Proceedings of the *Second International Enterprise Distributed Object Computing Workshop; EDOC'98*, San Diego, CA, USA, November 1998

Towards CORBA-based Enterprise Management: Managing CORBA-based Systems with SNMP Platforms

Alexander Keller
Department of Computer Science, Munich University of Technology
E-Mail: keller@informatik.uni-muenchen.de

Abstract— **Apart from the well established OSI/TMN and Internet management architectures, the Common Object Request Broker Architecture (CORBA) is increasingly considered as a promising framework for Enterprise Management.**
On the downside, management platforms being able to survey and control management agents via CORBA are still hard to find on the market while SNMP-based platforms are in widespread use. The consensus on the advantages of CORBA for managing complex system infrastuctures and applications exacerbates the need to open well-established management platforms for CORBA.
This paper will present a feasible and practical approach to this problem: it allows the extension of existing SNMP-based management platforms for managing CORBA-compliant agents without the need of modifying already existing platform code. Thus, it can be considered as a step towards CORBA-based Enterprise Management by ensuring the interoperability of heterogeneous management architectures. Key to our approach is the creation of a conceptually integrated management information base on the platform side where management-related information is collected and evaluated independent of the originating base architecture. The work described in this paper can be considered as a case-study on the seamless integration of legacy systems into emerging distributed object environments.

Keywords— **CORBA, Enterprise Management, SNMP Platforms, Distributed Systems Management**

I. Introduction

Today, distributed systems based on the client/server paradigm have made their way into commercial IT infrastructures. The price of the flexibility gained is a more complex technical management of the computing environment. Efficient operation and administration requires an integrated management, in other words administration should be based on a single conceptual framework, namely a management architecture. This task has become even more complex through the recent introduction of additional management architectures (see e.g. [1], [2]); on the one hand, there is the well-known OSI management architecture that is primarily used in the telecommunications area; on the other hand, many IETF working groups are extending the scope of the Internet (SNMP) management architecture, i.e. applying it to the management of distributed environments. Additionally, CORBA [3] is becoming increasingly important for management applications. Whereas this architecture has not been developed specifically for management applications but to generally support communication and cooperation within distributed applications, it seems promising to use it for management purposes, too. Unfortunately, the current state of CORBA-based management of end systems and applications leads to an isolated management island. This produces a situation where not only the managed resources introduce a large amount of heterogeneity, but also the different standardized management frameworks. Therefore, bridging between the mentioned management architectures is an important research topic today. Our overall goal is to make integrated management feasible in an environment consisting of different management architectures.

The aim of the work described in this paper consists in helping to establish a "native" CORBA management environment that can also cope with managed systems in "classical" management environments that rely e.g., on the Internet management architecture: The management system should be ready for CORBA-based management but also take advantage of the functionality delivered by already existing management platforms. We achieve this by establishing a conceptually integrated management information base where management related information can be collected and evaluated independent of the originating management framework. It contains data of resources as well as events which have been raised somewhere "out in the network". The aim of our work lies in developing mechanisms which can be used for integrated but also distributed management of services, systems and networks.

In this paper, we will focus on two major integration problems, namely the transfer of events between both (CORBA and SNMP) worlds and the use of the management platform services by CORBA-based management applications. Therefore, we will explain the functionality of the Event Filtering and Logging services of the platform and a method to build a bridge for events flowing between CORBA and the platform. Events emitted by CORBA agents can be transparently routed via the bridge into the event services of the management platform, making use of its filtering, displaying and storage capabilities. Platform filters can be set up using CORBA interfaces. By using regular features of the platform, these events may trigger management actions. We will also explain the functionality of the platform Topology Services, which manage topological relationships between managed objects and show how these generic services can be used in a straightforward manner to visualize the relationships between CORBA-based services and objects. For this visualization, we have developed a simple relationship model tailored for the LEO/MEO satellite environment described in section II. To achieve the integration of events and access to platform services, we made use of the features of the CORBAservices [4] by building "wrapper" objects which open the programming interfaces of the platform and its services to objects anywhere in the CORBA environment.

The structure of the paper is as follows: Section II introduces a real-life management scenario, namely the CORBA-based management of LEO/MEO satellite constellations. They are an extreme example of a distributed environment and have management requirements similar to other distributed systems. We believe that distributed environments should also be managed in a distributed manner and therefore propose CORBA as management architecture. As currently no CORBA-compliant management platforms exist on the market, there is a strong need for achieving interoperability between well-established SNMP-based management platforms and the managed nodes represented by CORBA agents. The third section of the paper presents the different possibilities for bridging the gaps between heterogeneous management frameworks and gives reasons why we decided to perform the integration on the side of the managing system, i.e. making an SNMP-based management platform suitable for CORBA. It also discusses the different platform-side integration alternatives and gives an overview over the services provided by the platform that are most useful for our purpose. The techniques how this integration can be achieved will be described in section IV which presents the results of a study and a trial implementation of a CORBA-compliant management platform which integrates a "conventional", i.e. SNMP-based network management platform product (*IBM NetView for AIX*[1]) with an Object Request Broker. Section V concludes the paper and gives an overview of further steps.

[1]Since the IBM/Tivoli merger, the product is now called *Tivoli TME 10 NetView*.

II. Managing LEO/MEO Satellite Constellations

Since 1965, several communication services (long-haul telephony, TV distribution, network relay, maritime and land communications) are provided by *geosynchronous earth-orbit (GEO)* satellites residing at an altitude of about 36.000 km above the equator. Due to its high altitude, one GEO spacecraft covers roughly a third of the earth surface; as a consequence, three equally-spaced GEO satellites provide full coverage of the earth, with the exception of the polar regions. An advantage of GEO satellites comes from the fact that they move at the same pace as the earth; thus, only very few ground stations are needed to handle their signals. On the other hand, apart from their high costs, GEO satellites present drawbacks for telephony transmission: Due to the long signal paths, the signal propagation delay (about 260 ms of round-trip propagation time) is significant. Furthermore, due to power restrictions, it is not possible to provide a direct satellite connection for handheld mobile phones ([5]).

