Experiences on Integration of Network Management and a Distributed Computing Platform

Sakari Rahkila and Susanne Stenberg
Nokia Research Center
sakari.rahkila@research.nokia.com
susanne.stenberg@research.nokia.com

Abstract

The integration of the two recognized network management protocol standards, Common Management Information Protocol (CMIP), and Simple Network Management Protocol (SNMP), and Common Object Request Broker Architecture (CORBA) technology, allows management applications to take advantage of distributed object computing as well as the standardized network management protocols.
This paper describes the Distributed Computing Platform (DCP) prototype developed in Nokia Research Center. The DCP prototype is a framework, including tools, compilers and gateways, built to support both Internet and OSI management through a CORBA infrastructure.

1. Introduction

Telecommunications networks are continuously growing in scale and complexity, and the number of equipment and services they provide is increasing. Usually, with each new technology deployed, a new Network Management System is installed as well. Despite that the functionality in each new Network Management System is nearly the same as in the previous systems, the lack of a common approach in the implementation of these systems prevent them from being integrated. Yet, integrated, distributed management of heterogeneous networks is increasingly important, since today a carrier must be able to set up new services in weeks rather than months, and to deliver integrated services. There is thus a need for a consistent approach to integrate management solutions. Additionally it would be desirable to use off-the-shelf (buy vs. build) components and to leverage existing investments by integration of 'legacy' management applications. Distribution in management applications is needed for scalability and for the cost/performance benefits.

The most promising approach to solve the distribution, interface and integration problems is the Common Object Request Broker Architecture (CORBA) [1] technology by the Object Management Group (OMG). However, CORBA does not provide a network management architecture; it provides a distributed object computing architecture.

The use of standardized network management protocols is important because of cost and time savings and network management system reuse. During the past years the telecommunications industry has gained knowledge and experience establishing telecom network management on the Telecommunications Management Network (TMN) [2] paradigm. In the Internet community, on the other hand, SNMP [3] based network management has gained widespread acceptance due to its simplicity of implementation. Thus, TMN and Internet management will co-exist far in the future. But both of these approaches have problems when it comes to scalability and distribution.

Below we present the TMN, SNMP and CORBA concepts, and describe the effort in the Nokia Distributed Computing Platform (DCP) prototype project to integrate the three technologies.

2. Basic Concepts

2.1. Telecommunications Management Network

The goal of TMN is to enhance interoperability of management software and to provide an architecture for management systems [4]. TMN is defined in the ITU-T M.3000 series of recommendations. The base document is M.3010, Principles for a Telecommunications Management Network [2].

A TMN is a logically distinct network from the telecommunications network that it manages. It interfaces with the telecommunications network at several different points to send/receive information to/from it and to control its operations.

Proceedings of The Thirtieth Annual Hawwaii International Conference
on System Sciences ISBN 0-8186-7862-3/97 $17.00 © 1997 IEEE

1060-3425/97 $10.00 (c) 1997 IEEE

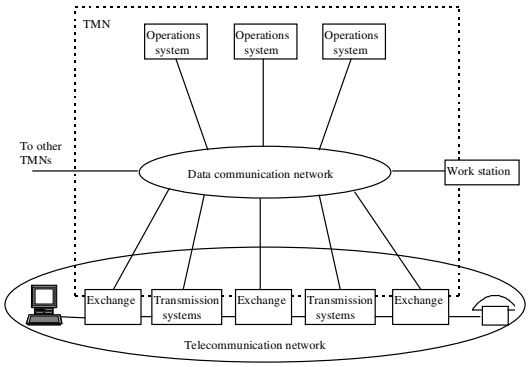

Figure 1. Telecommunications Management Network.

A TMN may use parts of the telecommunications network to provide its communications. Figure 1. presents the relationship of a TMN to the telecommunications network.

The TMN information architecture is based on an object-oriented approach and the agent/manager concepts that underlie the Open Systems Interconnection (OSI) systems management [5].

For a specific management association, the management processes involved will take on one of two possible roles:

A Manager is defined as part of a distributed network management application process, which is responsible for network management processes.

An Agent performs the management operations on the managed objects at the request of the manager, and reports events that have occurred in association with the managed objects.

A Managed Object is the OSI abstract view of a logical or physical system resource to be managed. The managed objects are defined according to the International Standardization Organization (ISO) Guidelines for the Definition of Managed Objects (GDMO) [6]. GDMO attribute values types are described using the Abstract Syntax Notation One (ASN.1) [7]. ASN.1 describes an abstract syntax for data types and values.

The set of managed object classes and instances under the control of an agent is known as its Management Information Base (MIB), an abstraction of network resources, properties and states for purpose of management.

Figure 2. illustrates the interaction between a manager, an agent, and managed objects.

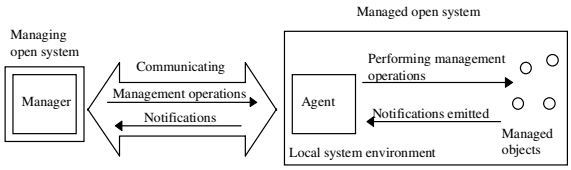

Figure 2. Interaction between manager, agent, and managed objects.

