtering values, etc.).

Digital's DECmcc Management Director is an implementation of an EMA director.[3] The DECmcc product provides a platform for the development of new management capabilities and offers specific Phase V management capabilities as well as a number of generic network management tools. The DECmcc director supports both GUI and NCL CLI user interfaces.

Entity Model

To manage all entities in a consistent manner, we required a single, consistent method for structuring any manageable component (regardless of its internal complexity) and for describing its management properties: the operations that it can perform, the variables it makes available for its management, the critical occurrences it can report to managers, etc. The EMA entity model was developed to answer these needs. The structure of a manageable component in this model is shown in Figure 3. Essentially, the entity model defines techniques for specifying an object-oriented view of

an entity. Each entity has the following properties:

- A position within an entity hierarchy. To ease management of networks with large numbers of complex components, entity classes are organized into logical structures that reflect the relationship of their

corresponding components; individual entities are named in terms of that structure. The name of the top-level entity in each structure is globally unique, and it is referred to as a global entity. All its child entities, however, have names that are unique only within the context of their level in the structure. Therefore, they are referred to as local entities.

- A hierarchically structured name. An individual entity's local name is constructed by concatenating its class name to its instance identifier. The class name is a keyword that uniquely identifies the class (object type) of an entity. The instance identifier is the value of an identifying attribute used for naming instances of the entity's class, for which each instance of the class has a unique value.

  A target entity's globally unique name is constructed by concatenating its local name (a <class name, instance identifier> pair) to the local names of each of its ancestors in turn, beginning with the containing global entity and ending with the target entity's immediate parent. The construction of an entity's name and the containment hierarchy are shown in Figure 4.

- A collection of internal state variables, called attributes, that can be read and/or modified as a result of management operations. Attributes have names unique within the context of the entity. Attributes have a type that defines the values the attribute can have.

- A collection of operations that can be performed by the entity. Operations allow managers to read attributes, modify attributes, and perform actions supported by the entity. Actions are entity-specific operations that result in changes of state in the entity or cause the entity to perform an operation that has a defined effect.

- A collection of events that can be reported asynchronously by the entity. An event is some normal or abnormal condition within an entity, usually the result of a state transition observed by its service element or its agent. Event reports are sent asynchronously to the manager; they indicate the type of (entity-specific) event that occurred and may also contain arguments that further describe or qualify the event. For example, arguments could indicate the number of times the event occurred before a report was sent to announce that a threshold was reached, or give the old and new states in an event that reports a state transition.

- A specification of the behavior of the entity in relationship to the functions that the entity's service element provides. This is usually specified as some abstract state machine, through pseudocode, or as a set of preconditions, postconditions, and invariants.

The entity model provides specific requirements and recommendations about the way entities can be modeled in terms of these properties. These restrictions, placed on entity class definitions for purposes of both internal and global consistency, take several forms: (1) restrictions on the types and ranges of attributes that can be used for various purposes (e.g., as identifying or counter attributes); (2) constraints on

operations (e.g., examine operations can have no side effects on the value of attributes whose values they report); or (3) restrictions on events (e.g., all events and event reports must have an associated time stamp and unique identifier).

Readers familiar with open systems interconnection (OSI) management will find the entity model very similar to OSI's structure of management information (SMI) standard.[4,5] This is no coincidence. During the early development of Phase V and the entity model, we recognized the need for an open management architecture. Portions of the technology were therefore contributed to ISO/IEC JTC 1 SC21/WG4, a working group of the International Organization for Standardization (ISO) that is responsible for efforts to define standards for OSI management. Although some details of OSI SMI and the corresponding EMA features diverged slightly from each other during their evolution, the EMA entity model and OSI SMI are still compatible. At this writing, work is under way to align certain parts of the EMA entity model with the final international standard (IS) versions of OSI SMI.

Entities

The EMA entity model describes how to specify the management of an architected subsystem. However, for Phase V, we chose to make the management specification of a subsystem a part of the subsystem's specification. As described in the Modules section, that may have been the most important decision made in the network management architecture.

As the entities for DECnet/OSI Phase V were defined, a collection of folklore grew on how typical design issues could or should be solved. As with any folklore, these guidelines were passed from one architect to another, either verbally, or as selected portions of the management specifications were copied from one subsystem to another. This folklore is continually changing, as new and better solutions are found. Much of the folklore has already been described.[6] Some other guidelines are described below.