During the last years, two approaches have been developed to overcome these restrictions by placing satellites closer to the earth: *Low earth-orbit (LEO)* satellites reside typically between 500 and 1500 km above the earth and have very low round-trip delays (10 to 30 ms) while their *middle-earth orbit (MEO)* counterparts are 5000 to 12000 km high and have round-trip delays of about 100 ms. For global coverage, obviously, the number of required satellites varies with altitude. Examples of commercial LEO projects are Iridium (780 km, 66 satellites) and Globalstar (1400 km , 48 satellites). A typical example of a MEO global satellite system is ICO (10335 km, 10 satellites). Figure 1 gives an overview over characteristic GEO and LEO/MEO satellite configurations.

A property of LEO/MEO satellites grouped into *constellations* is that they move at a different pace than the earth; it is therefore not only necessary to provide a larger number of ground stations, but the number of satellites in their reach varies with time.

Currently, satellite ground segments are composed of mission control centres and satellite control centres linked to ground stations. Mission control centres aim to manage satellites in terms of payload. They use satellite control centres in order to configure satellites, manage the onboard resources and ensure the orbit control. Finally, to communicate with satellites, control centres use a network of ground stations. The geographical location of these ground stations allows a complete coverage of the satellites orbit. As described above, in the past, typical space systems used massive spacecraft, huge control centres and large ground

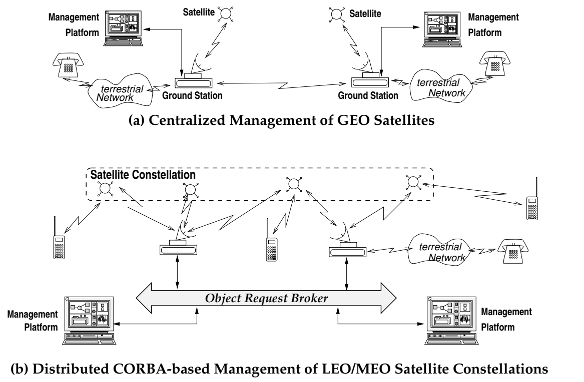

Fig. 1. Centralized vs. distributed satellite management

stations. As depicted in figure 1(a), every station is equipped with its own management platform. In the future, however, constellations of LEO/MEO micro-satellites will be deployed. The control of these constellations will be based on many micro-ground-stations, inter-communicating through a network in order to keep expenses low and time-to-availability short.

This approach to satellite management follows a structure similar to that employed in other distributed systems. However, it poses an extreme example of such a structure, since it requires high system reliability. Thus, resolving the problems related to space systems would, as a by-product, solve the current management problems of other distributed systems with high availability requirements, such as power-line networks, telephony systems, or digital TV/data networks. Unfortunately, very limited experience with the distributed operations management of such ground station networks exists in the space industry. This constitutes a major obstacle for the deployment of micro-satellite constellations. Experience gained from other fields, such as Civil Aviation Air Traffic Control, shows that international co-operation and interoperable distributed management systems are essential to keep investment costs at a reasonable level, and to enable scalability of future systems. This experience also shows that new management paradigms should be carefully designed to address the specific requirements of the target systems.

With a constellation of satellites, there always will be several ground stations simultaneously in visibility as depicted in the lower part of figure 1. Although it is possible to concentrate enough equipment in a set of ground stations to allow each of them to handle its task correctly, a more efficient solution is to dynamically reallocate tracking slots among the ground stations. This way this otherwise redundant capacity will be better utilised. Another issue is an improved visibility pattern when tracking satellites. When a ground station is used to track one or several satellites, its work plan may be prepared by a management system in advance and it is possible to get the work done using scattered yet co-operating ground stations.

To meet these challenges, it is necessary to define up-to-date operation management software for ground stations. Existing systems management frameworks such as HP OpenView, Tivoli TME 10 and CA Unicenter TNG are relevant to such management contexts and may be used to reduce costs. The management requirements of LEO/MEO satellite networks and an approach for representing their topological relationships with available management platforms are described in [6]. However, many non-standard equipments remain to be managed at a lower level, and, at a higher level, space operations management not addressed by the standard management tools must still be performed. Therefore, an important research problem consists of creating a framework of inter-operable components. These components satisfy a market need which is to build rapidly and on-demand an operations management software to handle a network of ground stations. These stations are intended to co-operate via CORBA (see figure 1(b)) in order to manage a constellation of satellites.

To support this effort, application frameworks based on distributed processing standards such as CORBA and performant and easy-to-implement scripting languages like Java present several advantages to lower the costs and shorten the delay of building operations management systems:

- Object-oriented frameworks provide reusable design and components based on domain-specific knowledge, while distributed computing using CORBA allows the integration of different existing frameworks found in other system infrastructures and middleware layers, hence a market target of fast time-to-availability operations stations;
- Interoperability among systems makes it possible to consider a more efficient resource sharing among different operators of widely varying skills, resources and geography ([7], [8]). This would enable satellite servicing and sales to many customers from many different geographical areas, thus vastly extending the market potential;
- Programming languages such as Java may be used, at a low cost, to add SNMP capabilities to communicate with otherwise unmanaged equipment. The *Java Dynamic Management Kit (JDMK)* in its current version includes an SNMP agent toolkit and may provide OMG IDL interfaces for Java classes in the future;
- The mobile agent paradigm as described in the OMG Mobile Agent Facility [9] can simplify management tasks in many ways. For instance, CORBA/Java-based mobile agents can resolve software installation

problems by allowing on-demand downloading of appropriate management instrumentation from a central location. Java also allows easy modification of agent-side software – a task that cannot be easily accomplished with SNMP (see e.g. [10], [11]) although the IETF Distributed Management Framework (disman) aims at resolving this drawback.