All management exchanges between a manager and an agent are expressed in terms of a consistent set of management operations and notifications. These operations are realized through the use of the Common Management Information Services (CMIS) [8] and the Common Management Information Protocol (CMIP) [9]. CMIP is a connection-oriented protocol, i.e. every message is sent over an association established for that purpose, and closed when the communication is completed.

2.2. Internet Management

Simple Network Management Protocol (SNMP) is a protocol suite developed for the management of the Internet. SNMP was designed to be an application level protocol that is a part of TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite. SNMP is a connectionless protocol, i.e. every message is sent independently.

The SNMP framework is based on the principle of minimally simple agents and complex managers.

Internet uses the term network element to describe any object that is managed. The network element consists of the managed entity and the managed entity's agent [10]. The SNMP structure of information uses simple two-dimensional tables as it's basic containment structure for managed objects. The Internet standards also use ASN.1 constructs in the MIB to describe the syntax of the object types.

2.3. CORBA

The goal of CORBA is to provide a single architecture, using object technology, for distributed application integration.

2.3.1. Object Management Architecture

The Object Management Architecture (OMA) Guide [11] describes an outline of a distributed application architecture, an abstract object model, an overview of the integration model, a reference model of the architecture, and a glossary of terms.

Proceedings of The Thirtieth Annual Hawwaii International Conference on System Sciences ISBN 0-8186-7862-3/97 $17.00 © 1997 IEEE

The Reference Model of the OMA (Figure 3.) includes the Object Request Broker (ORB), Application Interfaces, Common Facilities (CORBAfacilities), Object Services (CORBAservices), and Domain Interfaces.

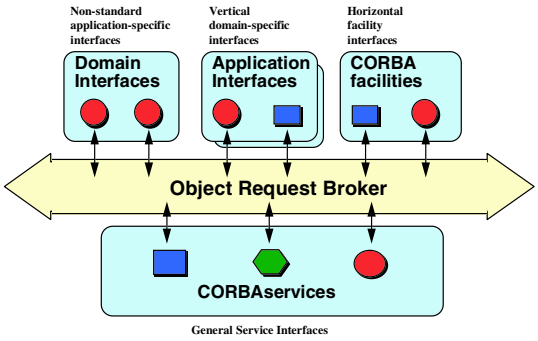

Figure 3. Proposed Object Management Architecture

In the OMA architecture an object can be a tool, an application, a document, a component inside a document, a daemon or a service. Practically any set of services that can be wrapped inside an interface, and invoked through an object reference, is an object.

2.3.2. Application Interfaces

Application Interfaces are specific to end-user applications, they are not defined in the OMA architecture. Applications only need to provide an OMA compliant interface, i.e. an interface defined with CORBA Interface Definition Language (IDL), they need not be constructed using an object oriented paradigm in order to be objects within the architecture.

2.3.3. Domain Interfaces

Domain Interfaces are vertical application domain-specific interfaces, e.g. domain interfaces for telecommunication applications or financial services.

2.3.4. Common Facilities

Common Facilities [12] are horizontal end-user-oriented facilities applicable to most application domains (e.g. Distributed Document Component Facility).

2.3.5. Object Services

Object Services [13] is a collection of services with object interfaces, which provides basic functions for all objects to be shared by all applications. Object Services are used to support construction of applications in a distributed environment. The Object Services are intended to be modular so that clients in the distributed computing environment are free to use as many or as few as necessary to accomplish a task. Examples include Life Cycle Service, Event Service, Naming Service, and Transactions Service.

2.3.6. Object Request Broker

The Object Request Broker provides the basic mechanism for transparently making requests to - and receiving responses from - objects located locally or remotely. The ORB supplies delivery services, activation and deactivation of remote objects, method invocation, parameter encoding, synchronization, and exception handling.

2.3.7. Internet Inter-ORB Protocol

For interoperability between ORBs, the CORBA 2.0 specification defines a mandatory message format, called General Inter-ORB Protocol (GIOP), which can be hosted on any network transport in theory, however, TCP/IP is mandatory. Hosted on TCP/IP the protocol is called Internet Inter-ORB Protocol (IIOP). CORBA 2.0 products thus have to offer IIOP either natively, or as a so called half-bridge, i.e. a gateway to IIOP from the protocol used natively in the ORB.

2.3.8. Interface Definition Language

The CORBA IDL is used to define the interface to a CORBA object. IDL uses the same lexical rules as C++, extended with some new keywords. IDL is programming language independent. It is mapped to the implementation language, i.e. the programming language in which the object implementation code is to be written. IDL compiler output binds clients and object implementations to the ORB. Headers are produced for client and server implementation for the C and C++ mappings. For Smalltalk an implementation template is produced. The implementations are added to an Implementation Repository.

