Issues and experiences of CORBA-based management agents

D. Ranc
Institut National des Telecommunications
Rue Charles Fourier, 91011 Evry Cedex, France
daniel.ranc@int-evry.fr

G. Pavlou
University of Surrey,
Guildford, Surrey GU2 5XH, UK
G.Pavlou@ee.surrey.ac.uk

D. Griffin
University College London
Torrington Place, London WC1E 7JE, UK
D.Griffin@ee.ucl.ac.uk

J. de la Horra
Polytechnical Center Saragossa
Spain
Juanza99@worldonline.es

Abstract

This paper addresses a number of issues related to the evolution of the TMN framework in terms of the associated interface technology. A review of the manager-agent model is followed by an analysis of the challenges raised by emerging distributed object architectures which may result in highly distributed management systems; from these, fundamental requirements for CORBA-based TMN agent architectures are derived. The paper describes two approaches for realising CORBA-based management agents that attempt to fulfil these requirements. Both approaches have been validated through implementation while operational experiences with the two approaches are presented.

Keywords

CORBA-based TMN, management agents, JIDM.

1. Introduction

Over the years, operators and vendors of telecommunication networks have progressively standardised, developed and deployed a comprehensive set of network management tasks for telecommunication networks and associated equipment. The ITU-T has produced the sophisticated Telecommunication Management Network (TMN) [1] architecture which is based on the OSI Systems Management (OSI-SM) framework [2], while vendors exhibited a variety of proprietary attempts. The current landscape of telecommunications management consists of a solid base of TMN-based systems which are, however, increasingly challenged by new distributed object architectures, e.g. the Common Object Request Broker Architecture (CORBA) [3], and associated frameworks for integrated management and control, such as the Telecommunication Information Networking Architecture (TINA) [4]. This paper tries to identify how these new architectures may be used in the context of TMN but

also considers their limitations with respect to the performance and reliability requirements of large-scale management systems.

Over the second half of the 90's a substantial amount of effort has been invested in identifying how distributed object technologies will be used in telecommunications management. In particular, the NMF-X/Open[1] Joint Inter-Domain Management (JIDM) task force has produced significant results [5][6] which were input to OMG's Telecommunications Special Interest Group (TelSIG). In the minds of the authors, the JIDM approach, however important it may be, is subject to debate regarding key TMN requirements such as scalability and other architectural difficulties that motivate the need for further refinement [7]. This paper presents two approaches for building TMN agent applications which use parts of the JIDM results: one choosing a native CORBA-based approach, but respecting TMN requirements and standards, and another one that migrates TMN software platform infrastructures to a CORBA-based environment.

While the manager-agent model of the TMN has been mainly conceived for network management, the situation is different in the service management layer, in which the amount of management information is not of the same scale as in network elements. In this environment, there is arguably no need for agent-like entities that provide optimised access to managed objects but generic distributed object services such as trading and naming may be used instead. The last part of the paper examines the possibility to use such generic services for object location and access and discusses the briefly the issues and potential problems with this approach when last amounts of highly distributed objects need to be accessed.

The rest of the paper has the following structure. In section 2, we present an overview of existing and emerging telecommunications management architectures, including the manager-agent paradigm, CORBA in the context of the TMN, TINA and the JIDM work. In section 3, we identify key requirements for a credible telecommunications management agent architecture. In section 4, we present an architecture and design for a CORBA-based agent which relies on existing TMN software platform infrastructure. In section 5 we present a different architecture and design for a native CORBA-based agent. In both cases, we present our operational experiences with the respective agents and we comment on the extent to which they meet the requirements identified before. Finally, in section 6 we present our conclusions.

2. Existing and Emerging Telecommunication Management Architectures

2.1. The TMN and the Manager-Agent Paradigm

The manager-agent paradigm is a fundamental part of traditional network management systems, appearing in both the Simple Network Management Protocol (SNMP) and OSI Systems Management (OSI-SM) [2] approaches. The TMN

1 Now TMF and Open Group respectively.

(Telecommunication Management Network) [1] architecture defined by ITU-T uses the manager-agent entity as a fundamental building block of large-scale, hierarchical TMN systems.

In the TMN, Operations Systems (OSs) are dual manager-agent while Network Elements (NEs) are agent entities which use an OSI 7 layer stack and CMIS/P [8] as the communication service and protocol. In this scheme, the manager issues CMIS/P requests via the OSI stack, which are processed by the agent, with the results being passed back later in an asynchronous fashion. The agent and manager are *roles*, i.e. this notion is dynamic in time, with OSs taking either of the two roles in different instances of communication.