The market of such operations management systems may be extended from the space stations operations management to other types of "networked facilities" management such as fixed services transmitting stations, power lines, digital-TV networks or digital data networks. Operations management is also emerging for terrestrial transportation, telecommunications and messaging, and operations scheduling: The control centres are currently equipped with software and tools on an as-is basis, due of the high cost of tailoring operations management centres. The availability of reusable application frameworks may open a new and rich market there.

III. Managing CORBA-Environments with SNMP Platforms

The CORBA-based management of LEO/MEO satellite constellations through ground stations (as described in section II) implies the availability of a CORBA-based management platform. Although several platform vendors claim to deliver CORBA-compliant management platforms, this statement is in almost every case only true for the internal platform communication mechanisms: These systems are based on a proprietary ORB and are thus not CORBA-2.2 compliant, i.e. they cannot inter-operate with other ORBs and therefore not with CORBA agents located on remote devices. It is therefore necessary to enhance SNMP-based management platforms for handling CORBA requests. This section discusses the principal alternatives for bridging the gaps between different management architectures and describes the concept of our solution based on platform services and standardized CORBAservices.

A. Interoperability between Management Architectures

As figure 2 shows, there are basically three different strategies for achieving a seamless interaction of components located in different architectural domains (see also [12]):

The first approach consists in the integration at the resource level, i.e. the managed systems support more than one management protocol; they are equipped with **Multiarchitectural Agents**. This is usually unfeasible for the following reasons: SNMP agents, for example, are often used to perform monitoring of simple network devices. They should not consume a large amount of resources and are usually built into the firmware of the device; the implications are that these agents can neither be enhanced to support another management protocol nor should they introduce additional complexity. For an in-depth discussion of this subject, the reader is referred to [13].

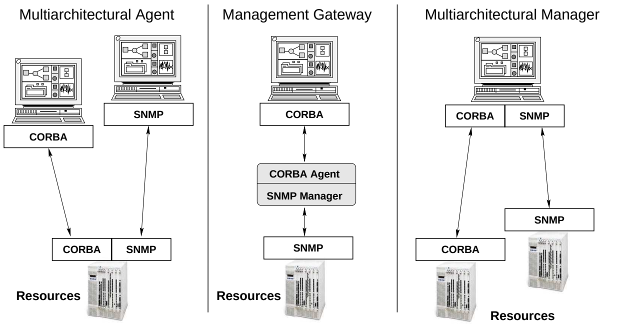

Fig. 2. Three approaches for interoperability

An alternative is the **Management Gateway** approach. It is then possible to manage services, systems and networks in different management architectures from a single point of control as demonstrated in [14] and [15]. It is even possible to apply the power of a management architecture with rich functionality to any resource in the different architectural domains. In management architectures having no notion of a management functional model such as the Internet framework, the application of management functionality "borrowed" from other architectures is particularly useful. Furthermore, a translation algorithm has already been specified by the OpenGroup *Joint Inter-Domain Working Group (JIDM)* [16] that allows the transformation of Internet SMI to CORBA IDL; the appropriate interworking architecture has been recently submitted [17] to the OMG in response to the *CORBA/TMN Interworking RFP*. Thus, the standardization process for CORBA/SNMP management gateways is currently underway. Furthermore, our experiences with building management gateways [18] have shown that implementations of required CORBAservices (like e.g., Notification, Topology) are also not available on the marketplace today. At their current stage, one has to use proprietary mechanisms for achieving interoperability; another issue is that management gateways are complex pieces of software and are likely to become bottlenecks with respect to performance [19]. However, one can expect that this situation will be improved in the near future as the recent proposals for new services will be adopted.

The third solution is to place the burden of integrating the different architectures on the managing system. Such a **Multiarchitectural Manager** supports a set of management protocols which are implemented onto the platforms' communication stack. A conversion between different management protocols is therefore not necessary. The transformation of the management information descriptions is often handled by tools bundled with the management platform like MIB-compilers and therefore need not be handled by the developer. Furthermore, the service APIs of available management platforms can easily be accessed thus yielding the opportunity of reusing a large amount of platform services. In our opinion, this solution represents a good compromise between the quality of the integration and the amount of work that has to be undertaken.

Although the gateway-based approach seems to be very promising as a long-term solution, the manager-based integration approach currently demands the least amount of work. Thus, we decided to follow the integration on the side of the managing system for the design and implementation of our interoperability solution. In the following two sections, we will therefore discuss two approaches for doing the platform-based integration and identify the reusable platform services. Section IV explains how our prototype implementation works.

B. Building Multiarchitectural Managers

There are basically two possibilities for the integration of CORBA as a new management architecture into an existing management platform: The first approach makes use of the platform's communication interfaces by integrating a new protocol into the protocol stack. The second consists of establishing IDL "wrappers" around the interfaces of the platform services so that the whole platform appears as a set of CORBA objects. We will discuss each approach in turn.

B.1 Integration at the Protocol Level: XOM/XMP-Approach

The goal of this approach is to perform the integration work on the "lowest level" as possible. The XMP/XOM (*X/Open Management Protocol, X/Open OSI-Abstract-Data Manipulation*) approach [20] has been for a long time the best-known representative of this idea. Initially, its purpose was to enable architecture-independent management by presenting a uniform API for the SNMP and CMIP management protocols and a mapping of ASN.1 constructs into C data types. From today's point of view, this approach failed due to the missing transparency with respect to the underlying protocols. Additionally, its complexity led to the fact that it was only used as a C-based interface to CMIP while the SNMP communication is handled by a separate and easier-to-use dedicated SNMP stack. Furthermore, the recently standardized TMN/C++ API developed by the NM-Forum and described in [21] is now replacing XMP.