2.4. Integration of Technologies

Standards bodies have recognized the need for the technologies described above to co-exist, and provide specifications for the integration of SNMP, CMIP and CORBA.

2.4.1. Joint Inter-Domain Management

The Joint Inter-Domain Management (JIDM) working group sponsored by X/Open and Network Management Forum (NMF) is seeking to enable inter-operability between CMIP, SNMP, and CORBA. JIDM concentrates on CMIP/CORBA and SNMP/CORBA inter-working. The work is divided into two parts. The first part is referred to as Specification Translation and is expressed as a mechanism for translating between GDMO, SNMP

MIB Definition language, and IDL. The second part is called Interaction Translation and covers the mechanisms to dynamically convert between the protocols in one domain and the protocols within the other [15].

2.4.2. ISO/CCITT and Internet Management Coexistence

In NMF, the ISO/CCITT and Internet Management Coexistence (IIMC) working group has produced specifications that enable interworking between SNMP and CMIP environments. The SNMP/CMIP interworking package consists of five specifications. Three of the specifications address the relationships among object definitions in the two models, e.g., the translation of the Internet MIBs to GDMO [16], [17], and vice versa [18]. In addition, the package includes a specification describing a CMIP to SNMP proxy agent [19].

2.4.3. OMG Telecom Domain Task Force

As stated in the mission statement of the Telecommunications Domain Task Force, it provides a forum for the exploration, specification, and application of object technology within the telecommunications industry. It captures specific technology requirements for the industry and serves as a liaison with appropriate telecommunications organizations. The Task Force has outlined an architecture for a CORBA-based telecommunication network management system, using ORBs to facilitate the monitoring and control of telecommunications network elements, networks and services [20].

3. Nokia DCP Prototype

3.1. Background

The problem with the TMN/OSI model is, apart from the very rich, complex protocol, that it often is implemented as two rather large monolithic applications, the manager and the agent. The SNMP is less complex as a protocol, but also implements the same manager-agent approach. In addition the Internet management is not based on an object-oriented approach. Objects would provide for better maintenance and future enhancements.

Object-oriented distributed computing will certainly be essential for the telecommunications business in the future, and the most promising approach to distributed object computing is the CORBA technology. Ideally, future network, and service management should be based on a common object-oriented approach provided by CORBA IDL. However, CORBA does not provide a network management architecture, and also world wide installations of existing CMIP and SNMP systems, makes integration of these three technologies important.

3.2. Objectives

Commercially available network management products, which support CMIP or SNMP management, do not currently support CORBA based distribution and scalability. Neither do they offer tools which enable development of CORBA based management solutions. The current network management platforms also assume a homogenous environment from one vendor only, i.e., they do not support a multi-vendor component based approach.

The purpose of the Nokia Distributed Computing Platform prototype is to provide a framework which supports the creation, management and invocation of component based distributed telecom services. CORBA and the inter-working specifications from JIDM and IIMC form the basis for the integration framework. In order to provide a development environment which allows the software engineer to concentrate on building management applications, the framework also includes tools, such as compilers to generate code and browsers to view generated meta-data, and integrated scripting tools e.g. TCL/TK.

3.3. Overview

The DCP prototype framework is built on an object-oriented software bus that facilitates both service development and service integration. Objects form the components of all services. All the objects, and thus all services, provided by the DCP prototype are available through this software bus and all the new objects can be plugged into the software bus (Figure 4.). The technology providing the software bus is CORBA.

CORBA IDL becomes the contract that binds the framework together. Programmers using the framework do not need to know where the object they want to use physically resides. They only need to know its name and the interface it publishes.

Proceedings of The Thirtieth Annual Hawwaii International Conference
on System Sciences ISBN 0-8186-7862-3/97 $17.00 © 1997 IEEE

1060-3425/97 $10.00 (c) 1997 IEEE

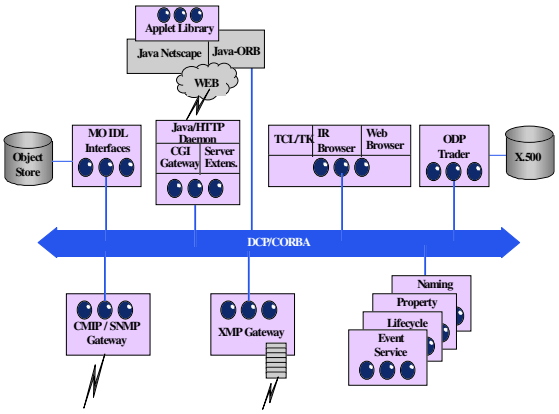

Figure 4. Nokia Distributed Computing Platform Prototype

The following chapters describes the parts of the DCP prototype, which are related to network management.

3.4. GDMO/ASN.1 to IDL Compiler

The DCP prototype includes implementations of GDMO/ASN.1 to IDL compilers in accordance with the JIDM Specification Translation and related class libraries. The compilers utilize a persistent metadata repository for storing the knowledge about the GDMO and IDL definitions. The metadata repository stores the information about the GDMO objects using the Management Knowledge Management [21] metadata object definitions. The metadata repository uses an object database for persistence. The IDL definitions are stored corresponding to the CORBA Interface Repository (IR) interface.