The Network Management Specification describes the central structure of Phase V network management, and in particular defines the node entity class.[7] In the following sections, we describe the properties of the node entity class and, as a representative example, the OSI transport module entity class.

3 Node Entity Class

A single computer system in the DECnet/OSI network is called a node. The bounds of that system depend on the system's architecture; a personal computer (PC), a single-processor workstation, a multiprocessor mainframe, a diskless system, even a VAXcluster system can be considered a single

node. Nodes are modeled by the node entity class.

A node entity has only a few functions in management.

- A node is a global entity that is the parent for many subsystems and provides an agent for all of them.
- A node has an identity, a name, and an address that allow it to be managed remotely.
- A node plays a major role in system initialization and start-up.

Identity

The following attributes identify a node:

- An address, the application layer address(es) of the node's agent
- A name, a DECdns fullname as defined by the DECnet/OSI distributed name server[8]
- A synonym, a Phase IV-style node name for backward compatibility
- A spatially unique identifier (ID), a 48-bit quantity used as an Institute of Electrical and Electronics Engineers (IEEE) 802 local area network (LAN) or Ethernet address
- A space- and time-unique value

A node's address is the application layer address(es) of the node's agent. The DECnet/OSI network supports multiple protocols at any of the seven layers, and the agent can operate over multiple protocol stacks. Each protocol has its own addressing conventions. Thus a node's address is actually a set of protocol towers. Each tower defines a sequence of protocols, each with its associated addressing information. A protocol tower provides all the information needed by a director to connect to the node's agent and to issue management directives to the node or any of its children.

Users and network managers rarely refer to nodes by their addresses. First, it is difficult to remember the addresses and second, moving the node from one place to another in the network generally changes its address. Thus each node has a name, a DECdns fullname. The node knows its name and address. Each node's name is stored as a DECdns entry, and one of the entry's DECdns attributes holds the node's address. Thus, any director can look up the node's name in the DECdns and the address associated with it, and then use any one of the towers to connect to the node's agent.

To ensure backward compatibility with DECnet Phase IV, a node also has

an attribute called its synonym, which is a six-character, Phase IV-style node name. If a node has a synonym name, that name is entered in a special directory in the DECdns name space as a soft link to the node's Phase V name. A soft link is a form of alias or indirect pointer, from one name to another, that allows an entry to be reached by more than one name.

Each network layer address of the node (a node can have more than one) is encoded in a standard way as a soft link to the node's name. This allows a manager (or director) to translate a node address into the equivalent node name, making many diagnostic problems much simpler.

DECnet/OSI includes many features that allow most nodes to autoconfigure their addresses. Network layer addresses consist of an area address and a 48-bit ID. This ID can be obtained from an ID read-only memory (ROM) chip on many devices (for example, each Digital 802.3 LAN device has one). End nodes detect area addresses from messages sent by the routers adjacent to the end node. Higher-level addresses used by management are architecturally defined constants.

Managers and users choose the name and synonym of a node. The manager uses the rename action to tell the node its name. Rename is an example of a situation in which an action is more appropriate than a set operation. Renaming a node is a fairly complicated operation. Not only is the name attribute changed, but also the information is stored in the DECdns name space. Although the operation can fail in many ways, actions allow errors to be reported to the manager with enough detail on what went wrong to allow corrective action to be taken. This is not easily done with a set operation.

One of the more difficult configuration problems to track down occurs when two nodes in a network have either the same name or the same address. DECnet/OSI has several management features to prevent this from occurring or to detect the situation when it does occur.

First, each node has a spatially unique 48-bit ID, i.e., no two nodes in the enterprise have the same ID at the same time. The ID is usually derived from an ID ROM chip in a LAN adapter. Special manufacturing procedures ensure that no two ID ROMs hold the same ID. Nodes with multiple ID ROMs, for example a router with two Ethernet interfaces, choose one with a simple algorithm. Nodes without an ID ROM must be assigned an ID when the system is first booted, and that ID must come from the locally administered IDs. However, an ID is not always tied to the same node. Hardware devices can be removed from one machine and inserted in another. Indeed, this is a common diagnostic procedure.