The main problem for using XMP in conjunction with CORBA stems from the fact that XMP assumes that every incoming *Protocol Data Unit (PDU)* has a fixed format. While this is true for SNMP and CMIP, CORBA event messages are a method call on an arbitrary *consumer* object (see also section III-C.2). Of course, the *Postmaster*-daemon being responsible for the maintenance of the XMP-stack in network management systems like *HP OpenView* and *IBM NetView* could act in the role of an event consumer. Consequently, if typed event communication is used, the consumer interface must be modified every time if new event types have to be considered. This happens very frequently, e.g. if new resources are introduced into the CORBA environment. Furthermore, the above described extensions of the Postmaster-daemon require the availability of its source code – if commercial platforms are used, this is obviously impossible.

B.2 Encapsulating the Platform Service Interfaces

An alternative to the XOM/XMP approach is the direct access to services provided by the platform like the facilities described in the following section III-C. This is possible because a large part of the platform services is available to third-party applications through C-APIs. Therefore, we decided to encapsulate the platform interfaces with IDL wrappers so that the platform appears as a set of CORBA objects. The use of wrappers is a standard technique for migrating legacy systems into new object-oriented environments. This is feasible because the "perception is reality"-principle applies to any kind of object-oriented system: It is not necessary that a system is implemented in an object-oriented way; the important thing is that its interfaces look like objects. Other CORBA objects can then communicate with the platform the same way as they would do with "native" CORBA objects. On the other hand, the use of platform-specific service APIs naturally implies the loss of independence from concrete products; the portability of the solution is thus restricted.

C. Necessary Platform- and CORBA-services

Management platforms available on the market (e.g., *IBM NetView*) usually contain the following key features:

- *Topology Management*: Network nodes are dynamically discovered and polled for configuration information. This information is stored in a database and a graphical view of the network topology is presented at the platform's user interface. The database can be accessed through a programming interface by management applications which want to retrieve or supply topological information. In addition to the generic discovery and status polling applications for IP networks, the *Generalized Topology Manager (GTM)* application collects and maintains information about "non-IP" nodes.

- *Event Management*: The platform is able to receive and process network events via SNMP (traps) and CMIP or CMOT, respectively (event reports). The platform *Event Management Services (EMS)* receive, filter and store the events in log files that can then be accessed by users or by applications via a programming interface. Events can be presented on the graphical user interface. The platform allows to filter events by application-defined criteria and to forward them to registered applications. This allows automatic triggering of actions in case of pre-defined network situations.
- *Performance Monitoring*: Network administrators can define thresholds on the attributes of managed objects. An event is generated automatically when a previously defined threshold is exceeded so that an operator or an application can be informed e.g., whenever the QoS (Quality of Service) of a network resource falls below a critical level.
- *Configuration Application*: The platform contains a Browser for *Management Information Bases (MIBs)* that can be used to query a device's configuration by reading MIB information and to configure a device remotely by modifying its MIB variables.
- *Monitoring the State of Resources*: All network resources are periodically polled for state changes. A change of state is visualized through a colour change of the icon which represents the resource on the graphical user interface.
- *Integration of Third-Party Applications*: Although state-of-the-art platforms provide a number of basic services that are needed for network management, these services can be extended by integrating additional management applications that are tailored to specific needs. Many platforms also contain a set of APIs that allow the integration of user-written applications.

The services described above fulfil basic requirements for the management of networks and computer systems. For a platform that manages CORBA objects, these services must be enhanced to support the management of distributed applications and services in addition to network resources and computers. This is due to the fact that not only the amount of fine-grained managed objects representing the application components tends to be much higher than the number of networked devices but these managed objects are extremely dynamic: The installation of new applications must be monitored and the starting and stopping of application processes must be surveyed and controlled by the managing system. This poses high requirements w.r.t. scalability.

Our work focuses on the Event and Topology Services of the platform which provide a common basis for all kinds of management services. In addition, both access and maintain the platforms' Management Information Repository. Therefore, they are the main building blocks for the development of further CORBA management services like configuration applications. The **handling of events** is a major aspect of resource management, especially for detecting fault situations. Since the availability of CORBA-based applications and services depends on the functionality of computer systems and network components, the application and service management may not be separated from systems and network management. Therefore, interfaces have to be developed that allow CORBA events to be received and processed by the manager in the same way as SNMP traps; they will be described in section IV-A. **Topology Services** manage topological relationships between managed objects. In section IV-B we will show how these generic services can be used in a straightforward manner in order to visualize the relationships between CORBA objects.

C.1 Event Management Services provided by the Management Platform

SNMP-based management platforms have several components for receiving and processing SNMP traps. Received events are forwarded through a chain of processes, each performing specific actions on them. The services of a filter process can be used to send certain kinds of events to registered management applications in order to allow automatic actions to be taken in case of unusual circumstances. Several filters can be active concurrently, each one defined for certain event attributes like source, type, time, etc. A log daemon stores all event data in a log file. This file can be accessed by users and applications to determine the event history of the network. A GUI application presents a graphical representation of events in so-called *Dynamic Workspaces* at the platform user interface. These dynamic workspaces can be used to cluster all incoming events or only those according to a specific, pre-defined filter.

C.2 The CORBA Event Service

The CORBA Event Service [4] standardizes the transfer of asynchronous notifications between objects. An object that generates events is called *supplier* whereas an object which receives events is called *consumer*. A supplier passes an event to a consumer by invoking an appropriate method on the consumer interface. Suppliers and consumers can be decoupled from each other by *event channel* objects. An event channel forwards all events it receives from any of its suppliers to all the consumers that have registered with the channel. The events themselves can be either of *generic* or *typed* format. A generic event has one single parameter of the OMG IDL datatype *any*, whereas a typed event may have an arbitrary number of parameters which can be of any OMG IDL datatype.

In the future, the currently standardized CORBA **Notification Service** developed by the OMG Telecommunications Domain Task Force will provide much more flexible event filtering capabilities. It is then possible to establish user-defined event filters by assigning priorities to events, generate timestamps, and introduce QoS criteria for the handling of events. Mechanisms for modifying the persistency properties of events will also be available.