The information presented to the manager by the agent is standardised and consists of publicly available Information Models. The latter are specified using the Guidelines for the Definition of Managed Objects (GDMO) [9] formalism; which is object-oriented and consists of classes, attributes and methods (actions and notifications). Values of GDMO attributes, action and notification parameters are represented using the Abstract Syntax Notation One (ASN.1) [10] structuring language and particular coding rules in order to be transmitted by the OSI stack.

An agent entity is composed of three fundamental elements: the *Managed Information Base* (MIB), the collection of *implementation objects*, and the *resources*. The MIB consists schematically of the *naming tree* and the *Managed Objects (MOs)*. The naming tree is the access structure for the Managed Objects via a standardised hierarchical naming scheme. The managed objects, specified in GDMO, are the entities subject to manipulation by the manager through CMIS/P operations.

The scenario depicted above is, however, not cast in stone. New, powerful distribution infrastructures and telecommunication architectures are emerging. The following two sections summarise the two emerging frameworks in this area.

2.2. CORBA from a TMN Point of View

The Object Management Group (OMG) has specified the Object Management Architecture (OMA) and is now defining higher level business-oriented services and entities, making the overall architecture more and more complete [3].

The distribution architecture of CORBA is built upon a software bus, the Object Request Broker (ORB), which provides the infrastructure allowing distributed software components to communicate. Key aspects are: ubiquitous use of the client-server paradigm, true object orientation, implementation language independence, and implicitly distant objects i.e. no difference between local and remote objects and access transparency.

The latter property is particularly powerful for TMN systems designers. Within the OSI communication framework, the programmer has (even with sophisticated software layers between the local representation of the Managed Object, and the distant one) to deal explicitly with communication aspects; whereas in a CORBA-based environment, the programmer does not even distinguish (after some simple initialisation) between local and remote objects: the remote object appears, and is manipulated, in the same manner as a local one in the programming language.

Additionally, data representation burdens, which are taken care by the ASN.1 standard in the ITU-T world, are dealt with in a completely transparent way by the ORB.

An additional benefit of CORBA, compared to OSI Systems Management, is its ease of use. An OSI programmer is faced with relatively expensive tools and complex software Application Programming Interfaces (APIs) such as the low-level XOM/XMP[2] or the more high-level TMN/C++, whereas CORBA systems are comparatively cheap and easy to learn. Typically, competent C++ programmers are able to build simple client-server systems in half a day under CORBA; this is, unfortunately, not the case with OSI-SM technology. These shifts in distribution technology are the main motivations that make CORBA attractive environment when projecting new TMN systems.

This decision-making point however hides some key questions regarding the TMN architecture. CORBA does not have the powerful access aspects of CMIS/P such as scoping/filtering, neither does it provide a suitable architecture for credible telecommunication management e.g. thinking of building blocks like the Event Forwarding Discriminator, an essential entity for scalable, fine-grain event dissemination in telecommunications systems, and the Systems Management Functions (SMFs) which support generic Fault, Configuration, Accounting, Performance and Security (FCAPS) functionality.

In summary, the TMN system designer is very much tempted by the comfort of the CORBA architecture, but has no real valid support regarding the management of telecommunications systems. The temptation may be so strong, however, that some had-hoc implementations may arise and solve particular local network management problems but at the cost of ITU-T standards respect and, as a consequence, at the cost of the hope for general interoperability between all stakeholders of the telecommunications service architecture.

2.3. The TINA Architecture

The Telecommunication Information Networking Architecture Consortium (TINA-C) [4] aims at providing an advanced object-oriented software architecture for integrated telecommunication network and service management and control. In summary, the TINA architecture consists of two major building blocks:

- the *Service Architecture*, which represents a step forward beyond the specifications delivered by the ITU-T, which are limited to the network and element management, and which addresses service control. The service architecture introduces the concepts of access and service session and integrates service control with service management.
- The *Network Resource Architecture*, which is the TINA view of TMN network and element management. This uses the Network Resource Information Model (NRIM) to model connection-oriented networks in a technology-independent fashion.