Second, each node has a space- and time-unique value provided by the unique identifier (UID) service. UIDs combine a spatially unique ID with a time stamp in such a way that no two generated UIDs will ever have the same value.[9] The UID is stored in nonvolatile storage (if the node has some), so the UID remains constant across system reboots. Nodes without nonvolatile storage will generate a new UID on every reboot.

Third, a change in the name, address, ID, or UID attributes is reported

by the node as an event, which aids in detecting duplicate node names and addresses. Two nodes can end up with the same name when the disk where a node stores its system image, name, address, and UID is copied, and then the copy is booted on another machine. When the disk is booted on the second machine, that machine would have a different ID ROM. The node would

detect that its ID is different, and thus an event would be generated. The event would not prevent the duplicate node from booting, but it would allow the manager to detect that a duplicate node may be on the network.

Start-up

A node is responsible for system start-up. We model start-up through four states.

- Dead, when the node is down and requires manual intervention to start.

- Booting, when the node is in the initial stages of software start-up. The booting process is highly system specific and may be initiated by hardware, by software, by a power failure, or by a manager's console request. Booting loads a system image, starts it running, and brings it to a known state. The system image can be loaded from a disk or equivalent storage, or it can be loaded over the network using the maintenance operations protocol (MOP) down-line load protocol.[10] MOP is layered directly over the data link protocols. In Digital's communications devices, MOP is generally implemented in the hardware or firmware and does not require a working operating system.

- Off, when the node is initializing itself and its internal configuration. When booting completes, the node changes to the off state. This transition is called the "big bang." In the first instant after the big bang, the node has at least the following things available, as shown in Figure 5:

  - A working clock and time service used to time stamp events.

  - A UID generator used to give entities and events a unique identifier.

  - The node entity (and possibly some of the node's child entities) together with its agent (which includes both the directive dispatcher and event logging).

  - An initialization script, a series of management commands to configure the system. This can be in the form of a text NCL command file (described later in the section on NCL), or it can be a compiled script, one that has been encoded as a series of common management information protocol (CMIP) requests. MOP can be used to down-line load an initialization script.

  - An initialization director, which reads the script and invokes the directives in the order given. Errors and other output may be displayed on a console (if the system has one) and/or reported as events.

- On, when the node has "completed" initialization to the extent that it can be managed remotely. Somewhere in the initialization script (probably near the end), the node is enabled, which changes its state to on, i.e., it can be managed remotely.

Modules

A node has many subsystems, called modules in DECnet/OSI. Each module may or may not be configured within any particular node. Within the modules are the various subsystems that make up DECnet/OSI. A node never has more than one instance of a module contained within it. A general-purpose node allows the manager to flexibly configure a node to serve a particular purpose by creating and deleting the appropriate modules.

In the DECnet/OSI Phase V network, the specification of the management of each module is an integral part of the architecture of the subsystem. Moving responsibility for the management of a subsystem from a central network management architecture to the subsystem architecture has made the specifications clearer and more complete. In Phase IV, a great deal of effort was spent coordinating the subsystem specifications and the network management specification. Placing responsibility in one person's hands made writing an internally consistent subsystem much easier. Besides, the sheer size of DECnet/OSI Phase V management would have made it impossible for a single person to design the management of the whole system.

The development of the OSI management standards in ISO/CCITT (Comité Consultatif Internationale de Télégraphique et Téléphonique) has been done in a similar way and for the same reasons. ISO/IEC JTC1 SC21/WG4 is the group that has developed the OSI management information model, management specification language, and guidelines for module developers. While SC21 /WG4 has itself also developed the management of specific subsystems (e.g., for event forwarding and logging), typically, the job of doing this has been left to other groups more expert in particular areas. For example, Working Groups 1, 2, and 4 of ISO/IEC JTC1 SC6 have developed management standards for the ISO data link, network, and transport layers, based on Digital's contributions derived from the DECnet/OSI Phase V work in these areas.

In DECnet/OSI, the transport, network, and data link subsystems were among the first to have the EMA concepts applied to their management. Others quickly followed and, presently, more than 50 modules have been specified, with others being added as new subsystems are designed. Not surprisingly, during the early days considerable interaction took place between the architects responsible for the central network management architecture and those responsible for developing the management of specific subsystems. The EMA evolved and was refined based on the experiences of the many subsystem architects using it.