D. Topology Services

Topology Services are needed to visualize and monitor managed resources on the graphical user interface. In network management this means displaying symbols for all existing network components and computer systems, showing the connections between them (e.g., on the IP level) and visualizing the state of a resource (up, down, test etc.) through the colour of the symbols.

D.1 Platform Topology Services

The main components of platform topology services are discovery modules that find managed resources, store information about them in the database of the management platform and poll them for configuration and state changes at regular intervals. The information in the database is then used to display a model of the network and its resources in a hierarchy of so-called **submaps** on the user interface. An operator may navigate through the submaps and examine their content by selecting the symbols representing managed resources. By extending the topology tervices for the management of CORBA systems and applications, we achieved not only a common user interface for monitoring network components, computers and applications, but also an integrated Management Information Repository inside the platform. Section IV-B will describe this in detail.

D.2 The CORBA Topology Service

The topology service whose goal is to maintain and manipulate the logical topology of distributed systems is currently undergoing the OMG standardization process. It defines mainly three generic APIs that are particularly useful for topology management purposes: The *Metadata Manager API* allows the definition of rules for topology relationships between **object classes** based on criteria like their type or the number of objects that take part in a topological relationship. The *Data Manager API* makes use of the former API for establishing, maintaining or deleting relationships between **object instances**. Every change requires a lookup for ensuring that the rules defined in the metadata are fulfilled. The *Query Manager API*, finally, is used by management applications for inquiries concerning topology relationships.

Although these features are particularly useful for managing distributed systems, the standardisation of the topology service has been stopped in december 1997 because it seemed that recent developments in the OMG (Portable Object Adapter, Meta-Object Services) already provide parts of the topology service functionality.

IV. The CORBA/SNMP Management Platform Prototype

This section describes our prototype which has been implemented using the following products: The SNMP-based management platform is *IBM NetView for AIX version 4.1*; the CORBA development environment is the *IBM SOMobjects Developer's Toolkit version 3.0*.

The structure of this section is similar to section III; we will therefore describe in subsection IV-A how the implementation of the event handling mechanism works and focus in subsection IV-B on our solution for topology management.

A. Event Handling

Our goal was to open the NetView *Event Management Services (EMS)* for CORBA events in addition to SNMP traps in order to create a single point of reception and processing of events from network, systems and application management. Our approach is based on the CORBA Event Service, which has been described in section III-C.2. We have decided to use event channel objects and typed event communication because we believe that the generic event format is insufficient for passing the complex information needed for managing CORBA-based distributed applications. Event channels have the following useful properties: The amount of event consumers and their object references are completely transparent to the suppliers; these send their events directly to an event channel and thus only need to know one reference (the one of the event channel) although an arbitrary number of consumers may receive the event. Typed event communication yields not only the advantage of being able to distinguish between application, network and resource events but presents also the opportunity of structuring the events stemming from CORBA agents according to its severity and w.r.t. different categories like fault, topology or security. This gives us the possibility to select events according to their kind and severity: it is feasible to use the push-model between the event channel and the consumer (i.e., the platform) for important events and to rely on pull-type communication for less critical events.

Two interfaces were defined between the CORBA environment and the management platform (see also figure 3):

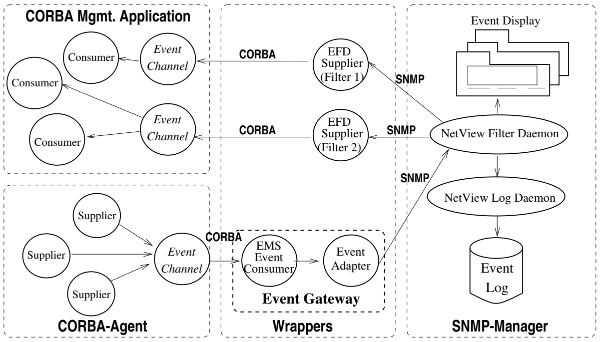

Fig. 3. Encapsulating the platform event handling APIs

1. An interface that allows CORBA events to be sent to NetView. Once events have been received, they can be filtered, logged and displayed on the user interface of the platform in the same way as SNMP traps.

2. An interface that allows CORBA applications to register themselves and their filters with NetView to receive selected events.

We will discuss the implementation of both interfaces in the following sections.

A.1 Interface 1: Reception of CORBA Events by NetView

Since NetView is not able to receive CORBA events directly, they have to be converted into an appropriate format. For this purpose, we have developed an event gateway whose instances offer a TypedConsumer interface for CORBA management events. The event gateway receives events from suppliers (i.e., CORBA-compliant management agents) via an event channel and transforms them into SNMP traps; these can then be handled by NetView. The actual event-to-trap conversion is carried out by the two object classes that form the event gateway depicted in figure 3: The `EMS_Event_Consumer` object registers itself with the event channel in order to collect all kinds of events sent out by the suppliers. The transformation of the received event messages into SNMP traps and their forwarding to the management platform is done by calling the appropriate methods of the `Event_Adapter` objects. An enterprise-specific SNMP trap-PDU is then created, filled with the delivered parameters and finally sent to NetView. The mapping of specific kinds of typed events to enterprise-specific traps is similar to the approach described in [17]. These enterprise-specific traps must be defined on the side of the management platform either manually or by setting the trap daemon (*trapd*) configuration by corresponding API calls. The mapping rules between the names of the CORBA events and their enterprise-specific trap counterparts must also be defined. Such a mapping table can either be defined in a flat file or as a separated object or in the `EMS_Event_Consumer` object. We decided to implement a common, predefined set of asynchronous notifications and stored these in a file that was parsed by the `EMS_Event_Consumer`. The IDL definition of an `Event_Adapter` object class is given below:

```
interface Event_Adapter : SOMObject
{
 void create_pdu(in long tid, in string src);
 void add_arg_long(in long arg);
 void add_arg_string(in string arg);
 void send_pdu();
}
```

A trap-PDU is created and initialized by calling the method `create_pdu`; the purpose of this method is to determine the enterprise-specific trap-identifier (`tid`) and the IP-address of the source (`src`, determined by the *host*-parameter of the CORBA event) and to insert these parameters into the newly created trap-PDU. The two `add_arg_()` methods are used to write the arguments of the event as variable-bindings into the PDU. Finally, `send_pdu()` sends the PDU to the platform. This IDL interface can be considered as the "wrapper" for the appropriate methods of the NetView SNMP-API [22]. As this API is implemented in C, the C language mapping was used to establish the binding between the IDL interface and the SNMP API. The CORBA development environment that we used requires that every object class must be derived from `SOMObject`, the base object class.

