Interoperability Issues in Heterogeneous Network Management

Pramod Kalyanasundaram
Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716
(email: kalyanas@cis.udel.edu)

Adarshpal S. Sethi
Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716
Ph: (302) 831-1945
(email: sethi@cis.udel.edu)

Published in *Journal of Network and Systems Management* **Vol. 2 No. 2 (June 1994), pp. 169-193.**

Interoperability Issues in Heterogeneous Network Management

Pramod Kalyanasundaram Adarshpal S. Sethi

Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716.

March 24, 1994

Abstract

The presence of dissimilar network models and standards necessitates interoperability as a means of achieving ubiquitous connectivity and management. The primary focus of this paper is to identify interoperability issues, independent of the network management model, which form the basis for developing interworking paradigms. Network Management and Interoperability related concepts and terminology are introduced and a generic network management framework is presented. Based on the generic framework, interoperability issues for developing paradigms are identified systematically. Different paradigms for interoperability are described and compared. The techniques used by these paradigms in addressing the issues identified are discussed.

Keywords: Network Management, Interoperability, Management Framework, MIB mapping, Protocol Translation, Application Gateway.

1 Introduction

The evolution of computer networks was driven by the fundamental human need for communication and the promise of cost-effective resource sharing. But their very existence spawned a plethora of new applications that caused the phenomenal growth of networking witnessed in the past decade. The variety of applications linked with diverse user requirements inevitably

led to the construction of different types of networks with different functional specifications. Faced with the possibility of diverse heterogeneous and incompatible solutions, users and vendors alike favored standardization efforts that provided users with a wider choice of compatible products. Just as computer networks evolved parallelly in different parts of the world, so did the standardization process. This parallel process resulted in the creation of several standards with different communities adopting different standards.

The ultimate goal of computer networks is universal connectivity. In order to achieve this goal networks all over the world must be connected together. More importantly, these connected networks must work together in a cooperative manner to satisfy the end user needs. Ironically, the existence of multiple standards prevents the attainment of this goal. The concept of *interoperability* provides a possible solution to this problem. *Interoperability,* in the context of computer networks, can be defined as the ability of two or more connected, dissimilar networks to work together in a cooperative manner to provide a common set of services to users on any of the connected networks and facilitate sharing of network-wide resources.

The growth of user needs and advances in networking and computing technology have given rise to increasingly large networks. Demands on computing resources and the dependence of users on the continuous availability of such resources have increased dramatically during the last decade. Performance and reliability have become major concerns. Networks are now too complex to be easily managed by human managers alone. Network Management aids the task of human managers in achieving their complex, sometimes conflicting, goals by providing facilities for monitoring and controlling the operation of the network and its components.

Research in Network Management has made rapid progress over the past few years. Nevertheless, different network management models exist, with each model conforming to a different set of standards. The need of the day is to make these models interoperate so that artificial barriers that impede efficient network management are broken. Due to the heterogeneity inherent in the different networks, interoperability of management services of such dissimilar networks is an important problem that needs and is beginning to get serious thought.

In attempting interworking of dissimilar management models, it is imperative to identify and understand the issues involved in detail. Interoperability issues are currently being addressed based on the specifics of the models involved. From a research viewpoint it is desirable to adopt a conceptual approach (an approach that is model or implementation independent) in

identifying interoperability issues. This paper undertakes such a task. The primary focus of this paper is to define interoperability issues at a conceptual generic level instead of addressing them using the specifics of a model; to focus upon abstractions without delving into implementation details.

This paper presents a generic framework for network management with emphasis on interoperability. Existing models are shown as specializations derived from such a framework. Interoperability issues are identified and presented based on the framework. Such a framework lends itself to the identification of interoperability issues at a conceptual level. Since the interoperability issues are based on abstractions that transcend models they are applicable to the interworking of different models that conform to such abstractions. The use of this framework also helps to develop a methodical approach for defining interworking paradigms and techniques.

The organization of this paper is as follows: Section 2 introduces concepts and associated terminology related to Network Management and interoperability in the context of a generic framework. Section 3 discusses interoperability issues related to Network Management models using the generic framework. Section 4 surveys proposed and existing schemes for interworking different management models. It also evaluates the schemes vis-a-vis interoperability issues identified in the previous section. Finally, this paper concludes with trends and future work.

2 Overview of Network Management Concepts

Network Management is the ability to plan, configure and monitor a network in order to effectively utilize local and global network resources and to ensure reliable, secure operation of networks. Based on this definition, the scope of network management includes configuration, performance, fault, security and accounting management. These sub-areas have been specified as part of the OSI standards [1].

A *Network Management Framework* encompasses the various components needed for network management. The Network Management Framework (Figure 1) includes a Managing Entity or Manager, a Managed Entity or Agent, Managed Objects, Management Protocol and a Structure of Management Information. Managing entity and managed entity are also known as *Management Entities*. The number of instances of each component is dictated by individual network management models.

On any network node the operational protocols (e.g., IP, TCP, X.25) maintain variables (or real resources) which are used by the management entities. A projection of these variables as seen by the management framework is termed *Managed Objects*. The management framework has access to the managed objects only. Any operation requested on a managed object is translated into a suitable operation on the corresponding real resource. A *Management Information Base* (MIB) is a repository of managed objects. A MIB is defined using a set of structural rules (schema) called *Structure of Management Information* (SMI). Management objects have widely varying capabilities and characteristics depending on the management model. Ideally, anything that needs to be managed should be represented as a managed object. However, in a particular model the structure or definition of management information may restrict the capabilities of managed objects. Such restrictions may result in only a subset of the real resources being represented as managed objects.

A *Managing Entity* provides a set of services to a Management User to control, monitor and configure a network. The managing entity passes management commands received from a user to one or more managed entities and acts upon responses or abnormal event reports received from managed entities. A *Managed Entity* carries out the commands received from the managing entity and performs certain operations on managed objects. A managed entity supports interfaces with the operational protocols and the MIB to access real resources through managed objects. A managed entity is also responsible for forwarding abnormal events to the manager. The bidirectional flow of information between a managing entity and a managed entity is governed by a set of rules known as the *Management Protocol*. Both Manager and Agent support one or more management protocols at least one of which is common between them.

A network management framework also defines the interfaces between components. These interfaces are shown in Figure 1 and are described below:

User-Manager Interface: User management commands, command completion status and event reports are exchanged across this interface. Information flowing across this interface forms the highest level of abstraction of management information.

Manager-Agent Interface: This interface provides the rules governing the manager-agent interactions. The data (PDUs) flowing across this interface is governed by the management protocol used. Information (PDUs) exchanged across this interface includes:

transformed user commands and responses and event reports from the agent.

Agent-MIB Interface: The agent accesses the managed objects across the Agent-MIB interface. Requests for operations on managed objects flow across this interface from agent to MIB and responses to such operations are returned in the reverse direction.

These interface definitions are useful in defining different types and levels of transparency which are discussed in Section 3.3.

A *Management User* is any application that uses the services provided by a management framework to perform network management functions. A *management service* is a facility that is made available to a management user by a management framework provider across the user-management framework interface. Management services include but are not restricted to: management object access and modification, MIB traversal to locate managed objects, event reporting and secure management information transfer.

A *Management Model* is a specific instance of the management framework. The framework specifies the components needed, whereas the model specifies the overall functional aspects of the components and details of their interactions. A management model may be object-oriented or non-object-oriented.

Examples of Management Models are the Internet Management Model and the OSI management model. Figure 2 shows the OSI and Internet Network Management models superimposed on the Network Management Framework. In the case of the Internet Management Model, the managing entity is the Network Management Station (NMS), the managed entity is the agent, the management protocol is Simple Network Management Protocol (SNMP) [2], SMI is defined in [3] and the MIB is MIB-II [4]. The resources modeled as management objects are those maintained by TCP, IP, ICMP etc. The OSI model is an example of an object-oriented model. The managing entity is an OSI Manager, the managed entity is an OSI Agent, the management protocol is Common Management Information Protocol (CMIP) [5], the MIB is specified using Guidelines for the Definition of Management Objects (GDMO) [6].

3 Issues in Interoperability

The problems in interworking dissimilar management models can result from incompatibilities between any two corresponding components in the respective management models. For instance, incompatibilities can be caused by different SMIs, different management protocols or different service specifications. Some of these incompatibilities may be capable of being resolved while there may exist others that cannot be resolved due to various reasons. Based on these incompatibilities, the issues involved in interoperability between two or more dissimilar management models can be broadly classified into the following categories:

- MIB or SMI related issues.
- Protocol or Service related issues.
- Transparency related issues.
- Performance related issues.
- Standards/Specification related issues.