In almost all cases, modules contain one or more entities, each representing some management aspect of the subsystem. These entities in turn may contain other entities (subentities). This nesting can occur to an arbitrary depth, reflecting the management complexity of the subsystem.

Note that modules themselves are entities, albeit with the restriction that a node never has more than one instance of a module contained within it. An entity is formally described using Digital's Management Specification Language (MSL).[11]

We next consider in more detail the structure and contents of the DECnet /OSI Phase V OSI transport module. Complete descriptions of this and other Phase V subsystems can be found in the Digital Network Architecture (Phase V) Documentation Kits.[12,13,14,15]

4 OSI Transport Module

In DECnet/OSI Phase V, the OSI transport module contains port, template, local network service access point (NSAP) address, and manufacturing automation protocol (MAP) entities. A local NSAP entity contains remote NSAP entities. The containment hierarchy is shown in Figure 6.

The OSI transport module has characteristic attributes. A manager can change the configuration of the module by modifying its characteristic attributes. This is done for several reasons, including

- To limit the maximum permissible number of active transport connections at any one time
- To control the maximum credit window that may be granted on an individual transport connection
- To control the maximum number of transport connections that can be multiplexed on any single network connection, when the OSI transport protocol is operating over the connection-mode network service

Modification of these attributes is needed only if the manager requires anything other than a standard configuration; working default values are defined for all characteristic attributes.

Status attributes show the current operating state of the module, e.g., the number of transport connections currently active. Status attributes cannot be modified directly by a manager. To start the operation of the OSI transport module, the manager uses the enable action. If successful, the state attribute changes from off to on.

In the DECnet/OSI Phase V architecture, a port entity represents the interface between layers, making visible to a manager how one layer (a client) is using the services of a lower layer. Ports are not created by a manager; they are created when a client of the service requests use of the service (by "opening a port"). The exact information held in a port entity varies for each subsystem. In general, a port entity contains attributes that identify the client and the service being used, and how that service is being used (e.g., as usage counters). The port entity is an example of how the EMA evolved through feedback from the subsystem architects. Before being adopted as a general mechanism in the overall management architecture, the concept was first developed and used in subsystem

architectures.

In the case of the OSI transport module, the port entity also corresponds to the local end of a transport connection (TC), and it provides a window to the status information associated with the TC. For example, the OSI transport port status attributes give

- The name of the user of the OSI transport service
- Local and remote NSAP addresses and transport selectors
- The protocol class being operated on the TC

In addition, a port entity has counter attributes that record the total number of times something of interest occurred on the TC. For example, there are counters recording the number of octets and protocol data units (PDUs) sent and received. A management station can poll these and determine usage over time. A port entity also maintains counters for both duplicated transport PDUs (TPDUs) detected and retransmitted TPDUs. Taken with the usage counters, these can be used to calculate error ratios and rates on the TC.

When a client opens a port onto a service, the client can then use the service interface to select options such as which features to use or which profiles. Maximum flexibility, however, also poses a problem. In many cases, a client has little or no knowledge or understanding of the service options available in an underlying layer. Further, it would be unrealistic to expect all clients of a service (or, ultimately, an end user) to acquire this in-depth knowledge.

One alternative was to provide default values for all the service options. However, a single set of default values satisfies only a single subset of uses. Instead we adopted the template, which is an entity that represents a set of related option values. A manager can create as many templates as required for different sets of related option values. A client needs to be configured only with the single name of the template to use, not the details of every service option. The OSI management standards groups have adopted the template concept in the form of their initial value managed object (IVMO).

A template in the OSI transport module is a collection of characteristic attributes used to supply default values for certain parameters that influence the operation of a TC. When a port is opened to the OSI transport service, a template name may be specified by the client. The characteristic attributes in the template are then used as default values for TC parameters not supplied by the user, including, for example,

- The value of the window timer

- The set of classes of protocol that may be negotiated for use on a TC
- The use of checksums that might be negotiated for a TC that operates the class 4 protocol, a variant of the OSI transport protocol defined in ISO 8073

A default template is automatically created and used if no template is specified when a port is opened.

There is one local NSAP entity for each NSAP address used by the OSI transport. A local NSAP entity is automatically created when an NSAP address used by the OSI transport is added to the network routing subsystem (the adjacent lower layer).