The code generated (C++) inherits from a set of base classes in the DCP framework, so that it integrates with the DCP prototype with minimal repeated manual coding effort.

3.5. Browsing Tools

The framework includes tools for browsing the persistent metadata generated by both the GDMO and the ASN.1 compilers. These tools include facilities to browse the GDMO metadata using a Web browser such as Netscape and UNIX shell invocation using Tcl/Tk [22] windows. Tcl (tool command language) is a scripting language for controlling and extending applications. It consists of a library that can easily be incorporated into applications. Tk extends the core Tcl facilities with commands for building Motif-like user interfaces by writing Tcl scripts.

Both Java [23] applet (Figure 5.) and HTML forms interface are provided. Alternatively, the user can browse GDMO metadata using UNIX command line invocation with Tcl/Tk. The browsers have proved invaluable for testing purposes, and extensions to view 'live' objects alongside the definitions are under development.

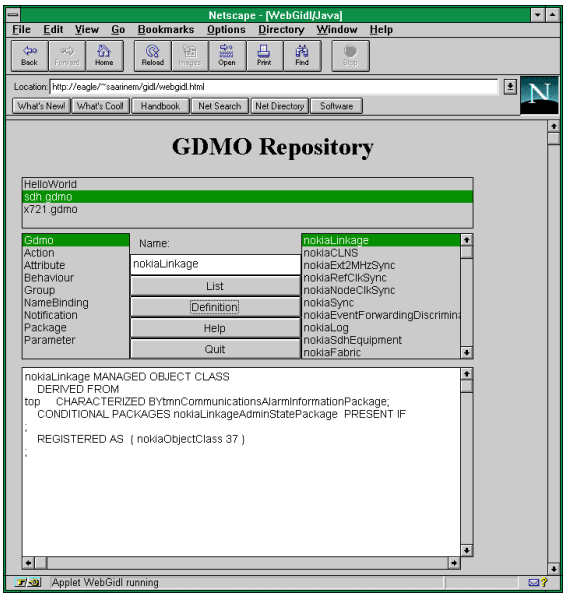

Figure 5. Java Applet for GDMO Metadata Browsing

3.6. String PDUs

It is possible to represent all the structures (syntax) used to build CMIS(P) messages using a string representation. This allows the building of management entities, which send and receive string messages. The approach used in the DCP prototype follows the one outlined in [24].

The string API is based on ASN.1 and hence allows any syntax defined in ASN.1 to be passed. The string messages can be passed as part of commands issued using a scripting language e.g. Tcl/Tk. The disadvantages are that integer and real values must be converted into/from the string format, and that messages can get large.

Each message is built as a string which includes three basic pieces of information. These are the ASN.1 module name, the ASN.1 type name, and a string containing the value. The value strings are built as label-value pairs, i.e. each primitive element of the ASN.1 data structure is input as a <label> <value> pair. The <label> portion of the pair is only required when it is necessary to indicate the resolution of a CHOICE, the omission of

OPTIONAL fields in a SEQUENCE, or the identity of fields in a SET (since they may occur in any order).

The string PDU package in the DCP prototype is used for building, parsing and passing CMIP string PDUs using CORBA as transport mechanism.

The architecture of the package can be divided into three separate areas: ASN.1 data type classes, CMIP data type classes and builder classes.

The ASN.1 classes are used to represent the basic ASN.1 data types such as Integer, Real etc. The CMIP classes represent the CMIP datatypes. These classes use the ASN.1 classes to compose a more complex data types. The builder classes are used to create CMIP String PDUs.

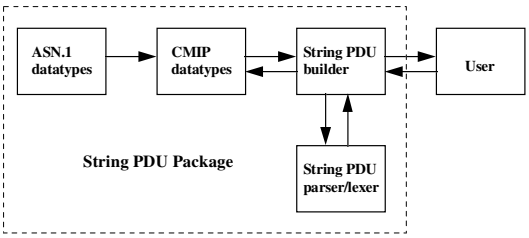

Figure 6. String PDU Package.

Figure 6. illustrates how the separate subcomponents are associated (an arrow represents use/create type relation).

3.7. CMIP/SNMP Gateway

The CMIP/SNMP gateway enables management of SNMP devices by using CMIS services.

The gateway supports sending and receiving of CMIP and SNMP protocol events and timer events, and dispatches methods to managed objects also residing in the gateway.

The managed objects of the OSI/Internet management gateway communicate with Internet agents by using the SNMP protocol to carry out the operations invoked through their IDL interfaces.

The managed objects derived from base classes in the DCP prototype represent the NMF translation of Internet MIBs to ISO/CCITT GDMO MIBs. Only the managed objects realizing a part of the translation of Internet MIB-II to GDMO MIB will be implemented in the first phase of the project.