Of the abovementioned issues, performance and standards/specification related issues are implementation dependent and hence not covered in this paper.

3.1 MIB or SMI Related Issues

In this section, issues arising from dissimilarities in the structure and characteristics of management information are discussed. This type of incompatibility may be caused by the existence of different SMI definitions, management object characteristics and access methods. The notable differences in the MIB can be due to the information model (e.g., object-oriented vs non-object-oriented), naming schemes for management objects, managed object instance identification and access, operations allowed on objects and notation used for defining objects. MIB or SMI related incompatibilities are resolved by a technique referred to as *MIB translation*.

MIB translation is the process of mapping a MIB (called source MIB) in one management model to a MIB (called destination MIB) in a different management model. The source MIB and the destination MIB must be equivalent in terms of capabilities but need not be the same

i.e., the destination MIB must offer the same views including access control restrictions as the source MIB. There are two types of MIB translations:

Direct Mapping: This type of translation maps each management object in one model into one or more management objects that represent the original management object, but in a format consistent with a different management model [7, 8]. The effect of direct mapping is to project a management object in one model into an equivalent object (or objects) in another management model. This projection may involve syntactic and object type changes. Direct mapping is usually adopted in (but is not restricted to) application gateway based schemes where a manager uses the projected management object to address the actual object in a different management model.

Abstract Mapping: An abstract mapping scheme attempts to project the salient information content and relationships among objects in a MIB in one model onto another management model. This mapping scheme does not attempt a one-to-one translation of management objects. Abstract mapping may aim at removing redundant or extra information in MIBs thus presenting only condensed information that is needed by management applications belonging to a different management model. Such mappings are adopted when a management application in one model is based on an SMI that is less complex than the SMI supported by the source model and the entire MIB details are not needed or cannot be handled. Conversely, a management model supported by a powerful SMI and services can abstract several management objects in the source model into a smaller set of objects. Such mappings may result in the projected MIB having different level of detail and different structure. Abstract mappings need external inputs to specify the abstraction details which increase the complexity of the mappings.

In a management information model where objects are truly object-oriented, there is a clear boundary that separates the management information user from the internal implementation details of a managed object. The behavior of an object to an external operation is specified as part of the object definition. An object-oriented model may support inheritance and allomorphism which facilitate the construction of complex relationships among managed objects in the model. In a non-object-oriented model there is no clean separation between external behavior and internal implementation. Moreover, in some models there may be no explicit specification of internal or external behavior. Translation between two dissimilar information models is one

of the major issues that needs scrutiny in the area of network management interoperability. Currently, there are no known formal techniques or guidelines that achieve such a translation.

Each management model has its own scheme to name managed objects that facilitates uniform and consistent access to such objects. These naming schemes could be radically different. If these differences are not resolved, a manager in one model will not be able to address managed objects in another model. Dissimilar naming schemes may be overcome by a process called *Name Mapping*. Name mapping allows managed object names in one model to be mapped onto names in another model so that a manager in the mapped model can use the mapped names to access the original managed objects. For example, the OSI and Internet management models have dissimilar naming schemes. Thus, an OSI manager will not be able to access Internet managed objects and cannot manage Internet nodes. Name mapping, in this example, allows managed objects in the Internet model to be mapped onto names that conform to OSI managed object name format. Such a mapping allows an OSI manager to uniformly access both OSI and Internet objects.

An example based on [7] for name mapping is shown in Figure 3. This example shows the *udp* group of MIB-II mapped to conform to the OSI method of addressing the *udp* objects. Name mapping in this case involves registration, class definitions and creation of name bindings (Figure 3(a)). The classes needed to map the *udp* objects are shown in Figure 3(b). Two classes are defined: (1) **udp** class with the *udp* group scalars as attributes and (2) class **udpEntry** with the *udpTable* columns as attributes of the object class. Each of these classes contains a special attribute called *naming attribute* which helps in instance identification. The **udp** class has **udpName** as its naming attribute and **udpEntry** class has **udpEntryName** as its naming attribute. Registration involves assigning new OIDs to existing *udp* objects, new classes and name bindings created for purposes of name mapping. Registration is shown in Figure 3(c). The registration scheme in this case duplicates the *udp* objects at a different node in the OID tree (Figure 3(c)). The OID of *udpInDatagrams* and its mapping are shown in Figure 3(c) [1].

Name bindings establish the parent-child relationship and facilitate the specification of the relative position of a class in the Management Information Tree hierarchy. The name bindings created and the effect of such name bindings on the *udp* objects are shown in Figure

[1] Mapped names are shown with a ' at the end of the name throughout this paper. For instance, the mapped OID of *udpInDatagrams* will be referred to as *udpInDatagrams'*.

3(d).

Some management models allow for dynamic instantiation of objects while others may not. Some allow creation of new objects dynamically with some constraints. In cases where a model allows multiple instances of a managed object type, the model provides for schemes that help uniquely identify such instances. Incompatibility in such instance identification schemes is an issue that needs consideration. *Instance Mapping* is the process by which an instance of a managed object in one management model is mapped onto an instance in another management model that is consistent with the instance identification scheme in that model.

The Internet model identifies scalars by using a suffix zero and uses an index value along with the Object Identifier (OID) of the table entry object to uniquely identify a row in a table, whereas OSI uses Distinguished Names (DN). An instance mapping scheme would map a DN to the corresponding instance identifier (i.e., the OID and the index value) on the Internet side.

To continue our previous example of the *udp* group, in case of OSI access, since the scalars are defined as the attributes of the class **udp**, they can be accessed using attribute related operations. Instance identification for table entries is achieved using a combination of name binding and value of the naming attribute. Table object instances are accessed using an index value (which is the value of the naming attribute) as the RDN. The name binding provides the rules for concatenation of RDNs to form the DN. The naming attribute value is defined to be the value of the column variables that form the index. Thus the naming attribute uniquely identifies a particular entry in the table. For *udpEntry*, the naming attribute **udpEntryName** contains the combined value of *udpLocalAddress* and *udpLocalPort*. Figure 4(a) shows the instances of *udpEntry* class for the sample *udpTable* shown in Figure 4(b).

During translations aiming at removing the incompatibilities across management models, some restrictions may have to be imposed thereby reducing the functional capabilities of either management model. While attempting translations such reductions in capabilities must be minimized. Based on the model, the notation used to define management objects may be different. In order to translate MIBs, one would have to translate the objects specified using one notation into objects in a different notation. This task involves both syntactic and semantic translation (since the notation used to define objects could be associated with different semantics).

3.2 Protocol and Service Related Issues

If management models that need to interoperate support different management protocols, there is a need to (bidirectionally) map the protocols in order to ensure flow of management information between the manager and the agent. Such a mapping is achieved using a technique called *Protocol Translation*. Protocol Translation is a systematic, deterministic, bi-directional mapping of PDUs from one protocol to another. This may result in a one-to-many or many-to-one transformation of PDUs. The translation does not involve functional mapping of management services. For protocol translation to be complete each PDU in one protocol must be mapped onto one or more PDUs in the other protocol along with associated parameters and error codes.

Figure 5 shows the mapping of PDUs between CMIP and SNMP. For detailed parameter mapping of PDUs, the reader is referred to [9].

In many cases such complete translation may not be feasible. An example of such a case is the scoping function specified through a parameter (or field) of a CMIP PDU. In order to map the scoping function it is necessary to map the semantics of such a function. Hence a single CMIP PDU with scoping specified may result in several SNMP PDUs (GetNext PDUs) being generated to retrieve the managed objects specified in the CMIP PDU. Such mappings are commonly referred to as *Service Emulation* or functional mappings.

Service emulation is adopted as a solution when services supported by one management model are different from those of the other. Service emulation is defined as the mapping of services offered by one model into services offered by another model. The primary need for service emulation is to internetwork management models supporting services that are not one-to-one correspondent. While protocol translation tries to perform mapping at a PDU level, service emulation attempts to map a service (one or more PDUs) in one management model to a service or set of services (one or more PDUs) in another management model.

Figure 6 shows a M_GET request with full sub-tree scoping to retrieve the *udp* group of objects maintained by the SNMP agent. This figure also shows the flow of messages with parameters. It can be seen that a single M_GET request is sufficient to retrieve all objects under the *udp* group. This M_GET request (or CMIP PDU) results in the generation of a series of SNMP GetNextRequest PDUs on the SNMP side. The actual sequence of PDUs exchanged is shown on the arrows. When a filter condition such as "udpLocalPort=62" is added to the same scoped request as in Figure 6 the M_GET responses marked 4, 7, 13 and 16 are not forwarded

to the manager. This is due to the fact that the filter condition evaluates to "true" only for the second entry in the *udpTable*. Thus only the M_GET response marked 10 in the figure gets forwarded to the manager. Filters are normally processed on an application gateway as part of service emulation.