2 The XOM/XMP API has been specified by the X/Open consortium, now known as the Open Group.

The distribution infrastructure proposed by TINA is the Distributed Processing Environment (DPE), which is based on CORBA but is enhanced with telecommunication-oriented features. The DPE runs on an overlay network that is used for control and management, known as the Kernel Transport Network (kTN). TINA makes an extensive use of Open Distributed Processing (ODP) [11] methodology in its specifications.

A discussion on the architectural relationship and potential migration of the TMN to the TINA architecture can be found in [12], which was written with the idea that TINA will eventually replace today's Intelligent Network (IN) and TMN functionality. However, it is no longer clear whether TINA will be fully applied in a unifying telecommunications software architecture. More realistically, a selected set of TINA concepts and components are expected to be found in future telecommunications systems.

2.4. JIDM Agents

A number of studies and projects have considered CORBA as distribution platform for network management systems. In this section we present a brief summary of the major one. The Joint Inter-Domain Management (JIDM) [5][6] study group of the OMG has worked on both a mapping scheme of GDMO/ASN.1 to IDL, and on an interaction translation for mapping an OSI-SM agent environment to CORBA.

- *GDMO/ASN.1 to IDL mapping*. This study group has specified a complete GDMO/ASN.1 to IDL mapping, proposing a number of rules enabling to build a complete compiler for automatic IDL generation from GDMO specifications.
- *JIDM agent*. An architecture for a management agent was specified, the features of which could have made it an interesting candidate for a TMN agent. This architecture defines managed objects as individual CORBA objects e.g. servers possessing individual IDL interfaces that are generated by the GDMO to IDL mapping rules. The relations between instances, containment for instance, are modelled using CORBA references.

Two remarks emerge from a closer examination of these specifications. The first one concerns the mapping of ASN.1 to IDL. Recalling that the function of ASN.1 is to transport attribute, action and notification values between agents and managers in a platform-independent fashion, and that this function is taken care transparently by CORBA's Object Request Broker (by its internal marshalling/unmarshalling functions), one may question why to duplicate this function a second time by translating the ASN.1 structures to IDL. Moreover, this encoding/decoding is left to the managed object itself, because it holds the corresponding IDL, whereas this function was largely taken care by platform services in existing TMN frameworks.

A second remark concerns the mapping of managed objects into CORBA servers. This feature is an attempt to use the CORBA paradigm as deeply as possible, however the consequences are questionable. The granularity of a CORBA server is, in the eyes of the authors, the size of a system component (a distributed system holding, for example, the order of 10 such components) – whereas the granularity of a managed object instance is the size of, for instance, a C++ object instance. This

difference results in a granularity mismatch. Moreover, in the JIDM scheme the managed objects are not any more structured in a MIB – they are all visible at the same level (even if the references emulate a virtual naming tree). This may become hazardous in realistic TMN situations with, typically, 1^E6 managed object instances for an agent, because existing ORBs are not designed to manage such a large number of server objects.

These two remarks motivated us to consider the possibility of further enhancements and to propose the solutions presented in this paper.

3. Some requirements for a credible telecommunications agent

With the previous considerations in mind, a credible telecommunications agent should exhibit a number of characteristics, including:

- *Scalability.* Network equipment, especially public Wide Area Network (WAN) equipment, such as Synchronous Digital Hierarchy (SDH) add-and-drop multiplexers or large scale Asynchronous Transfer Mode (ATM) switches, become more and more complex. This is reflected both in the complexity of the management software at the desk of the operator, and at the level of the agent. The latter results in very large MIBs where a magnitude of 1^E6 managed objects is typical. Up to now, only the sophistication of the TMN architecture through the manager-agent model was able to deal with such vast quantities of data.
- *Dynamicity.* This equipment offers a large number of dynamic features which were absent of previous technologies. These features allow operators to offer new, more dynamic telecommunication services. On demand establishment of connectivity for e.g. multimedia applications is a typical operation that requires, at the agent level, a large number of interactions to insure the configuration of the Virtual Channels, their capacity and Quality of Service (QoS), as well as adequate cross-connections. These interactions have to be fast enough to ensure near real-time access to the required telecommunication service.
- *Interoperability.* The telecommunications market is becoming more heterogeneous. The traditional operators are challenged with aggressive competition. Actors such as major software vendors (Microsoft) or content providers (Bertelsmann, Disney) are more and more involved in the telecommunication business. As a contemporary example of the level of interactions between telecommunication companies, a simple telephone call from e.g. a fixed line in Paris to a mobile phone in London requires interactions between at least 4 companies: the French traditional operator (who owns the access part of the line), the Eurotunnel operator (owning the optical fiber in the tunnel), an arbitrary English operator up to London, and a mobile operator in London itself. Future multimedia services such as video on demand will require interactions between the different participants that together, deliver the service: the network operators (for the connectivity), the content provider at least. The requirements for interoperability are therefore becoming even stronger.