A.2 Interface 2: Platform-based filtering of CORBA Events

The second interface allows CORBA-based applications to receive events which have been filtered by NetView. The NetView filter API supports filtering according to the following criteria: Enterprise ID, IP-address of the agent system, event time, event logged time, generic and specific trap types, frequency of occurrence. Filter expressions are represented by strings that consist of keywords representing the above criteria, values, logical and numeric operators.

As we want to ensure that CORBA-based management applications make use of the platform filter functionality, filtered SNMP-traps had to be converted back again into CORBA events. In the terminology of the CORBA event service, the platform must therefore be able to adopt the "supplier" role for filtered events. To make this possible, the object class `EFD_Supplier` was defined. This name of the object class was chosen because these objects have a functionality that is similar to the OSI *Event Forwarding Discriminators (EFD)*. An `EFD_Supplier` receives NetView events which have passed a NetView filter, converts them into CORBA events and forwards them to the registered consumer objects (i.e. CORBA-based management applications) via a CORBA event channel (see figure 4). For technical reasons, we have implemented a helper class `Event_Dispatcher` whose purpose is to handle the registrations of event consumers and to distribute the filtered events to the appropriate consumers. The IDL definition of an `EFD_Supplier` is as follows:

```
interface EFD_Supplier : SOMObject
{
 attribute string filter;
 void set_Dispatch(in Event_Dispatcher disp);
 oneway void activate();
}
```

A consumer can specify what kind of events he wants to receive by setting the `EFD_Suppliers` `filter` attribute. When an `EFD_Supplier` is activated, the `filter` attribute is converted into a NetView event filter and a session with the NetView filter process (*ovesmd*) is opened. Every NetView event that passes the filter is converted to the corresponding CORBA event and is forwarded to the registered consumers via the event channel. The conversion is done by a callback method that is activated whenever an event passes the filter. The purpose of this method is to map a filtered SNMP trap into an appropriate CORBA event according to the specific-ID of the trap. The parameters of the trap are included into the event message and forwarded to the `Event_Dispatcher` who introduces them into the corresponding event channels. As soon as the filter is initiated by the `activate()` method call, the traps are collected and handled by the callback method in an endless loop. In order to prevent that the consumer blocks, too, the activate method is defined as `oneway`. One `EFD_Supplier` object exists for every activated filter.

As the filtering happens completely in the SNMP-based management platform, the filter rules are restricted to the above described (SNMP-focused) criteria. A native CORBAservice that allows filtering according to more powerful criteria is therefore highly desirable. The standardization of an appropriate Notification Service is currently underway (see also section III-D.2).

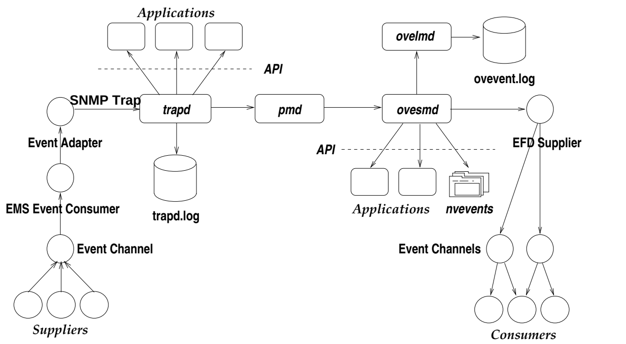

Fig. 4. Using the platform event propagation and filtering APIs

A.3 Handling Internal Errors in a Distributed Object System

The previously described interfaces seamlessly integrate the NetView *Event Management Services* into the CORBA environment. Events can be used to convey various types of information inside a distributed management system. One of the main application areas however is to notify a manager (application or human) when a fault situation has occurred inside a managed resource.

For CORBA-based application and service management, the managed resources are often the objects themselves (as opposed to objects representing physical resources). CORBA objects handle fault situations during method invocations by generating exceptions which are passed back to the caller of the method. In order to use NetView for fault management in a CORBA (here: SOMObjects) environment, we modified the object adapter of the server processes in such a way that it generates fault events whenever an exception has occurred during a method invocation on an object. Please note that this refers only to **internal errors** during the processing of events coming from resources. All method invocations by clients on an object inside a SOMObjects server process are first received by the object adapter that connects the server to the ORB. The object adapter forwards the requests to a server object which is an instance of the class SOMDServer. This server object performs the actual dispatching of the request on the target object. We extended SOMDServer by introducing a subclass called ManagedServer and whose purpose is to check for exceptions after dispatching method calls on target objects. Whenever it detects an exception, it generates a fault event and sends it to NetView. By replacing the regular server object with a ManagedServer, all exceptions inside the corresponding server process can be captured by the management platform because such a server object is present in every SOMObjects server process. Therefore, the platform is also informed if errors occur in the distributed processing environment.

Subclassing from the server object is a convenient way to integrate management functionality into the dispatching of method calls. One can then survey how well the management infrastructure itself performs. Other possible application areas of this mechanism are:

- Ensuring security by checking the principal parameter (user and hostname) of method calls and comparing this data with the access control lists of the server.
- Enabling accounting by deriving the account ID from the principal parameter (user and hostname) of method calls.
- Monitoring the performance of the distributed system by measuring the time it takes to dispatch a method.