Other issues pertinent to protocol and service translations include packet size restrictions, performance changes and bandwidth utilization effects resulting from such translations.

3.3 Transparency Related Issues

This subsection discusses the relevance of transparency in the context of interoperability and introduces concepts pertaining to transparency. The difference between interoperability and transparency can be understood from the following example. When two networks interoperate, they allow users on either network to access and manage the other network. This does not necessarily mean that users on either network are oblivious of the fact that there are different types of underlying networks. For instance, a user on one network may use two different management applications to manage two different networks. A typical example of such a situation is the case where a node runs both SNMP and CMOT protocols. A user can use different applications to manage Internet nodes and OSI nodes. In this case, although either network provides access to the other network, users on the networks must be aware of the location of what is being managed. Thus interoperability does not necessarily ensure transparency.

However, transparency and interoperability are closely related. Transparency results from bridging differences arising due to heterogeneity, which is one of the goals of interoperability. Thus, transparency could be used as a measure of the level of interoperability, with a greater degree of transparency implying greater level of interoperability. Transparency is introduced and discussed here based on the framework presented in Section 2.

Transparency can be defined as the ability of a service user to use the same set of facilities independent of the underlying provider(s). In other words, transparency is the ability of a service provider to hide the underlying details so that the service user sees a common set of services in spite of different service providers.

Some of the terminology presented here has been borrowed from Distributed Systems [10] and adapted to suit a network management environment. In the context of network man-

agement, transparency can be broadly classified as:

- Application or Service Transparency.
- MIB Transparency.
- Protocol Transparency.

Figure 7 shows the various types of transparencies based on the management framework definition.

Application or Service Transparency is defined as the ability of a Network Management application or user to use the same set of services provided by a management model to manage dissimilar networks. This level of transparency is defined between the user and manager and occurs across the user manager interface.

Figure 8 shows application level transparency. The network management application issues commands that are abstracted versions of the services. These commands are independent of the underlying network management system. The network management framework interprets these commands and them into management protocol PDUs (could be SNMP, CMIP).

MIB Transparency is defined between manager and MIB and spans the logical Manager-MIB interface. MIB Transparency is the ability of a managing entity to access and retrieve management information using the same method of access and retrieval across different dissimilar MIBs. MIB Transparency can be further sub-divided into:

Name Transparency: This is the ability of a manager to address managed objects in different models using a uniform name format.

Access Transparency: This denotes the ability of a manager to use the same set of operations to access objects belonging to different management models.

Data Transparency: The ability of a manager to interpret the management information retrieved from different management models using the same set of rules.

Protocol Transparency is defined across the Manager-Agent Interface. This type of transparency is defined as the ability of a managing entity to use a single management protocol to manage connected networks that support different management protocols.

In Figure 9, the managing entity achieves name transparency using an application gateway which allows the manager to access an Internet object using OSI object syntax and semantics. Since the manager uses the same set of operations to access both Internet and OSI objects, the fact that there are objects belonging to two different management models is transparent to the manager. The manager also achieves protocol transparency since it uses the same management protocol to manage two dissimilar networks supporting different management protocols (SNMP, CMIP).

Performance Transparency is defined at two different levels: a) at the user level across the User-Manager interface and b) at the manager level across the Manager-Agent interface. At the user level, Performance Transparency ensures that management functions requested by the user are carried out with the same degree of efficiency. In other words, Performance Transparency allows a user to achieve the same level of efficiency, turnaround time, and flexibility in requesting management services to effect management across heterogeneous network management models. This type of transparency is difficult to quantify and at present there are no schemes to measure this type of transparency.

At the Manager level, Performance Transparency is defined as the same amount of functionality and efficiency in retrieving management information that resides in one network vis-a-vis information that resides in another network supporting a different management model.

Transparency can be used to define levels of interoperability. A network that achieves all of the above mentioned transparencies can be considered completely interoperable. In addition, transparency definitions allow designers and researchers to decide which type of transparency is more important in developing a paradigm based on end-user requirements.

4 Interoperability Schemes

The previous section discussed the issues involved in interworking two or more dissimilar management models. This section presents some approaches for achieving interoperability. Section 4.1 provides an overview of each approach and describes the techniques used by it. Section 4.2 compares the approaches in terms of the issues identified in Section 3.

4.1 Overview

Multiple Stack Approach: Figure 10 shows a network management system implementing a multiple protocol stack. This approach is viable when existing large dissimilar networks need to be managed. In this approach, the management system features a common user front-end which hides, to the extent possible, the details of the underlying networks, management protocols and services [11]. A common set of services are offered to the user. A user makes use of these services to manage the underlying network(s).

A user interacts with the management system using the common set of services provided by the front-end. These service requests are passed on to an interface module. The interface module translates the user request into management requests of that management model for which the user command is intended. Once the user command is accepted by the protocol stack, the situation maps to that of single model network management.

Management objects for different management models are represented in a single global name space using a common format. There is a mapping function for each model supported, that maps managed objects in that model to the common global name space. The interface module, in addition to object name mapping, performs command translation to services of the protocol accepted by the specific stack.

The interface between the user and the protocol stack can be centralized or distributed. If it is at the Network Management Center (NMC), there is no need to duplicate the interface at each node. If the interface is distributed, each node being managed will have to support such an interface.

The multiple stack scheme increases the complexity and size of the managers in a network. This disadvantage is offset by the fact that there are very few managers for any given network.

Multiple Agent Approach: A Multiple agent scheme [12] is depicted in Figure 11. The figure shows only two agents but it can be extended to more than two agents. In this approach, each node has an instance of each type of agent that may be needed. For example, if there are n managers (each belonging to a different management model) that manage a particular node, that node will have to support an instance of an agent corresponding to each manager. The MIB corresponding to the real managed resources at that node is mapped into each of the supported models. There is a version of the MIB that is compat-

ible with each instance of the agent. The agent, on receiving the request, processes the request and forwards the response to the manager that issued the request. This approach necessitates a multiple protocol stack at the agent. Multiple stacks at the agent nodes keeps the manager simple and of reasonable size. However, the size and complexity of the agent is increased. Since the number of agent nodes in a network is large, addition of a new management protocol stack could be a serious performance and software management issue. The advantage with this scheme is that a manager can manage any agent node with its native management protocol and services and management applications are transparent to the addition of a new management model.

Proxy and Application Gateway Approach: Before discussing this approach, it is necessary to define the terms Proxy and Application Gateway.

A *Proxy* is a special agent that acts as a front-end for one or more nodes that need to be managed. It provides a common mode of access and services to manage nodes that may belong to either the same or a different management model. A proxy may thus act as a relay between nodes supporting different transport protocols or provide connectivity between different management models performing protocol translations.

An *Application Gateway* is a specialization of a proxy. Application gateways provide interworking between two dissimilar management models. An application gateway, in addition to protocol translation, provides service emulation or mapping.

In Network Management literature, the terms "Proxy" and "Application Gateway" are interchangeably used. In this paper these terms will be used as defined here.

Based on the scheme used to achieve interoperability, an application gateway may be *stateful* or *stateless*. A stateless gateway does not cache management information locally. A stateless gateway accepts a management service request, forwards the request to the destination management model after suitable transformation and/or mapping. On receipt of a response to a prior management request, a stateless gateway forwards the response to the manager that issued the service request.

A stateful gateway on the other hand caches management information and therefore maintains a snapshot of a remote MIB locally. Such caching primarily leads to improved efficiency and reduced bandwidth usage. The tradeoff is in terms of complexity in implementation.

The disadvantage with a Proxy or Application Gateway based scheme is that fairly complex MIB translation and name mapping algorithms have to be built. Thus, addition of new models may entail enormous effort. The major advantage is that the number of gateways or proxies in a network is comparable to that of managers in the network. This results in the complexity being concentrated at specific points in the network. Addition of a new model does not adversely affect existing proxies. A new model leads to addition of a new application gateway at the point where a network supporting the new model is connected to the existing network.

A scheme using an application gateway approach [13, 9] is shown in Figure 12. In this scheme application gateways are points of interworking between the two dissimilar management models. In this approach, the manager supports a management protocol that is different from that supported by a managed node. An application gateway receives management requests from a manager. These requests are communicated to the application gateway using the manager's native management protocol. On receiving the request, the application gateway maps the received request onto an equivalent request in the managed node's management model. The mapped request may be a single PDU or multiple PDUs. This mapped request is then communicated to the managed node using the management protocol supported by the managed node. On receiving a response from a managed node, an application gateway maps the response onto an equivalent response in the manager's management model. In order to accomplish protocol translation and service emulation, the application gateway has to maintain information on at least some of the following:

- MIB structure
- Managed objects
- Traps or Notification
- Security

For this approach to be functional, algorithms for mapping objects across models must be specified. Protocol translation rules and functional or service emulation schemes must be defined.