The remote NSAP entity is a subentity of a local NSAP entity. Each remote NSAP entity maintains counter attributes resulting from interactions between the superior local NSAP and a remote transport service provider. Events are defined for the remote NSAP entity, to provide immediate notification to the manager of error conditions. For example,

- A checksum failure event occurs whenever checksum validation fails when performed on a received TPDU

- An invalid TPDU received event occurs whenever a TPDU received from the remote NSAP is in violation of the transport protocol

Consider this second example. Whenever an invalid TPDU received event is generated, a counter is incremented. Thus, even if the manager has configured event logging to filter out these events, an indication that they are happening remains, prompting the manager to change the filtering criteria. The event contains a number of arguments as well. All events identify the generating entity and the time the event occurred. The invalid TPDU received event also has arguments that give

- A reason code, indicating in what way the TPDU was invalid, as specified in the ISO 8073 standard[16]

- The part of the TPDU header that was invalid

- A specific Digital Network Architecture (DNA) error code, which was added to qualify the ISO 8073 reason code and to help customers diagnose problems

The MAP places a number of requirements upon implementations of the OSI transport protocol beyond simple conformance to ISO 8073. The MAP entity contains the additional management needed to meet these extra requirements. The MAP entity is optional; implementations with no business requirement to support MAP would not provide the MAP entity.

5 Supporting Mechanisms

Network management in DECnet/OSI is built on a number of supporting services. Wherever possible, management uses the services of the network

to manage the network. This approach minimizes the number of special mechanisms we had to define specifically for network management. Some key services used by network management include

- Session control
- DECdns name service

- Digital's distributed time service (DECdts)
- A unique identifier service (UID)

A few services were developed specifically to support network management. Most had existed in earlier phases of DNA.

- DNA CMIP
- Event logging
- MOP down-line load protocol
- Application loopback

In the following sections, we describe DNA CMIP and event logging.

Digital Network Architecture Common Management Information Protocol

The entity model describes what an entity can do. Those concepts must be expressed in the management protocol. DNA CMIP, the management protocol for DECnet/OSI Phase V, is an evolution of the Phase IV management protocol (called NICE). The two protocols are remarkably similar. Both include the set, show (also called get), and event report operations. The main differences between the two protocols are in the following areas.

- Treatment of other operations. In NICE, each operation required a new kind of message; in CMIP, a general extension mechanism, the action, is provided.
- Naming. NICE supported a limited number of entity classes (eight) and provided a rudimentary naming hierarchy based on the notion of "qualifying attributes." CMIP supports hierarchical entity names and is essentially unlimited in the number of entities with which it can deal. Similarly, CMIP is much more extensible in naming attributes, attribute groups, and event reports.
- Encoding. CMIP uses ISO Abstract Syntax Notation 1 (ASN.1), a standard tag, length, value (TLV) encoding of attributes and arguments, and NICE used a private TLV encoding.

DNA CMIP is not quite the same as the IS version of OSI CMIP, although it was based on the second draft proposal of the CMIP standard. There are two reasons for this.

- First and foremost was timing. DNA CMIP was developed before the OSI CMIP was standardized. The inevitable changes to the standard led to

many minor differences in the protocols. Still, because the concepts in the EMA entity model and OSI's SMI are aligned, the DNA and OSI CMIP protocols are fundamentally the same. The authors are currently migrating DNA CMIP to OSI IS CMIP. The change will be transparent to any user.

- Second, DNA CMIP operates over a DNA protocol stack, not a pure ISO stack. This allows directors on Phase IV systems to manage Phase V systems.

DNA CMIP can be viewed as two separate protocols. One protocol, management information control exchange (MICE), is used by a director to invoke a directive (get, set, action, etc.) on an entity (or entities). The other protocol, management event notification (MEN), is used by an entity (or entities) to report events to a director. The two protocols operate over separate connections for important reasons.

- The times at which the associations are connected differ. A MEN association is brought up when an entity wishes to report an event, and is thus controlled by the agent. A MICE association, however, is brought up when a director (or manager) wishes to invoke an operation on an entity, and is thus controlled by the director. Attempting to share control of association establishment was not worth the complexity.

- Whenever an association is shared by two different users, the problem of allocating resources fairly to the two users must be addressed. Since transport connections deal with this issue between connections, the addition of a multiplexing protocol at the application level (with an attendant flow control mechanism) was again considered to be too complex. Transport connections are not (or should not be) expensive.