The CMIP/SNMP gateway conforms to CMIS service emulation requirements of a basic proxy as specified by NMF. A basic proxy emulates CMIS kernel services, and supports scoping and filtering for CMIS M-GET with a single attribute value assertion involving the `objectClass` attribute.

The naming service of the DCP prototype, providing a X.500 [25] names library interface based on Common Object Services Specification (COSS) [13] Naming Service, is used by the gateway to register managed objects and to enable scoping. The X.500 names-library allows conversion of X.500 names to the CORBA names.

The gateway also uses the CORBA Life Cycle Service of the DCP prototype to create a managed object corresponding to the managed-object class (object identifier) received as a parameter of CMIS M-CREATE request.

The managed objects of the CMIP/SNMP gateway take care of the actual translation of CMIP requests to SNMP requests and SNMP responses to CMIP responses. The gateway supports a forest of object instance trees, each contained by the root object, with one system object instance for each supported SNMP agent, and one system object instance for the gateway itself.

The system object instance corresponding to the gateway itself contains all object instances representing resources of the gateway, e.g. the `cmipsnmpProxy` and `cmipsnmpProxyAgent` object instances.

A system object instance corresponding to an SNMP agent contains all object instances representing objects that are implemented in the Internet MIB.

Each `internetSystem` managed object representing the GDMO translation of the Internet MIB's system group also maps all traps from an SNMP agent to `internetAlarm` notifications defined by NMF.

CMIP protocol messages are received and sent as string PDUs over CORBA, as described in the previous chapter.

The CMIP/SNMP gateway can handle several simultaneous managers and SNMP agents (Figure 7.).

Figure 7. The CMIP/SNMP gateway.

Internet MIB objects are translated to managed objects of the CMIP/SNMP gateway in accordance with [17]. For example, groups and row objects of Internet MIB are managed objects in the gateway. Scalar objects of a group or table entry object are attributes of the managed object corresponding the group or table entry object, respectively. Conceptual table objects (i.e. those that do not have any MIB variables, such as `ipRouteTable`), are not mapped to managed objects. This means that, e.g. `ipRouteEntry` managed objects are contained directly by the `ip` managed object.

Proceedings of The Thirtieth Annual Hawwaii International Conference
on System Sciences ISBN 0-8186-7862-3/97 $17.00 © 1997 IEEE

The CMIP/SNMP gateway is responsible for emulating CMIS services by mapping to SNMP messages. The stateless approach is used in which the gateway does not maintain the Internet MIB's data. Instead, for each received OSI management service request, the gateway generates one or more Internet management service requests (i.e. SNMP requests) to the Internet agent in order to achieve the function of the OSI management service request. A problem with this approach is that certain OSI management notifications which result from a change of state in the MIB cannot be emulated, since many such changes don't cause the Internet agent to send a trap, and a comparison of states in the gateway is impossible because the Internet MIB data is not replicated.

The stateful approach would require the gateway to replicate the Internet MIB locally, and to send periodic requests to SNMP agents to keep the replicated MIB current. The gateway would then try to fulfill each incoming CMIS request by using locally-replicated MIB data, instead of sending appropriate requests to the SNMP agent. The stateful approach could usually provide better response time, but has the drawback that the data retrieved might not be current. The poll frequency used to update the locally-replicated MIB has a significant effect on the accuracy of the response.

The stateless approach was selected because it is far less complex and it is also the approach used in the ISO/CCITT and Internet Management Coexistence (IIMC) documents of NMF.

3.8. Nokia XMP++

The Nokia XMP++ [26], [27] classes enable building of C++ based applications on top of the X/Open XMP API [28].

The XMP provides an industry-standard API to CMIP and SNMP management protocols. Using the XMP API also involves X/Open's XOM API [29], through which OM objects are created, manipulated and destroyed. OM object instances, which in practice are complex C structures, implement protocol messages.

The integration of XMP++ with the DCP prototype allows building of OSI/CORBA gateway applications.

The XMP++ supports application development as specified by OSI systems management [5]. The XMP++ interface-library includes both the features supporting a managing process (manager) and an agent process (agent).

In the role of an agent, the XMP++ supports the propagation of operation requests. A scoped service operation request can involve several managed objects, that are managed by different agents located in different nodes of the network. This means that the request must be propagated to other agents from the agent, which originally received the request.

In the managing process XMP++ facilitates several simultaneous managers. The manager process can thus connect to different network solutions, i.e., the manager acts as Mediation Device, which can be useful especially in telecommunications network management.

The XMP++ supports the XMP API Draft-3 [30], the XMP API Preliminary Specification [31], and user managed Association Control Service Element (ACSE) [32] control functionality of the final XMP API [28] version. The XMP API library implementation, which has been used, is the one provided by the HP OpenView Distributed Management (OVDM) platform [33]. Figure 8. presents the integration of the DCP prototype and the HP OVDM platform.