4.2 Comparative Study of Approaches

In this subsection, the approaches discussed thus far are compared and contrasted in light of the issues presented in Section 3. The merits and demerits of each approach are included to keep the discussion complete.

MIB related issues: MIB related issues are primarily name and instance mapping. Other issues pertain to operations allowed on objects.

In the multiple stack approach, the user is provided a common interface for managing different networks that are part of a single management environment. Name mapping needs to be performed between a single global space (that contains all objects that need to be managed) to objects in specific management models and vice versa. Objects are represented using a common format. These objects are mapped to model-specific names in the stack machines. In the reverse direction there are mapping functions that map names in different models to the common name space. The advantage of such a scheme is that if a new management model with a different SMI is added, the only changes are extensions to name mapping functions and interface to the new stack. Once the names are mapped to the model-specific names, the management scenario maps to that of a single model management system. Since the objects are maintained using a common protocol independent format, the management applications are not affected by the addition of a new stack. There is no incompatibility in operations permitted on objects since the operations requested are those allowed by that management model.

The multiple agent approach obviates the need for name and instance mapping. Each node supports an agent and MIB instance corresponding to a specific management model. Since there are agents and MIBs corresponding to each management model supported, management requests can be issued by any of the managers and the request is handled by a corresponding instance of the agent at the managed node. MIBs on the managed node correspond to a specific model and hence name and instance mapping are unnecessary. This approach, however, necessitates mapped MIBs. Multiple copies of MIBs (for the same set of real resources) are derived using static MIB mapping algorithms. These static algorithms address and solve the problem of name mapping. Once the network is functional, each agent refers to its own copy of the MIB. Thus, runtime name and instance mapping are avoided.

With the application gateway approach, managed objects in the agent model need to be mapped to objects in the manager's model. The application gateway approach requires both name and instance mapping. An example of direct name and instance mapping between Internet MIB-II and OSI GDMO MIB can be found in [7]. MIB translation includes the following:

- specification of registration schemes for managed objects in the MIB to be translated.
- translation algorithm to map objects in the source MIB to objects in the destination MIB.
- specification of instance identification schemes.

Protocol and Service Related Issues: Protocol and service related differences arise if the manager and agent support different protocols and services.

In the multiple stack approach, a user is provided a common set of services. Management commands issued by the user are translated to equivalent commands or service requests corresponding to the agent model. Thus there is no need for protocol translation since the management request is handed out to the corresponding protocol stack. The final communication between the manager and an agent is using the native protocol supported by the agent. Service emulation is indirectly performed when the common set of services made available to the user is translated to management service requests in a particular model and the responses received are again translated back to the common set of service requests.

In case of the multiple agent approach, it is essentially a manager communicating with an agent belonging to its own model. Hence neither protocol translation nor service emulation is necessary. However, with this approach, a user gets a different set of services to manage the network depending upon the manager chosen.

In the application gateway approach, the manager and agent are in different management models. They support different management protocols and different set of services. Hence both protocol translation and service emulation may be necessary. For the application gateway approach to work, guidelines for both protocol translation and service emulation are normally specified in conjunction with a MIB translation scheme. The reader is referred to [9] for an example scheme that attempts to interwork OSI and Internet. Protocol translation is specified on a PDU basis with each parameter and error being mapped

bidirectionally between the manager and agent models. Service emulation is specified on a service basis with the details of the mapping of the service. In this approach, a user gets the services that are available at the manager chosen to manage the network and could be different depending on the model chosen. For instance, if an OSI manager is chosen then the management applications can make use of the Common Management Information Services (CMIS) [14] to manage the network. If an Internet manager is chosen then the Internet services are available.

Transparency Related Issues: The different types of transparencies have been outlined in Section 4. The following paragraphs discuss the transparencies achieved by the multiple stack, multiple agent and application gateway approaches.

The multiple stack approach provides a common front-end to the user and a standard user-management framework interface. Due to this standard interface the underlying protocol stack differences are transparent to the user and management applications. Hence application transparency is complete. The multiple agent approach provides a front-end that is dependent on the manager's model and is common irrespective of the types of models supported by the agent nodes. Thus, management applications on any manager can be used to manage nodes that are part of the network since each agent will support an instance of an agent that belongs to the manager's model. The existence of different models is transparent to the management applications.

In case of the multiple stack approach, the manager has to do a conversion from the common object format to a model-specific format. Thus a manager has to know the differences in the naming and instance identification schemes. The manager is thus not transparent to the different types of MIBs supported in the network. In the multiple agent scenario, there is only one type of MIB that the manager deals with and hence the question of MIB transparency does not arise. With the application gateway approach, translation algorithms map MIBs in different models to corresponding MIBs in the manager's model. Hence a manager is able to view and access MIBs belonging to different models as though they belonged to the native management model. The existence of dissimilar MIBs is thus transparent to the manager.

Protocol translation is not needed in case of multiple stack approach. The user requests are translated to appropriate protocol stack machine requests at the manager and the manager then communicates with the agent using the appropriate management protocol supported

by the agent. Existence of multiple protocols in the network is transparent to the manager. The multiple agent approach also achieves protocol transparency. Since a management request is routed to the appropriate agent, there is only one management protocol involved in the transaction. Hence this approach also achieves protocol transparency. With the application gateway approach there is more than one management protocol. However protocol translation at the application gateway ensures that the manager is not aware of the fact that more than one protocol exists on the network. Thus application gateway approach also achieves protocol transparency.

It is important to realize that although each approach has been specified as an interoperability scheme all the approaches have primarily resolved the issues identified in this paper. The primary focus of this section is to emphasize the fact that an issue based approach will lead to better interoperability schemes.

5 Conclusions

Network management is rife with terminology. Even concepts are sometimes intermixed with specific model related terminology. Considering that network management is a fast paced research field, there is an urgent need to separate out the general concepts from the specifics. This paper serves to fulfill this exact need. A conscious attempt has been made to introduce the concepts and associated terminology without details of implementation. Interoperability issues are identified based on a generic framework. Different interoperability schemes are described and the techniques adopted by each scheme in resolving issues identified in this paper are discussed. It is the authors' opinion that paradigm development based on a careful consideration of the relevant issues will lead to clean and efficient schemes.

Currently, work is in progress towards achieving interoperability and coexistence of OSI and Internet management models [7, 9, 15]. Although some progress has been made, there is work to be done in automatic MIB translations, abstract MIB mappings and stateful application gateways. Additional areas that may need consideration are the formal specification of existing standards, development of formal notation for MIB specification and automating the generation of interworking components of management frameworks. We expect to see a lot of activity and interest in the field of interoperability in the next few years.

References

[1] International Organization for Standardization. *ISO IS10040, Information processing systems - Open Systems Interconnection - Systems management overview.*, July 1991.

[2] J. D. Case, M. S. Fedor, M. L. Schoffstall, and C. Davin. *Simple Network Management Protocol. (RFC 1157).* DDN Network Information Center, SRI International, May 1990.

[3] M. T. Rose and K. McCloghrie. *Structure and Identification of Management Information for TCP/IP based internets. (RFC 1155).* DDN Network Information Center, SRI International, May 1990.

[4] M. T. Rose and K. McCloghrie. *Management Information Base for Network Management of TCP/IP based internets: MIB-II. (RFC 1213).* DDN Network Information Center, SRI International, March 1991.

[5] International Organization for Standardization. *ISO IS9596, Management Information Protocol Specification - Part 2: Common Management Information Protocol*, January 1990.

[6] International Organization for Standardization. *ISO IS10165-4, Information Technology - Open Systems Interconnection - Management Information Services - Structure of Management Information Part 4: Guidelines for the Definition of Managed Objects.*, July 1991.

[7] L. LaBarre. *IIMC Draft 3: Translation of Internet MIBs to ISO/CCITT GDMO MIBs. (Internet Draft).* DDN Network Information Center, SRI International, May 1993.

[8] L. LaBarre. *OSI Internet Management: Management Information Base. (RFC 1214).* DDN Network Information Center, SRI International, April 1991.

[9] A. Chang and D. Liu. *IIMC Draft 3: ISO/CCITT to Internet Management Proxy. (Internet Draft).* DDN Network Information Center, SRI International, May 1993.

[10] A. Goscinski. *Distributed Operating Systems - The Logical Design.* Addison-Wesley, Singapore, 1991.