B. Topology Management

As outlined in section III-D, topology services manage topological relationships between managed objects. In this section, we describe first how a graphical representation of abstract objects bound to real resources might look like. This is necessary because (see section II) LEO/MEO satellites provide to a large extent telecommunication services (perceived as CORBA objects) that initially do not fit well into an object model for network resources as provided by SNMP-based management platforms. Another challenge comes from the high dynamics of satellite communications where not only the relationships between call connections and satellites may differ, but also the number of satellites bound to ground stations. Thus, new managed objects must be rapidly identified by a discovery task and the high frequency of changes requires regular topology updates.

B.1 Visualization of CORBA Resources

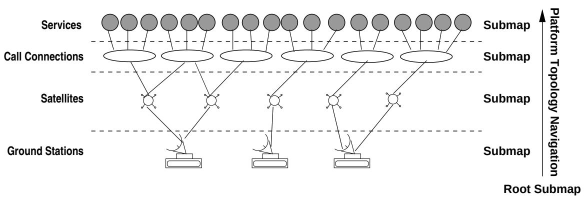

Fig. 5. A simple relationship model for satellite management

In order to visualize and monitor managed resources, a simple abstract relationship model of CORBA systems has been defined and integrated in the database of the management platform. The model is based on the layered architecture depicted in figure 5. The LEO/MEO satellite management scenario described in section II poses several requirements for a relationship model which is suitable for monitoring this CORBA-based environment. However, the topological relationships differ to a certain extent from relationships usually encountered in LAN segments. Therefore, we designed a relationship model suitable for satellite management requirements.

- The highest layer (*services*) consists of distributed objects which may have references to each other and communicate through method invocations. An object offers services to potential clients and may utilize the services of other objects. This is true for telecommunication services implemented as CORBA objects.
- The second layer (*call connections*) comprises the calls in which one or more service objects may participate. A call may be switched either through one satellite to a ground station or (as it is the case e.g. for the Iridium system) over inter-satellite links where several satellites may be involved.
- The third layer (*satellites*) contains the systems over which connections are switched. Although more than one ground station may be visible from a satellite, there is only a n:1 communication relationship between satellites and ground stations.
- The lowest layer shows the *ground stations* which act as the communication backbone for the LEO/MEO satellite constellation.

Although this relationship model is very simple, it is able to handle the dynamic relationships between services, call connections, satellites and ground stations. Please note that the services represent the most fine-grained objects in this model and therefore are placed in the highest part of the hierarchy in our model although one could argue that because call connections depend on services, they should be placed on top of the hierarchy. We believe that the former representation fits better into the topology map navigation mechanism of network management platforms: By selecting a symbol representing a ground station, the platform user reaches a submap that shows all the satellites that maintain actually communication relationships to this one ground station. The subordinate submap shows the call connections that are switched over the satellite in question. If a user clicks on a call connection symbol, he will get the objects that represent the services currently involved in the call.

B.2 The Discovery Task

The data for the above described relationship model is provided by a task that discovers all the CORBA resources and creates entries for them in the management platform database. This discovery task is based on the NetView *Generalized Topology Manager (GTM)* component, which includes an API and a set of abstract data objects.

In order to enable the implementation of a purely CORBA-based discovery application we wrapped the GTM API in a set of CORBA interfaces. From the point of view of a client, the instances of these classes can be treated like any other CORBA object and are used to create entries in the NetView database. To create a model of the CORBA resources, the discovery application provides the following information:

1. The services that are registered with the ORB and their references.
2. The connected calls.
3. The satellites involved in a call.
4. The ground stations over which calls are switched.

The object references are needed to determine the relationships between the services and the connections. Object references are represented by proxy objects inside a `SOMObjects` process. A proxy is an object inside a client's address space that forwards all local method calls to the corresponding target object on the server. In order to determine the server relationships it must be known what proxy objects exist inside a server and on which servers the target objects of the proxies reside. The information about a proxy's target is contained in the stringified object reference corresponding to both the proxy and its target. The calls and the services that run on them are discovered by accessing the CORBA Implementation Repository. A subclass of the `SOMObjects` server object acts as a registry for all the objects and the proxies inside the server. In addition to the already mentioned role of the server object in method dispatching, it also offers functions to create and delete objects inside the server (Like the factory objects from the CORBA lifecycle service). By subclassing from `SOMDServer` and overriding the appropriate methods we achieved that the creation and deletion of objects is registered and that the information about objects and proxies inside the server can be obtained by clients. Other potential information sources with valuable information to a discovery application are naming services for discovering registered objects and trader services for discovering registered services.

B.3 Updating the Topology Data

Since both the number and the state of managed resources dynamically change (e.g. new objects are created, calls are established or dropped) it is necessary to constantly update the topology data that is contained in the management platform database. The traditional approach to this problem is the polling of the network for new resources and for state changes of already known resources by the management platform. The disadvantages of this approach are on the one hand the considerable delay until new resources are identified and, on the other hand, the generation of potential excessive network traffic that reduces the available bandwith for other applications. To avoid these problems we developed an approach that is completely event-driven. Managed resources emit events to well-known event channels when their state changes or when they notice the creation or deletion of other resources. To realize the generation of topology events we modified the following SOMObjects runtime components:

- Every server object sends out events when objects are created or deleted in the corresponding server.
- The object representing the implementation repository generates events when services are registered or deleted.
- The service processes themselves send out events when they are started and when they are terminated.

The benefit of this event-driven approach is that the management related network traffic is greatly reduced because an event is only generated when a change in topology data has really occurred and not proactively by the management platform. This is particularly useful for LEO/MEO satellites where changes occur frequently and where bandwith is limited and therefore expensive.