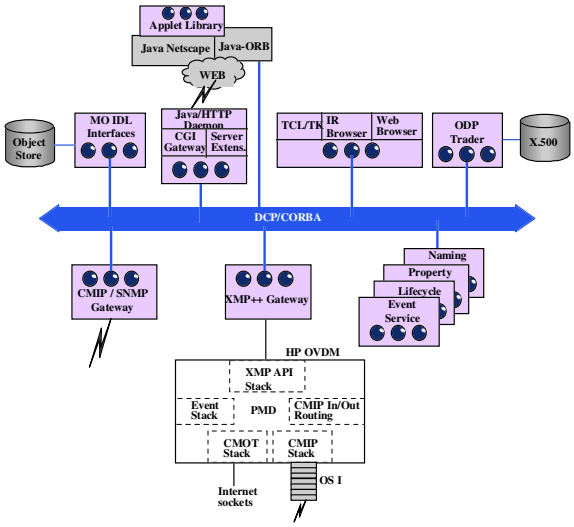

Figure 8. Integration of the DCP prototype and HP OpenView DM platform.

3.8.1. XMP++ Class Hierarchy

There are three interface layers in XMP++, represented by three groups of classes: `netHandle`, `xmpPlatform`, and `xmpWkSpace` and `xmpSession`.

The `xmpWkSpace` and `xmpSession` classes interface directly with the XMP API; respectively they correspond to the XMP API Workspace and Session concepts. The `xmpPlatform` class replaces the XMP API Workspace and Session concepts with a single management connection concept. It is intended that the `netHandle` and its derived classes are the interface classes for the network management applications to access the network. The `netHandle` class uses the

xmpPlatform class. It also uses the event handle class (Reactor class) from the ADAPTIVE Communication Environment (ACE) library V3.3 [34] for receiving messages from the network and for catching signals from the system. ACE is a collection of reusable C++ libraries and object oriented framework components. The ACE components are freely available.

The OM and message classes are the C++ wrapper classes of the XOM data structures for the parameters of the functions defined by the XMP API, e.g. for the parameters of the mp_release_req() function. For each OM data structure defined in the XOM API, there is an OM class. All the OM classes are derived from the same base class, omDescriptor. Depending on the contents of the OM data structures, each class provides a number of set and get operations.

There are xmpReceive and xmpSend base classes for receiving and sending messages over the network. The application uses the xmpSend class in conjunction with OM classes to pass a message to the peer via XMP++. Similarly, the user uses the xmpReceive class in conjunction with the OM classes to receive a message.

XMP++ provides both Automated Connection Management (ACM) enabled mode, i.e. ACSE related primitives or operations cannot be sent or received, and ACM disabled mode, when the ACSE is managed explicitly by the user.

3.8.2. XMP++ Integration to the DCP Prototype

The XMP++ gateway provides a conversion between the CMIP protocol messages which are received and sent as string PDUs over CORBA, and the information of the XMP++ interface. This conversion performs the mapping of operations of the CORBA interface to operations of the XMP++ interface and protocol interworking. This also includes connection establishment, connection negotiation and communication operations with the remote OSI stack, and data concentration/buffering and formatting of information.

The class model of the gateway is based on a MIT actor-dispatcher design [35]. The dispatcher provides for demultiplexing of timer events, signal events and I/O events received through communication channels. It defines an interface for registering actors for certain events, and dispatches the incoming messages to the pre-registered actors.

The actors are derived from CORBA objects. The actor (base) class declares virtual methods for handling different events that it can register for. The derived actor classes implement the event handling in a specific manner. The proxy actors represent managed objects and take care of the mapping of operations.

The DCP_Gdmo_Disp accepts incoming messages, and forwards them to the appropriate XMP_Proxy_Act (Figure 9.).

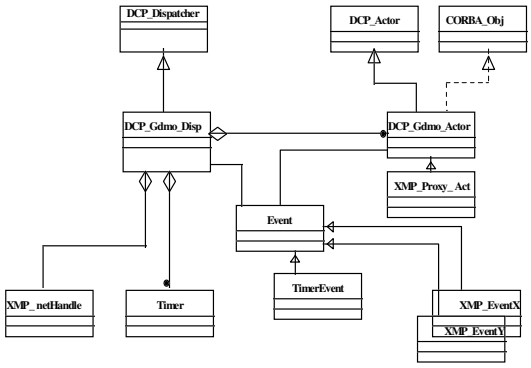

Figure 9. Actor class framework.

The XMP_Proxy_Act performs the conversion and calls the DCP_Gdmo_Disp which sends the message to the network using the XMP_netHandle class.

The DCP_Gdmo_Disp keeps track of pending operations and supports e.g. retries.

The Name Service finds the desired CORBA object reference to the XMP_Proxy_Act based on OSI distinguished names.

3.9. Web User Interface

Web based user interfaces offer some benefits for network management applications. Web browsers provide for cost-benefits, the user interfaces are easy to use, and enables mobile management in that remote access only requires access to the Internet and a Web browser [36].