[11] U.S. Warrier and C. A. Sunshine. A platform for heterogeneous interconnection network management. *IEEE Journal on Selected Areas in Communications*, 8(1):119–126, January 1990.

[12] G. Pavlou, S.N. Bhatti, and G. Knight. Automating the OSI to Internet management conversion through the use of an object-oriented platform. In *IFIP 6.4 Symposium on Advanced Information Processing Techniques for LAN and MAN Management*, Paris, April 1993.

[13] P. Kalyanasundaram and A. Sethi. An application gateway design for OSI internet management. In *IFIP Transactions on Integrated Network Management, III*, pages 389–401, San Francisco, April 1993. North Holland, Amsterdam.

[14] International Organization for Standardization. *ISO IS9595, Information Technology - Open Systems Interconnection - Common Management Information Service Element*, January 1990.

[15] L. LaBarre. *IIMC Draft 3: Translation of Internet MIB-II to ISO/CCITT GDMO MIB.* DDN Network Information Center, SRI International. (Internet Draft), May 1993.

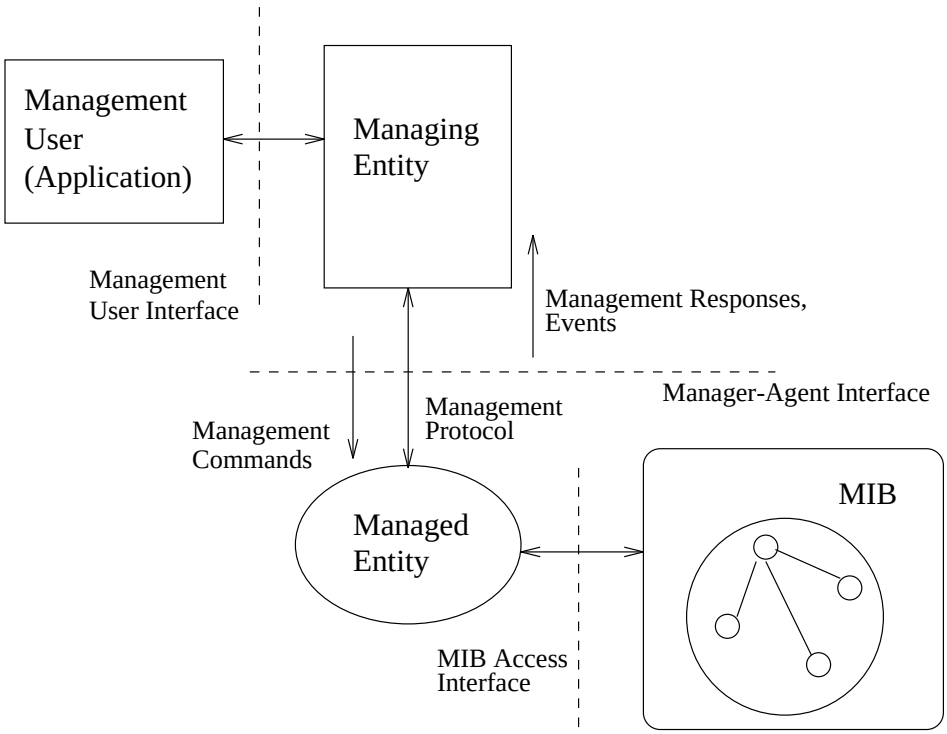


Figure 1: Network Management Framework

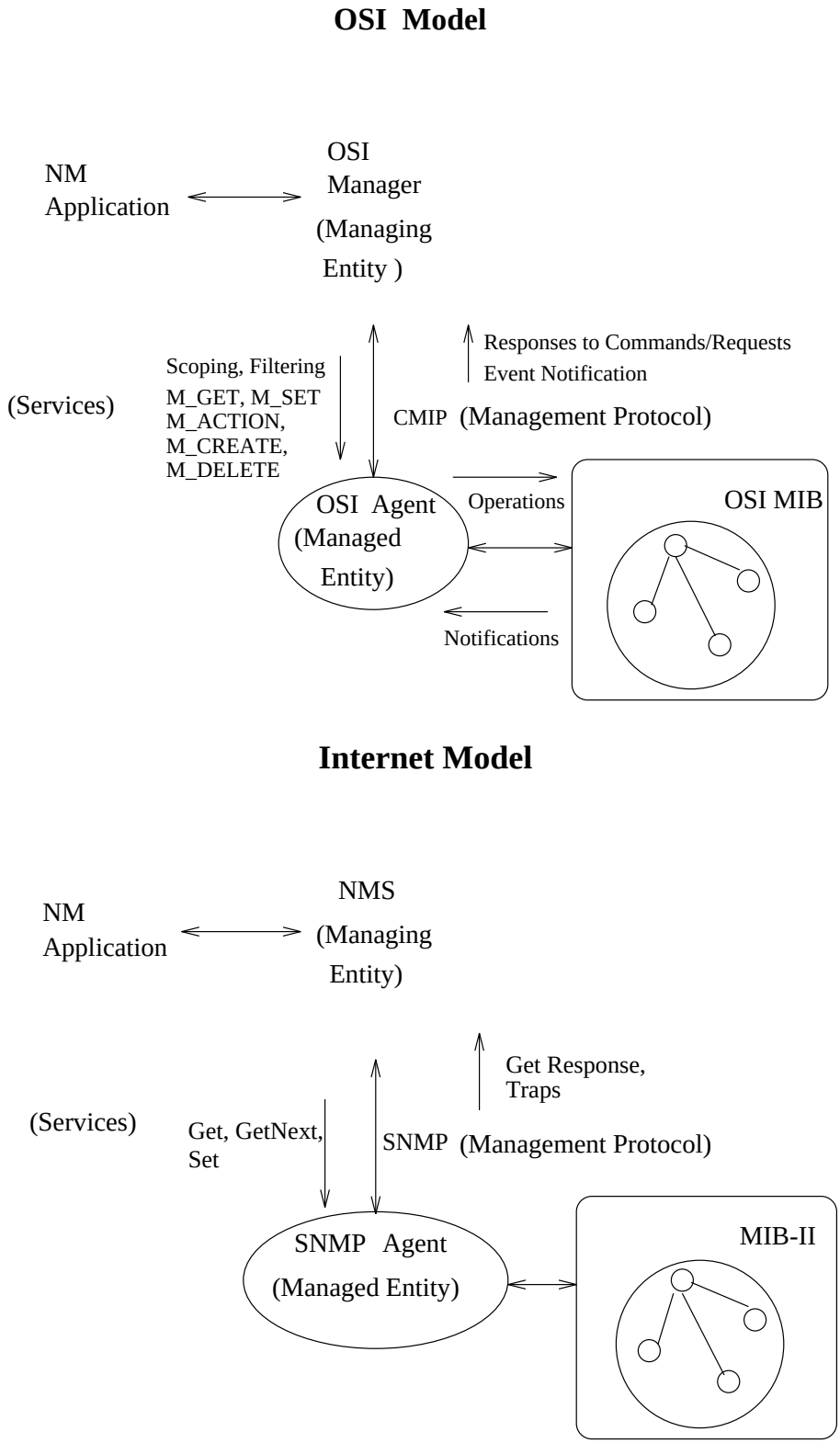


Figure 2: Superimposition of Internet and OSI entities on the management framework.