- *Standard compliance.* This requirement is related to the interoperability requirement and concerns both the computational viewpoint and the information viewpoint of the system. The computational viewpoint has already been discussed earlier, and refers to the powerful aspects of CMIS, the naming scheme of the managed objects etc. The information viewpoint refers to the important, and valuable, capital of existing GDMO information model specifications both at the element and network management layers. It is reasonable to assume that GDMO specifications will continue to be the ultimate source of the information in the TMN, even if the GDMO specifications are turned into other forms of interface specifications by translation tools. In the following sections, we present two approaches regarding the design and implementation of CORBA-based TMN agent environments. The first one uses already existing TMN platform infrastructure, parts of which are re-used to result in a CORBA-based agent. The second one is designed from the beginning based on a CORBA environment, without reusing any previous infrastructure. Both approaches try to address the requirements for scalability, dynamicity / performance and interoperability that were identified before. Another commonality of both projects is to remain faithful to the CMIS paradigm, even in a CORBA-based distribution scheme. Both teams indeed still believe in the strength of this protocol, and in the value of the associated GDMO-based information models.

4. A TMN platform CORBA-based Agent

4.1. Architectural and Implementation Considerations

A JIDM-based approach for mapping the OSI-SM model to CORBA has already been presented in [7]. One key aspect of this approach is that it tries to stay as closely as possible to the OSI-SM approach, minimising the use of CORBA services [13] and allowing reusability of existing modular OSI-SM platform infrastructures. In this paper we present an implementation approach which reuses parts of the OSIMIS platform [14] and allows a phased transition strategy that will ease compatibility and interoperability with existing TMN systems.

The first step for migrating towards the target framework is to support only agent discovery and CMIS interactions through CORBA, without individual IDL interfaces for managed objects. This essentially means that a Management Broker (MB) object with a CMIS-like interface will act as an agent that provides access to managed objects which are implemented by existing TMN platform infrastructure i.e. GDMO/ASN.1 compilers and relevant APIs. The MB may be used in conjunction to the existing Q_3 agent object within an agent application. In this case, the TMN application in agent role will have two interfaces: the existing Q_3 interface and the CORBA version of the CORBA-based "Q_3" interface. The latter may be according to JIDM pseudo-CMIS specification. The current implementation uses a different pseudo-CMIS which was designed in parallel with the JIDM one and stays as close as possible to the ITU-T CMIS [8]. In this approach, notifications are disseminated

through OSI-SM Event Forwarding Discriminators (EFDs) and a "push" model, supported by a i_CMISManager interface.

This minimal approach is depicted in Figure 1 and has no impact at all at the implementation of managed objects which are based on TMN platform technology e.g. OSIMIS [14]. Existing OSI-based manager applications will continue to function while new CORBA-based management applications may be developed. CORBA manager objects get access to the MB interface reference through the CORBA naming services [13] while OSI manager objects get access to the address of the OSI agent through the OSI directory [15]. It should be noted that two different notations have been used in this figure to depict interactions, one for CORBA using object interfaces and one for OSI-SM using arrows.

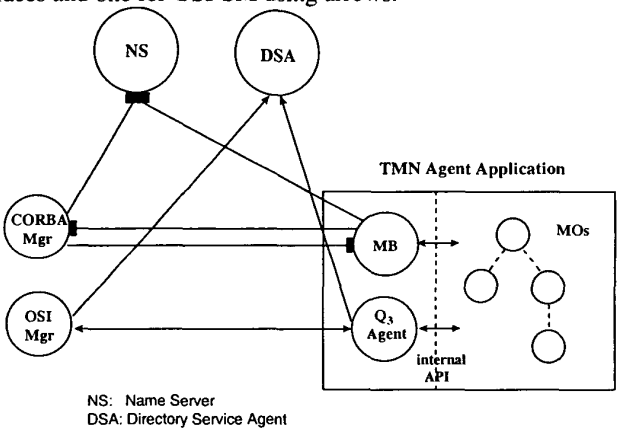

Figure 1: Dual Q3 and CORBA based Agent

This architecture exploits the fact that the object models are the same in the two frameworks, and provides a "dual-agent" access paradigm. In this framework, the managed object GDMO/ASN.1 specifications are translated to IDL but the resulting IDL interface specifications are not instantiated: they simply "document" the management broker interface which provides access to those objects in a dynamic, weakly-typed fashion. The attribute, action and event ASN.1 types are used across the management broker interface through the equivalent IDL types.