Event Logging

The entity emits an event report to the manager when an event occurs in an entity. The event logging module provides a service that transmits event reports from the reporting entities to one or more sink applications, which are considered to be a certain kind of director in EMA. Event logging in Phase V is based on concepts similar to those provided by Phase IV. Because the principal use of event logging is for reporting faults, event logging does not guarantee delivery of event reports to the sink application. Figure 7 shows the event logging architecture.[17]

When an event occurs within an entity (E) in a source node, the entity invokes the PostEvent service provided by the event dispatcher (a part of the node's agent). When posting an event, the entity supplies its name, the type of the event, all the arguments related to the event, a time stamp of when the event occurred, and a UID assigned to the event. UIDs ensure that each event can be uniquely identified, so that if a sink application receives more than one copy of an event report, it can detect the duplication. Time stamps allow the event reports to be ordered in time (an important step in determining causality). A time service (DECdts) is used to synchronize clocks across the network. It provides a consistent view of time for correlating observations. An important feature for

management is the inclusion of an inaccuracy bound on the time stamp.

The PostEvent service formats an event report and places it in an event queue (Q). Event queues are limited in the amount of memory they use; thus they limit the number of events that can be held in the queue. Because events can be placed in the queue at a rate faster than the queue server (S) can process them, the queue can fill, and any new events placed in the queue will be lost. The events lost event is recorded as a pseudo-event in the queue (it appears as an event report from the entity holding the queue). The events lost event carries an argument that records the number of events that were lost in a row.

The queue server for the event dispatcher compares each event report against a filter (F) associated with an outbound stream. The filter lists a collection of entities and events that are either passed through the filter or blocked by the filter. Event reports passing through the filter are placed in an event queue within the outbound stream. Each outbound stream's queue server sends events to a corresponding inbound stream in the sink application. Multiple outbound streams can be set up by the manager, allowing events to be sent to many sink applications. Outbound streams are modeled as entities in their own right, and standard management operations (create, get, set) are used to configure them.

Each inbound stream in a sink application has an event receiver (R). Inbound streams are generally created when a connection request is received from an outbound stream. Events received by the receiver are compared against a sink filter and queued to the sink application. Thus the events from multiple inbound streams are merged.

The protocol used between the outbound stream and the inbound stream is the CMIP MEN protocol, which operates over a connection (using either the DECnet transport layer protocol or OSI transport). The use of a connection lowers the probability that an event report will be lost, since the connection handles acknowledgments and retransmissions. It does not guarantee delivery, however, and events may still be lost due to failures of the sink application or the source node.

6 Conclusions

Our approach to Phase V management worked well. Defining the EMA entity model first provided a framework of consistency among all the architectures. Developing a management protocol (CMIP) expressing the basic concepts in the entity model in conjunction with the model placed the protocol in a position to meet the needs of the model. Giving responsibility for defining the management of a subsystem to the architects of that subsystem made each subsystem more complete and coherent. As problems were found in the model based on lessons learned during the specification of entities, any needed changes to the entity model were applied to correct those problems.

However, some things did not go as well. The number of entities, attributes, and operations in Phase V was beyond anyone's expectations. This reflects the overall complexity and feature-richness of Phase V over Phase IV as well as the increased control that the manager is given. This burden is eased somewhat by the use of intelligent defaults, autoconfiguration, and self-management. Still, simplifying the management of a Phase V network is an important area for continual improvement.

The biggest success of EMA/Phase V management is its general applicability. EMA is being applied to more than the traditional network management areas. Systems, networks, and applications are all managed by EMA.

7 References

1. N. LaPelle, M. Seger, and M. Sylor, "The Evolution of Network Management Products," Digital Technical Journal, vol. 1, no. 3 (September 1986): 117-128.

2. J. Harper, "Overview of Digital's Open Networking," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue): xx-xx.

3. C. Strutt and J. Swist, "Design of the DECmcc Management Director," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue).

4. OSI Management Information Services-Structure of Management Information-Part 1: Management Information Model, ISO/IEC DIS 10165-1 (Geneva: International Organization for Standardization/International Electrotechnical Commission, 1990).