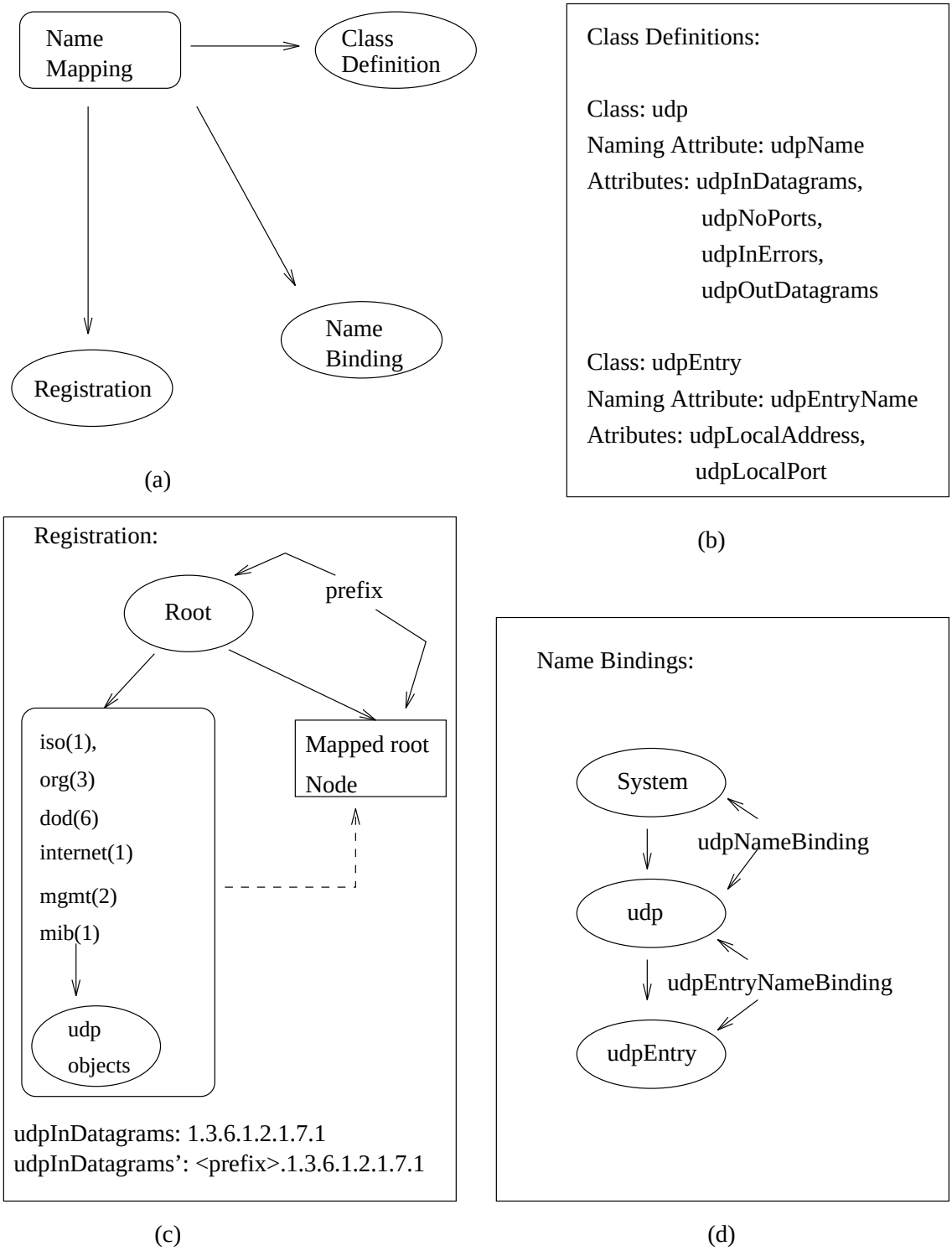


Figure 3: Name Mapping from Internet to OSI Management Objects

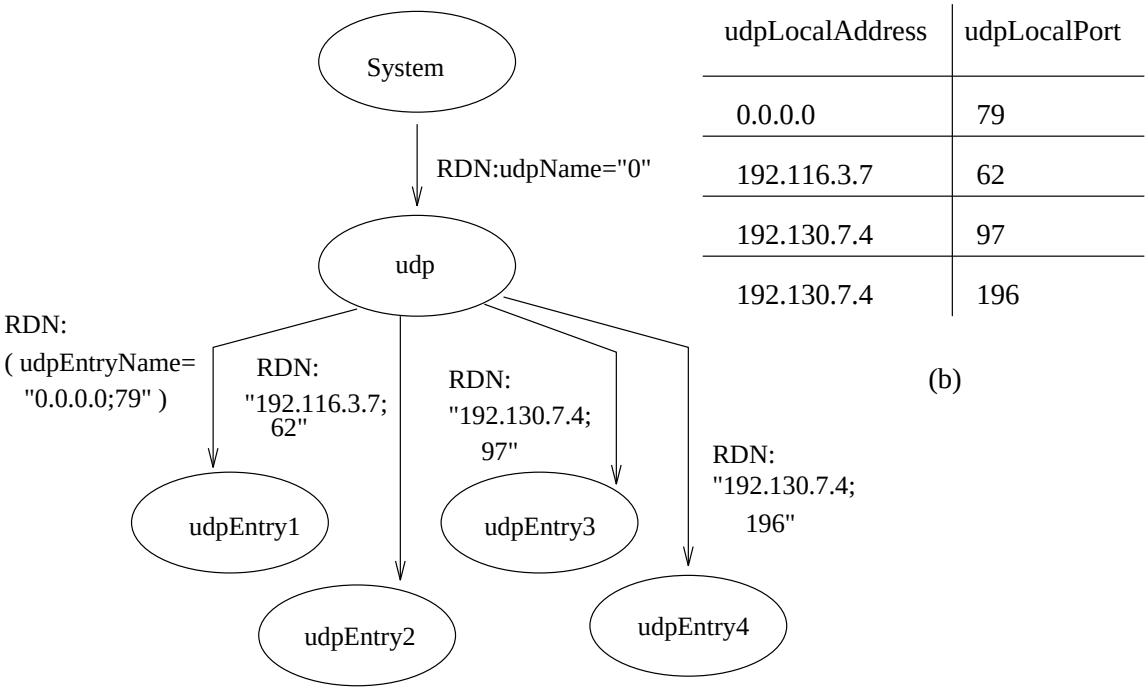


Figure 4: Instance Mapping from Internet to OSI Management Objects

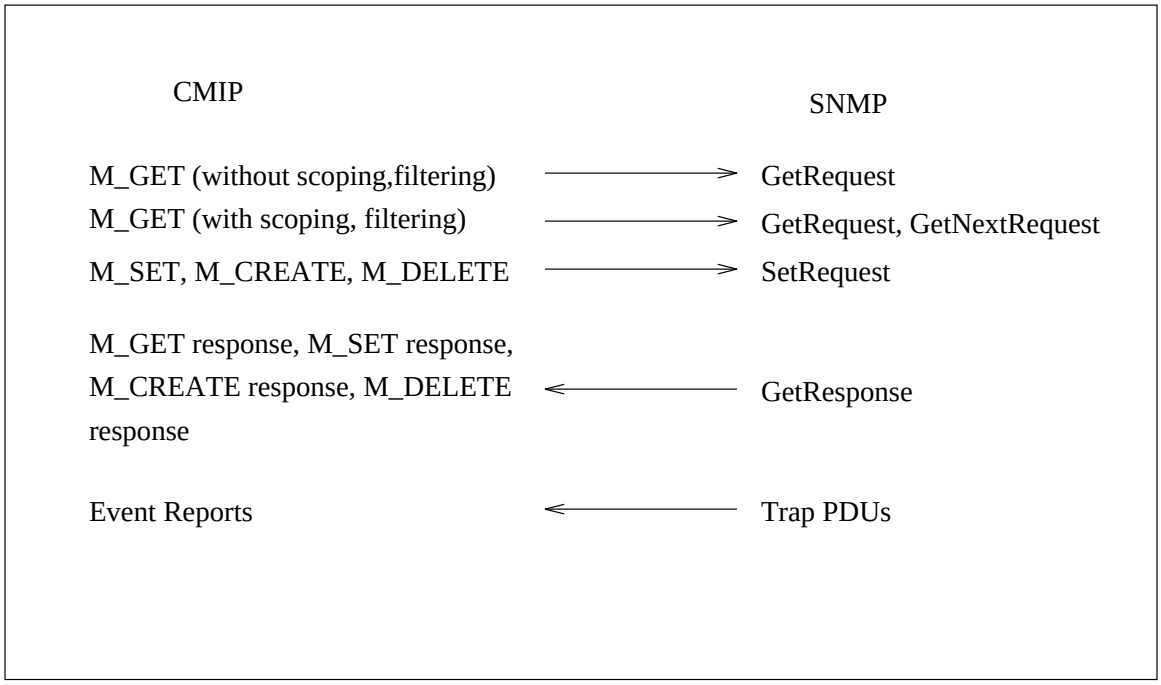
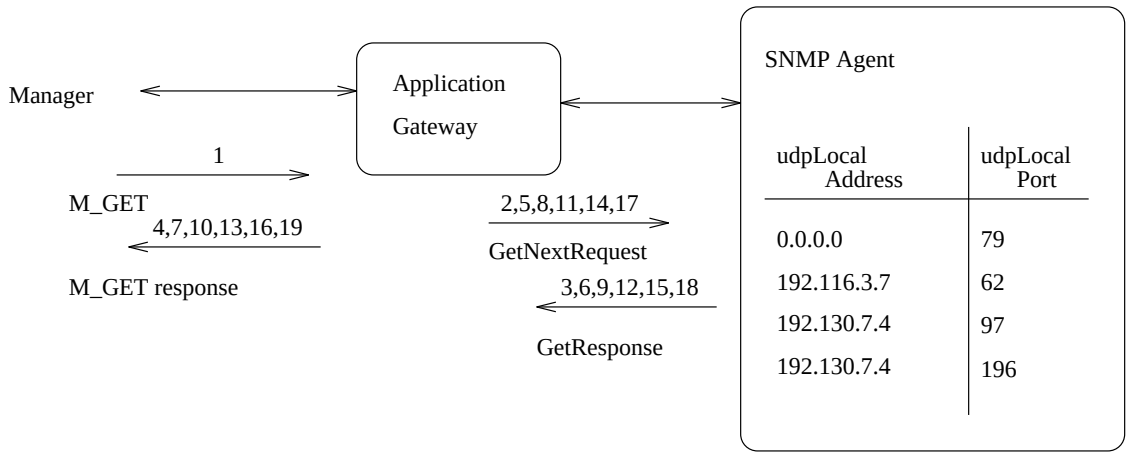