Implementing this framework was fairly straightforward. The only difficulty encountered was the bi-directional translation between ASN.1 and IDL data types. This can be automated if one has access to a flexible ASN.1 compiler, which could be customised to produce the equivalent IDL types as well as the conversion routines in both directions. Since OSIMIS uses the ISODE *pepsy* compiler which cannot be easily customised, these conversions had to be hand-coded. Implementing the tightly-coupled dual agent version was also straightforward given the well-defined OSIMIS APIs for accessing managed objects within an agent application.

The complete CORBA-based framework requires also that individual managed objects become computational constructs with IDL interfaces. The advantage in this

case is strongly-typed access, superior distribution aspects and potentially better performance. Having produced the relevant specifications using GDMO to IDL translation, the key issue is how to support CMIS features such as multiple attribute get/set and filtering, which are very important for performance and scalability. This functionality can also benefit from existing OSI-SM platform infrastructure and should be supported by a i_ManagedObject interface from which the OSI-SM i_top interface derives.

Classes for derived interfaces should pass to their i_ManagedObject parent class the names and types of their attributes when an object instance is created. The i_ManagedObject part will subsequently access the attributes of derived parts as if they belonged to separate objects, through the CORBA Dynamic Invocation Interface (DII). This scheme supports the multiple attribute get and set methods. The only way to deal with filtering is to provide the IDL compare methods by hand, in a hash table indexed by the relevant type. The i_ManagedObject part will retrieve the attributes involved in the filter and will invoke the relevant compare and traverse methods, getting access to them through the attribute type which is known from its "repository". While this is feasible, it is far from desirable since it requires hand-written routines for every attribute type. On the other hand this is the only approach to support filtering until OMG considers the inclusion of relevant features in IDL and their concrete support through the programming language mappings.

4.2. Performance and Scalability

We conducted a number of experiments on a pair of Sun Sparc20 workstations running Solaris and being connected to a lightly-loaded Ethernet. The Q3 protocol stack used operated over TCP/IP using the RFC1006 method while the CORBA protocol used was the Internet Interoperability Protocol (IIOP). The OSIMIS research prototype was used for the Q3 interface [14] as opposed to a commercial CORBA implementation.

While a detailed performance comparison could be the subject of a separate paper, the key conclusion was that the CORBA-based access times for method invocations through the management broker were faster by about 30% on average. Performing the same operation to the managed object through a direct IDL interface resulted in 45% faster times. In addition, the initial bind operation through IIOP was about 60% faster than association establishment using the Q3 protocol stack. The use of the CORBA IIOP protocol in this case seems to be a more lightweight solution compared to the relatively heavyweight Q3 protocol stack, especially for managed objects with IDL interfaces.

The next experiment concerned the size of packets exchanged, which were measured at the TCP payload level using the Berkeley *tcpdump* program. Performing an echo operation of 1 byte to an object in the first level of the MIT incurred 72/88 byte packets for the request/response in the case of Q3, 177/47 bytes in the case of the management broker and 101/32 bytes in the case of a direct IDL interface. Performing an operation asynchronously through the management broker, which means that an Interoperable Object Reference (IOR) is exchanged, resulted in 409/97

byte packets. After many different measurements with different argument types, we came to the conclusion that IIOP generates a relatively large amount of traffic in comparison to Q3, especially when IORs and IDL *any* types are exchanged, since the latter convey CORBA type-code information.

The final experiment concerned the memory required for managed objects in both TMN and CORBA platforms. In the case of CORBA, the ORB which is typically running as a separate process is required amounts to 2.7 Mb at runtime. The size of a process containing a single server object and an associated factory object was about 2.6 Mb. The size of the equivalent OSI agent was 2.3 Mb but the latter contains also functionality of name resolution, scoping, filtering event reporting and logging. The key issue though is not the size of the infrastructure which is incurred once but the data size of managed objects at runtime. After various experiments and memory size measurements, it became clear that the data size of a CORBA object is about 2-3 times more than the size of a GDMO object, depending on the data type of the attributes and the relevant values. In addition, the particular ORB used seemed to have a problem to cope with very large amounts of managed objects.