The Java [23] programming language provides an environment which allows dynamic loading of program segments to the client machine. Applets are Java programs that follow a set a conventions that allow them to run within a Java compatible Web browser. This allows new versions of the user interface to be deployed faster, as they are downloaded from the server.

The DCP prototype provides a Java applet library for accessing the distributed services provided by the platform. This includes a set of applets, which provide a user interface to the network management operations. The set contains applets for manipulating tree structures, e.g. MIBs, naming trees etc. The same functionality is also provided as a HTML forms [37] based interface.

Proceedings of The Thirtieth Annual Hawwaii International Conference
on System Sciences ISBN 0-8186-7862-3/97 $17.00 © 1997 IEEE

3.10. Development Environment

The DCP prototype development and demonstration environment spans Pentium PCs (Win/NT), HP 9000/735/J200 (HP-UX 9.x/10.x), and SUN UltraSparc (Solaris 2.5) workstations.

The DCP prototype development team has experience of several ORB products including HP Distributed Smalltalk, HP ORB Plus, Digital ObjectBroker, IBM DSOM, SunSoft NEO, and IONA Orbix. These products were evaluated and compared during 1995. In the DCP prototype project, the development has mostly been done using IONA Orbix, but also using HP ORB Plus and SunSoft NEO.

To enable Java and CORBA integration, we have used the OrbixWeb product by IONA, and to enable Web based user interfaces for network management applications, the DCP prototype uses Netscape Enterprise Server.

The HP OVDM platform combined with the HP OTS 9000 OSI stack, has been used to provide full OSI communication.

All of the DCP prototype classes support persistence using the ODBMS ObjectStore by Object Design Inc.

4. Conclusions and Future Enhancements

As networks become larger and more complex, the problem of network management arises. This will necessitate a paradigm shift from centralised management to distributed management.

The DCP prototype has provided us with 'proof-of-concept for CORBA and Java based distributed network management. We have found CORBA very successful in the integration of different technologies as well as in the distribution of management functionality for load-balancing purposes. The use of Java/HTML has provided remote access to network management functionality from wherever a Web browser is available. The combined use of CORBA and Java enables development of three-tier network management solutions in weeks rather than months.

We will continue to enhance the DCP prototype. The platform will be extended by adding new services to support e.g. security [38]. In the next phase of the project performance will be addressed. Instead of transferring CMIP string PDUs over CORBA, CMIP messages could be moved in binary format. Also, such issues as fault-tolerance, multi-threading, and support of real-time processing will be considered. We also intend to rewrite some of the C++ libraries in Java to support dynamic downloading of network management functionality.

Although, we demonstrated CORBA based network management through the XMP++ gateway, it is obviously not the most efficient way to accomplish CORBA/OSI management integration. Several papers [39], [40], [41], [42], [43], [44], [45], [46] have treated this integration issue with encouraging conclusions, but also bringing forth the drawbacks involved.

Notwithstanding, it seems that the next generation commercial network management platforms will support native CORBA communication [47]. In our case it means that the DCP prototype objects can communicate directly with services included into e.g. such systems as HP's future Synergy technology [48] based platform.

During the project it has become evident, that even though current commercial ORBs are not optimized [49], the CORBA technology is mature enough to be used as a base for the telecom software platform development. However, CORBA per se is not enough, also a lot of work is needed in the area of the surrounding class libraries, i.e. the application development enabling framework.

Experience is needed in order to create an understanding of the different architecture of open distributed telecom systems, as opposed to monolithic systems. We have found the learning curve of CORBA usage bearable when the developer has previous experience of C++ programming and the use of different commercial class libraries.

Acknowledgments

The authors wish to thank the Nokia DCP prototype project team and especially the main architect Petri Nuuttila for his valuable contributions.

References

[1] Object Management Group, The Common Object Request Broker: Architecture and Specification, Revision 2.0, July 1995.

[2] ITU-T M.3010, Principles for a Telecommunications Management Network, Draft June 1995.

[3] A Simple Network Management Protocol, RFC 1157, May 1990.

[4] Shrewsbury, J. K.,An introduction to TMN, Journal of Network and Systems Management, pp. 13-38, Vol. 3, no. 1, 1995.

[5] ISO 10040, Information technology - Open Systems Interconnection - Systems management overview, November 1992.

[6] ISO 10165, Information technology - Open Systems Interconnection - Structure of management information - Part 4: Guidelines for the Definition of Managed Objects.