Figure 5: Protocol Translation



1. M_GET(InvokeId=44, MOC=udp, MOI={udpName="0"}, Scope=FullSubtree, Filter=NULL, Access=None, Synch=BestEffort, Attrs=NULL)

2. GetRequest(RequestId=71, ErrorStatus=0, ErrorIndex=0, VarBind={(udpInDatagrams.0,0), (udpNoPorts.0,0), (udpInErrors.0,0), (udpOutDatagrams.0,0)})

3. GetResponse(RequestId=71, ErrorStatus=0, ErrorIndex=0, VarBind={(udpInDatagrams.0,1249), (udpNoPorts.0,8), (udpInErrors.0,23), (udpOutDatagrams.0,2593)})

4. M_GET response(InvokeId=44, LinkId=44, MOC=udp, MOI={udpName="0"},..., Attrs={udpInDatagrams=1249, udpNoPorts=8, udpInErrors=23, udpOutDatagrams=2593}

5. GetNextRequest(RequestId=72, ErrorStatus=0, ErrorIndex=0, VarBind={(udpLocalAddress, 0), (udpLocalPort, 0)})

6. GetResponse(RequestId=72, ErrorStatus=0, ErrorIndex=0, VarBind={(udpLocalAddress.0.0.0.0.79, 0.0.0.0), (udpLocalPort.0.0.0.0.79, 79)})

7. M_GET response(InvokeId=44, LinkId=44, MOC=udpEntry, MOI={udpName="0"@udpEntryName="0.0.0.0;79"}, ..., Attrs={(udpLocalAddress=0.0.0.0), (udpLocalPort=79)})

8. GetNextRequest(RequestId=73, ErrorStatus=0, ErrorIndex=0, VarBind={(udpLocalAddress.0.0.0.0.79, 0), (udpLocalPort.0.0.0.0.79, 0)})

9.GetResponse(RequestId=73, ErrorStatus=0, ErrorIndex=0, VarBind={(udpLocalAddress.192.116.3.7.62, 192.116.3.7), (udpLocalPort.192.116.3.7.62, 62)})

10. M_GET response(InvokeId=44, LinkId=44, MOC=udpEntry, MOI={udpName="0"@udpEntryName="192.116.3.7;62"}, ..., Attrs={(udpLocalAddress=192.116.3.7), (udpLocalPort=62)})

.

.

.

17. GetNextRequest(. . ., VarBind{(udpLocalAddress.192.130.7.4.196, 0), (udpLocalPort.192.130.7.4.196, 0)})

18. GetResponse(..., VarBind{(udpLocalPort.0.0.0.0.79, 79), (egpInMsgs.0, 24)})

19. M_GET response(InvokeId=44, response indicating completion)

Figure 6: Service Emulation

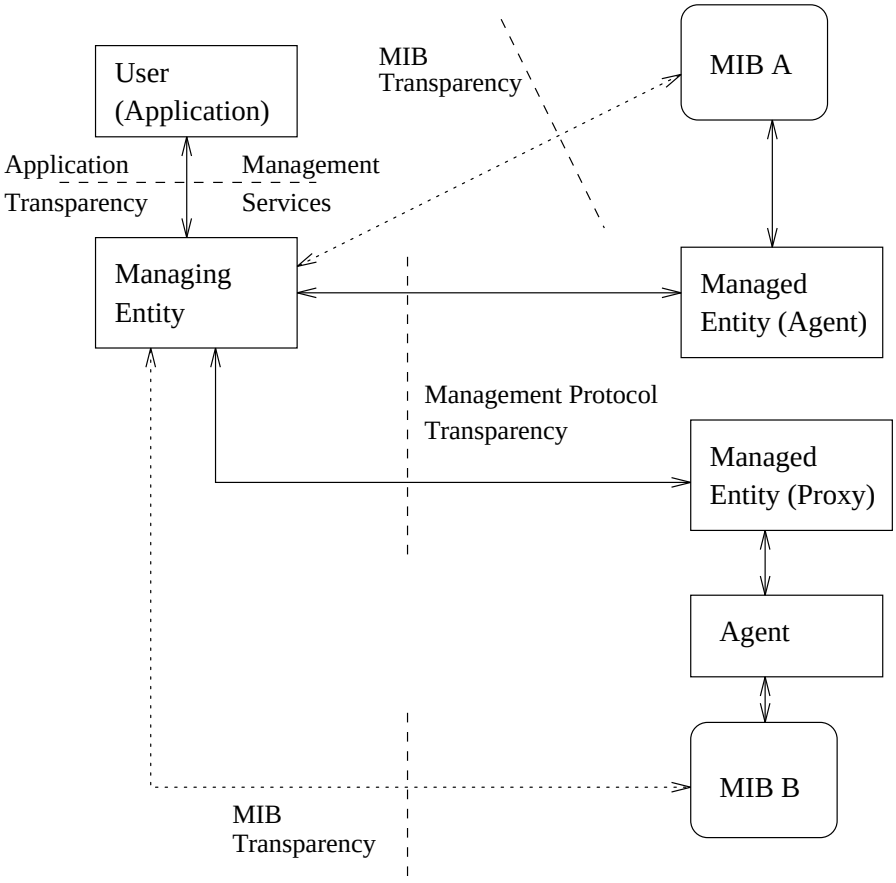


Figure 7: Transparency definition.