V. Conclusion and Outlook

The paper has described an approach for the integration of CORBA with existing SNMP-based management platforms and its proof-of-concept through a prototype implementation. Our work was motivated by the increasing demand for accessing CORBA-compliant management agents while CORBA-based management platforms are still hard to find on the market. The need for such an integration has been demonstrated through a concrete scenario, namely the CORBA-based management of satellite constellations.

We have demonstrated how the major integration problems can be solved with rather small effort by

- exchanging events between the CORBA objects and the platform through CORBA typed event channels,
- supplying information to the platform database, and
- using open platform services as a temporary replacement for currently specified, but not yet available CORBAservices.

The overall message is that there are products available on the market place today which allow to implement CORBA management platforms by the thoughtful combination of available components, e.g. CORBAservices and network management platforms. We have used off-the-shelf products and tied them together in order to obtain a CORBA-based integrated system and network management platform. Great care has been taken to base our work on open, standardized mechanisms and services for performing the integration. Especially the CORBA event service and the JIDM specification translation algorithm for mapping the Internet SMI to OMG IDL turned out to be particularly useful and powerful tools for building our interoperability solution.

The acquired solution permits the management of SNMP- and CORBA-compliant agents across the boundaries of the Internet and CORBA management architectures and therefore helps to preserve the large investments in the already deployed SNMP-based management platforms while presenting the opportunity of seamlessly integrating new and more powerful CORBA-compliant management agents. These advantages might encourage large network providers to deploy powerful distributed object systems and represents therefore a step towards integrated enterprise management.

Acknowledgment

The author wishes to thank the members of the Munich Network Management (MNM) Team and for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its webserver is located at `http://wwwmnmteam.informatik.uni-muenchen.de`.

References

[1] H.-G. Hegering, S. Abeck, and B. Neumair, *Integrated Management of Networked Systems — Concepts, Architectures and their Operational Application*, Morgan Kaufmann Publishers, 1998.

[2] Morris S. Sloman, Ed., *Network and Distributed Systems Management*, Addison Wesley, 1994.

[3] "The Common Object Request Broker: Architecture and Specification," OMG Specification Revision 2.2, Object Management Group, Feb. 1998.

[4] "CORBAservices: Common Object Services Specification," OMG Specification, Object Management Group, Nov. 1997.

[5] Barry Miller, "Satellites free the mobile phone," *IEEE Spectrum*, Mar. 1998.

[6] G. Dreo, B. Neumair, R. Wies, A. Böttcher, and M. Werner, "Management von LEO/MEO-Satellitennetzen: Anforderungen und Netzdarstellung," in *Kommunikation in verteilten Systemen, GI/ITG-Fachtagung*, K. Franke, U. Hübner, and W. Kalfa, Eds. Feb. 1995, pp. 270–284, Springer-Verlag.

[7] Andreas Vogel and Keith Duddy, *Java Programming with CORBA*, John Wiley & Sons, Inc., 2nd edition, 1998.

[8] Robert Orfali and Dan Harkey, *Client/Server Programming with Java and CORBA*, John Wiley & Sons, Inc., 2nd edition, 1998.

[9] "Mobile Agent System Interoperability Facilities Specification," Document, Object Management Group, May 1997.

[10] Franck Barillaud, Luca Deri, and Metin Feridun, "Network Management using Internet Technologies," In Lazar et al. [23], pp. 61–70.

[11] Michael Maston, "Using the World Wide Web and Java for Network Service Management," In Lazar et al. [23], pp. 71–84.

[12] Pramod Kalyanasundaram and Adarshpal Sethi, "Interoperability Issues in Heterogeneous Network Management," in *Journal of Network and Systems Management*, Manu Malek, Ed., vol. 2, pp. 169 – 193. Plenum Publishing Corporation, June 1994.

[13] S. Mazumdar, S. Brady, and D. Levine, "Design of Protocol Independent Management Agent to Support SNMP and CMIP Queries," in *Proceedings of the 3rd IFIP/IEEE International Symposium on Integrated Network Management*. Apr. 1993, North-Holland.

[14] Subrata Mazumdar, "Inter-Domain Management between CORBA and SNMP," in *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, L'Aquila, Italy, Oct. 1996.

[15] Nader Soukouti and Ulf Hollberg, "Joint Inter-Domain Management: CORBA, CMIP and SNMP," In Lazar et al. [23], pp. 153–164.

[16] "Inter-Domain Management: Specification Translation," Open Group Preliminary Specification P509, Open Group, Mar. 1997.

[17] JIDM Working Group, "CORBA/TMN Interworking - SNMP Part," OMG TC Document telecom/98-05-03, Object Management Group, May 1998.

[18] A. Keller and B. Neumair, "Using ODP as a Framework for CORBA-based Distributed Applications Management," in *Proceedings of the Joint International Conference on Open Distributed Processing (ICODP) and Distributed Platforms (ICDP)*, J. Rolia, J. Slonim, and J. Botsford, Eds., Toronto, Canada, May 1997, pp. 110–121, Chapman & Hall.

[19] A. Keller, "Tool-based Implementation of a Q-Adapter Function for the seamless Integration of SNMP-managed Devices in TMN," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 98)*, New Orleans, USA, Feb. 1998, IEEE Communications Society, pp. 400–411.

[20] "OSI-Abstract-Data Manipulation API (XOM)," X/Open CAE Specification C180, X/Open Ltd., Nov. 1991.

[21] Tom R. Chatt, Michael A. Curry, Juha Seppä, and Ulf Hollberg, "TMN/C++: An object-oriented API for GDMO, CMIS, and ASN.1," In Lazar et al. [23], pp. 177–191.

[22] IBM Corporation, Research Triangle Park, NC 27709-12195, *IBM NetView for AIX Version 4: Programmer's Reference*, 1st edition, July 1995, Order Number: SC31-8165-00.

[23] Aurel A. Lazar, Roberto Saracco, and Rolf Stadler, Eds., *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, CA, USA, May 1997. IFIP, Chapman and Hall.