These experiments confirmed the fact that IIOP is expected to perform better than Q3 interfaces, generating though larger amounts of management traffic. On the other hand, the experiments also confirmed the fact that the required memory size is bigger for native CORBA managed objects, which may result to scalability problems. This reinforces the management broker approach presented in this paper as an approach that benefits from the better performance of CORBA without the associated potential scalability problems. The native CORBA-based managed object approach could be the next step, with more mature, second or third generation ORB products coming to the market place.

5. A Native CORBA-based agent

5.1. Architectural motivations

Given the different constraints and requirements to build a realistically deployable TMN agent based on CORBA, a number of considerations and options come to the mind of the system designer. In this particular project, several options were taken as main architectural options:

- Referring to the scalability requirement, it has been chosen to design the agent such as to hide the managed objects inside the agent as the default option. This way, it is possible to accumulate managed objects by the millions without dealing with ORB limitations. An option to publish, on request, the CORBA reference of a particular sub-tree of the MIB is kept open (e.g. if repetitive access to the physical view of an equipment is foreseen, the manager could be interested in such a shortcut through the naming tree).
- The external access to the agent conforms to the JIDM pseudo-CMIS specification. This has the double advantage of a standardised publicly available IDL specification, and of a total independence from the information model used

by the agent. Full implementation of all CMIS features, such as scoping and filtering, were put at a top priority.

- Regarding the ASN.1 architecture mismatch referred in section 3, a heavy decision has been made: to discard the existing ASN.1 definitions and to replace them by native IDL types by hand. This has the drawback of deriving from strict standard handling, at the benefit of a considerable simplification of the designer's life. A close examination of ASN.1 types delivered by standards indeed leaves an impression of pointless complexity (as an example, all ASN.1 `graphicString`, `printableString` etc. types, many in the standards, are thus replaced with the unique IDL `string` type). The cost of hand-typing the types has been tested on the existing implementation, and has been felt as quite acceptable.

5.2. Realisation

The architecture of the agent has been designed as a GDMO template handling engine at the heart of which is the naming tree. The latter is implemented using a son-brother scheme holding the managed objet skeletons, which in turn refer to their packages, and finally to attributes and even later to values. The complexity of the system relies on list and memory management, thus quite classical algorithms could be used.

A special note has to be made on multiple result handling. In the CMIS/P context, the agent answers to requests by replies, as many as the result requires (quite many, for a wide scoped request for example). This scheme had to be emulated in the CORBA infrastructure, because the team did not project to depend on a notification service. The actual workaround has been to encapsulate multiple results in a IDL `sequence`, where each element represents one result.

5.3. Implementation notes

The implementation of the agent was based on a quite standard environment: a Sun workstation running Solaris 2.5.1, the Orbix CORBA system and the Sun C++ compiler.

The large use of dynamic memory allocation mechanisms and pointer management has to be emphasised. As an example, the instantiation of a managed object through a M_CREATE request determines the allocation of dynamic memory for each and every entity of the instance (the instance itself, its attributes, their values). Reciprocally, the deletion of an instance requires to free all the memory allocated previously. All the MIB management is implemented using pointers which model the relations between all instance entities. An efficient tree management structure using a son-brother machine representation has been used. This scheme links only one descendant node to its parent node, all other nodes of the same level building a linear list with the former one. The following figure shows partially how these entities link together.

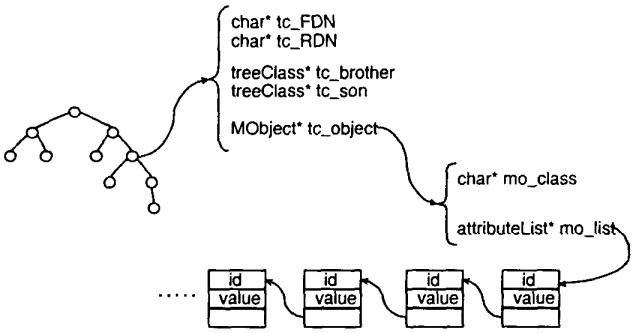

Figure 2: Management Information Tree Implementation

run pseudo-CMIS requests under CORBA. This interface is unique e.g. independent of the Information Model. All functions are confirmed e.g. returning results. The IDL definition followed the JIDM standard with the notable exception of the ASN.1 structures, which have been considerably simplified.