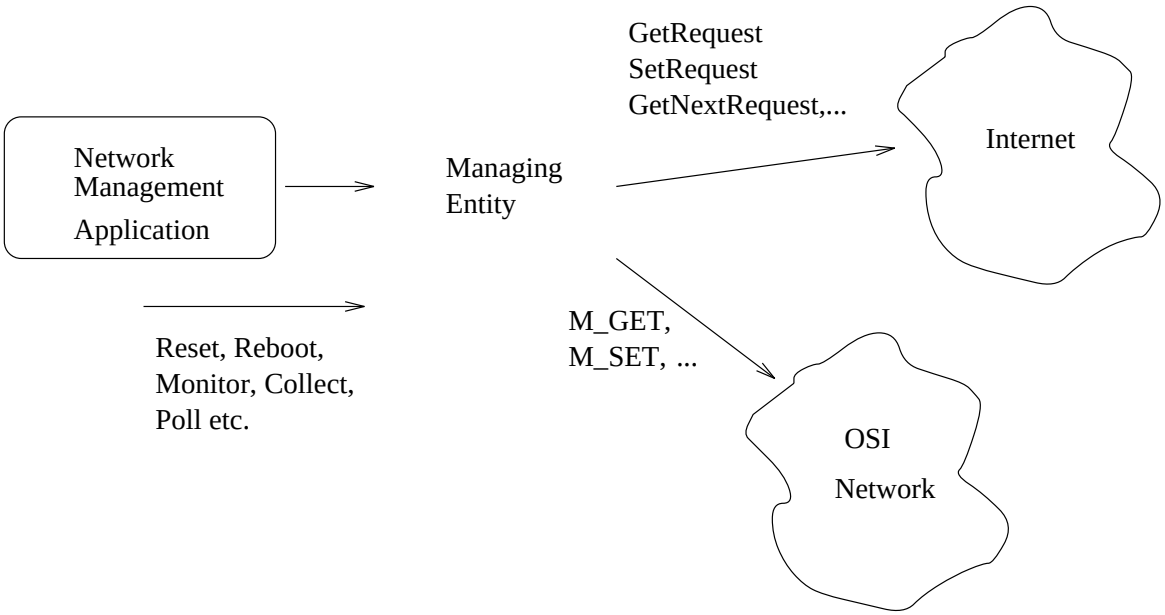


Figure 8: Application Transparency

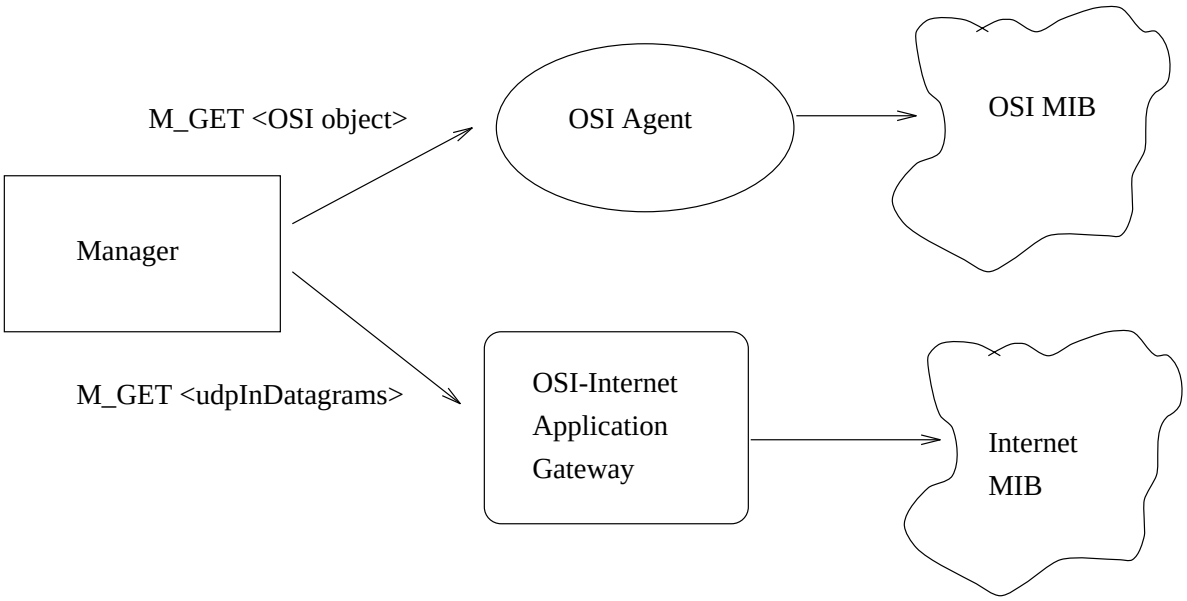


Figure 9: MIB Transparency

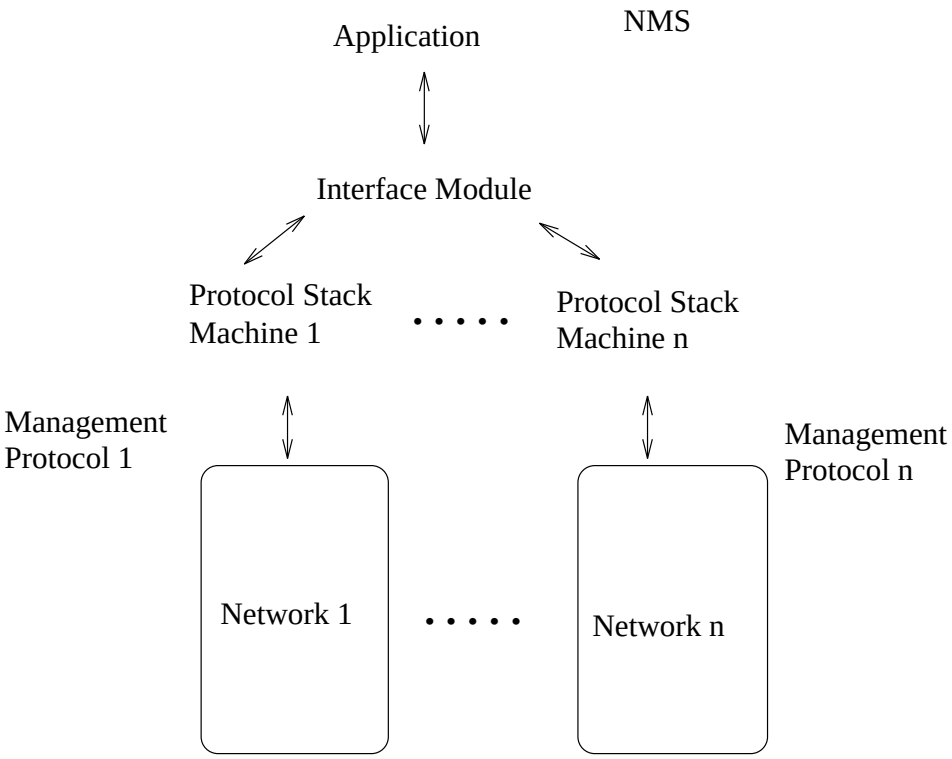


Figure 10: Multiple protocol stack approach.

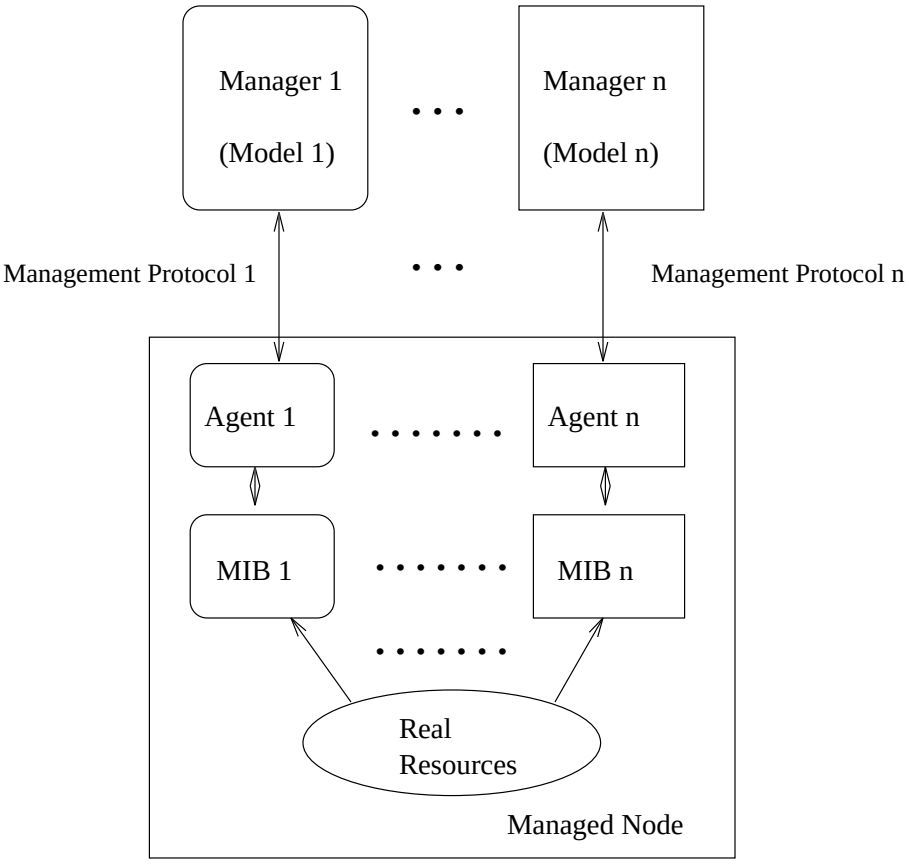


Figure 11: Multiple agent approach.

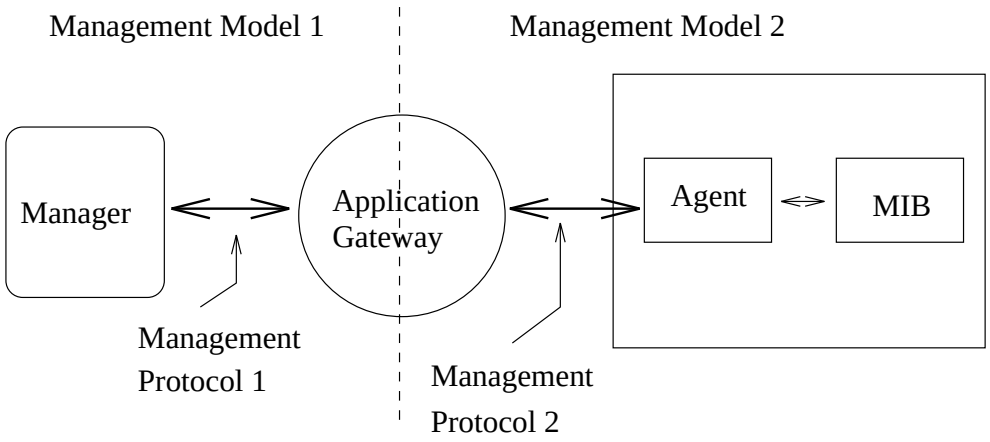


Figure 12: Application Gateway approach.