[7] ISO 8824, Information technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1).
[8] ISO 9595, Information technology - Open Systems Interconnection - Common management information service definition.
[9] ISO 9596, Information technology - Open Systems Interconnection - Common Management Information Protocol.
[10] Black, U., Network Management Standards, The OSI, SNMP and CMOL Protocols, McGraw-Hill, 1992.
[11] Object Management Group, Object Management Architecture Guide, Revision 2.0, Second Edition, September 1992.
[12] Object Management Group, Common Facilities Architecture, Revision 4.0, November 1995.
[13] Object Management Group, CORBAServices: Common Object Services Specification, March 1996.
[14] Schmidt. D. C., Vinoski, S., Object Interconnections, Comparing Alternative Server Programming Techniques (Column 4), 1995. http://www.cs.wustl.edu/~schmidt/report-doc.html
[15] X/Open, Specification Translation, X/Open Preliminary Specification, Draft August 9, 1995.
[16] Network Management Forum: Forum 026, Translation of Internet MIBs to ISO/CCITT GDMO MIBs, Issue 1.0, October 1993.
[17] Network Management Forum: Forum 029, Translation of Internet MIB-II (RFC 1213) to ISO/CCITT GDMO MIB, Issue 1.0, October 1993.
[18] Network Management Forum: Forum 030, Translation of ISO/CCITT GDMO MIBs to Internet MIBs, Issue 1.0, October 1993.
[19] Network Management Forum: Forum 028, ISO/CCITT to Internet Management Proxy, Issue 1.0, October 1993.
[20] Object Management Group, CORBA-Based Telecommunications Network Management System, OMG White Paper, Draft 2, January 1996.
[21] ISO/IEC 10164-16, Information Technology - Open Systems Interconnections - Systems Management - Management Knowledge Management Function.
[22] Ousterhout, J., Tcl and the Tk Toolkit, Addison-Wesley, 1994.
[23] Sun Microsystems, The Java Language: A White Paper, 1995. http://java.sun.com/
[24] IBM Corp., CMIP Run!, Vol. 2 No. 4, 94.
[25] ISO 9594-1, The Directory - Overview of Concepts, Models, and Services.
[26] S. Rahkila, S. Stenberg, XMP++: An Object-Oriented Solution For Hiding The Complexity Of Network Management Protocols, Proceedings PODC '94, ACM Press, Los Angeles August 1994.
[27] S. Rahkila, S. Stenberg, Nokia XMP++: An Object-Oriented Approach to TMN Application Development, Proceedings TINA '95, Volume 1, Melbourne February 1995.
[28] X/Open, Management Protocols API (XMP), CAE Specification, Mar 1994.
[29] X/Open, OSI-Abstract-Data Manipulation API (XOM), CAE Specification, Nov. 1991.
[30] X/Open, Management Protocols API (XMP), Preliminary Specification (Draft-3), Jan 3 1992.
[31] X/Open, Management Protocols API (XMP), Preliminary Specification, Jul. 1992.
[32] ISO 8650, Information Processing Systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element, 1988.
[33] Hewlett-Packard, HP Open View Distributed Management Developer's Reference, J1064-90008, September 1994.
[34] The ADAPTIVE Communication Environment (ACE), *http://www.cs.wustl.edu/~schmidt/ACE.html*
[35] ABCL : an object-oriented concurrent system, ed. by Akinori Yonezawa , MIT Press, 1990
[36] J. Reilly, The World-wide Web and Programming Future Broadband Network and Service Management Applications, Presented at the NWPER'96, Aalborg May 1996. http://misa.zurich.ibm.com/Consortium/doc/papers
[37] Hypertext Markup Language (HTML): Working and Background Material, http://www.w3.org/pub/WWW/MarkUp/MarkUp.html
[38] Object Management Group, CORBA Security, December 1995.
[39] Luca Deri, Network Management for the 90s, Submitted to ECOOP'96, Linz July 1996. http://misa.zurich.ibm.com/Consortium/doc/papers
[40] Della Torre, C., A Generic Distributed Service Management Test Bed Integrating CORBA and the XMP API, Proceedings ECOOP '94, July 1994.
[41] O'Sullivan, D., CORBA and Telecoms Management - can they live in perfect harmony?, Proceedings DOOC '95, October 1995.
[42] Davis, J., Du, W., Kirshenbaum, E., Moore, K., Robinson, M., Shan, M-C., Shen, F., CORBA Management of Telecommunications Network, Proceedings DOOC '95, October 1995.
[43] Park, J-T., Ha, S-H., Hong, J.W., Song, J-G., Design and Implementation of a CORBA-Based TMN SMK System, Proceedings NOMS '96, April 1996.
[44] Dittrich, A., Höft, M., Integration of a TMN-based Management Platform into a CORBA-based Environment, Proceedings NOMS '96, April 1996.
[45] Kong, Q., Chen, G., Integrating CORBA and TMN Environments, Proceedings NOMS '96, April 1996.
[46] Usländer, T., Brunne, H., Management View upon CORBA Clients and Servers, Proceedings ICDP '96, February 1996.
[47] Hewlett-Packard, Press Release, http://hpcc998.external.hp.com:80/nsmd/ov/whatisov/telecom.txt
[48] Herman, J., The Sorry State of Enterprise Management, Distributed Computing Monitor, March 1996.
[49] Gokhale, A., Schmidt, D., Measuring the Performance of Communication Middleware on High-Speed Networks, Submitted to SIGCOMM Conference, ACM, August 1996.