5. OSI Management Information Services-Structure of Management Information-Part 4: Guidelines for the Definition of Managed Objects, ISO/IEC DIS 10165-4 (Geneva: International Organization for Standardization /International Electrotechnical Commission, 1992).

6. M. Sylor, "Guidelines for Structuring Manageable Entities," Integrated Network Management I, B. Meandzija and J. Westcott (eds.), (Amsterdam: Elsevier Science Publishers, 1989): 169-183.

7. DNA Network Management Functional Specification, V5.0.0 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNA02-FS-001, 1991).

8. DNA Naming Service Functional Specification, V2.0.0 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNANS-FS-002, 1991).

9. DNA Unique Identifier Functional Specification, V1.0.0 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNA16-FS-001, 1992).

10. DNA Maintenance Operations Protocol Functional Specification, V4.0.0 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNA11-FS-001, 1992).

11. DECmcc System Reference Manual, 2 volumes (Maynard, MA: Digital Equipment Corporation, Order Nos. AA-PD5LC-TE, AA-PE55C-TE, 1992).

12. Digital Network Architecture (Phase V) Documentation Kit No. 1 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNAP1-DK-001, forthcoming 1993).

13. Digital Network Architecture (Phase V) Documentation Kit No. 2 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNAP2-DK-001, 1993).

14. Digital Network Architecture (Phase V) Documentation Kit No. 3 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNAP3-DK-001, forthcoming 1993).

15. Digital Network Architecture (Phase V) Documentation Kit No. 4 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNAP4-DK-001, 1993).

16. Information Technology-Telecommunications and Information Exchange Between Systems-Connection Oriented Transport Protocol Specification, ISO/IEC 8073 (Geneva: International Organization for Standardization /International Electrotechnical Commission, 1989).

17. DNA Event Logging Functional Specification, V1.0.0 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNA09-FS-001, 1992).

8 General References

EMA Entity Model (Maynard, MA: Digital Equipment Corporation, Order No. AA-PV7KA-TE, 1991).

M. Sylor, "Managing DECnet Phase V: The Entity Model," IEEE Networks (March 1988): 30-36.

S. Martin, J. McCann, and D. Oran, "Development of the VAX Distributed Name Service," Digital Technical Journal, vol. 1, no. 9 (June 1989): 9-15.

C. Strutt and D. Shurtleff, "Architecture for an Integrated, Extensible Enterprise Management System," Integrated Network Management I, B. Meandzija and J. Westcott (eds.), (Amsterdam: Elsevier Science Publishers, 1989): 61-72.

DNA Network Command Language Functional Specification, V1.0.0 (Maynard, MA: Digital Equipment Corporation, Order No. EK-DNA05-FS-001, 1991).

L. Fehskens, "An Architectural Strategy for Enterprise Network Management," Integrated Network Management I, B. Meandzija and J. Westcott (eds.), (Amsterdam: Elsevier Science Publishers, 1989): 41-60.

9 Biographies

Mark W. Sylor Mark Sylor is the manager of Digital's Enterprise Management Architecture Group. He is the author of the EMA Entity Model and the Phase V DECnet Network Management Specification. He was a member of the ISO and ANSI committees working on OSI system management and was the ANSI T5.4 ad hoc group leader on the structure of management information. Prior to this work, Mark was the principal designer and development supervisor for the NMCC/DECnet monitor. Mark joined Digital in 1979. He holds an M.S. in mathematics from the University of Notre Dame.

Francis Dolan Frank Dolan is a consultant engineer with Digital's Telecommunication Business Group Engineering in Valbonne, France. He is currently the project manager and technical leader of the GDMO translator, a tool being developed to support the DECmcc/TeMIP OSI access module and OSI agent presentation module. Prior to this work, Frank was the architect of several Phase V DNA specifications, including DDCMP network management, OSI transport, and network routing accounting, and was an active member of OSI management standards committees. He has filed a European patent application on message network monitoring.

David G. Shurtleff A member of Digital's Corporate Systems Engineering

Group, David Shurtleff consults in support of major systems integration projects and participates in CSE initiatives to improve engineering processes. Previously, he was a member of the EMA Architecture Group, where he worked on the specification of EMA director architectures and

the development of systems management standards. David has also worked in the DECmcc strategic vendor program as a senior technical resource. Before joining Digital in 1988, David was on the packet switch development staff at BBN Communications Corporation.

10 Trademarks