The agent has been run with a minimal manager reading requests from ASCII files. This scheme allowed to test the agent with very large request sets. A hierarchical scheme were the cities contain districts, the latter containing zones, and zones containing equipment, has been used. This network is variously meshed depending on the importance of nodes and their geographical situation.

The information model used for this experiment has been the TINA NRIM. The main class used to model networks is the `subNetwork` class which is subject to a recursive containment relationship e.g. `subNetwork` instances of layer (n) contain `subNetwork` instances of layer (n-1). Other classes used from NRIM include `link`, `topologicalLink`, `linkTerminationPoint` and `topological-LinkTermination-Point` (see figure below).

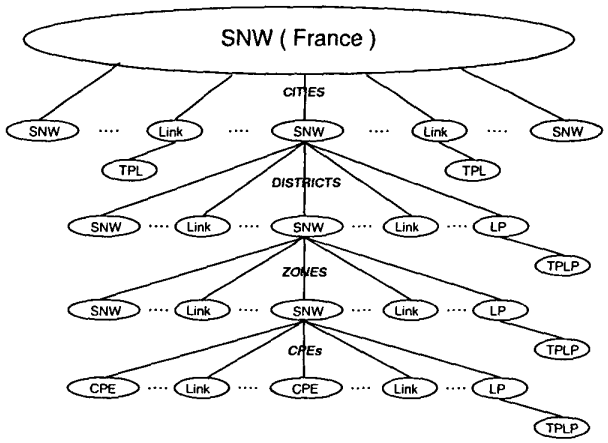

Figure 3: The NRIM-based Simulated National Backbone

Scalability tests included a large scale simulation of a french backbone including instantiation of 430,000 objects, demonstrating the scalability of the approach. Performance results were rewarding. One M_CREATE request including manager to agent and return paths through the available laboratory IP network takes less than 30 milliseconds. The agent has been extensively tested and its stability has been remarkable, both in terms of constant response time and of resilience.

5.4. Future developments

This architecture is expected to evolve in three areas:

OMG component. The OMG defined recently a new architecture for servers called components, which really are a mapping of JavaBeans features onto the CORBA architecture. Powerful items, such as introspection e.g. the ability of the CORBA object to possess a knowledge about itself, and to communicate it to other components, are very attractive to implement within a TMN agent, in the perspective of intelligent, open TMN software systems.

Scalability. One of the performance questions in the system is due to the depth of the naming tree. The longer the trail within the naming tree to access the object, the lower the performance. One defence regarding this aspect is a hash-code based access scheme. Another is the possibility for the agent, given a certain maximum size, to instantiate a new sub-agent holding a certain sub-tree of its parent. Requests to this part of the naming tree would then be subcontracted to this sub-agent by the original agent. Sub-agent instantiation would make profit of CPU and memory statistics to adequately distribute the load over the network. This scheme remains transparent to the manager, since the requests are still routed to the original agent before they are sub-contracted.

Software engineering. In order to enable quasi-industrial handling of this software, a suitable software engineering environment is needed. In particular, a GDMO compiler suite able to generate C++ classes for the agent is concretely planned as a next step.

6. Conclusions

The work presented in this paper addresses the complex subject of using distributed object technologies for telecommunications network management, moving away from protocols and trying to benefit from the ubiquity, usability and ease of ease of distributed object platforms such as CORBA. The key requirements are scalability and performance, since management functions may be used to replace some traditional control functions e.g. connection establishment in TINA. The work carried out needs to be verified further in this respect in terms of feasibility and viability in the context of large-scale telecommunications networks.

Deployment of distributed object technologies will be accelerated in the coming years but we observe that there is a huge demand from operators for such technologies while at the same time there is a low confidence in them. One of the reasons is that these technologies have originated in the computing world while

telecommunications environments pose different requirements of robustness, availability, scalability, integrity and performance. It is only when telecommunications requirements are taken into account in the context of emerging architectures that viable solutions will be provided. The manager-agent paradigm is of paramount importance for the TMN and, as we have shown in this paper, it can be realised in the context of CORBA while maintaining key characteristics of scalability and performance.

References

[1] ITU-T Rec. M.3010, Principles for a Telecommunications Management Network (TMN), Study Group IV,1996

[2] ITU-T Rec. X.701, Information Technology - Open Systems Interconnection, Systems Management Overview, 1992

[3] Object Management Group, The Common Object Request Broker: Architecture and Specification (CORBA), Version 2.0, 1995

[4] M. Chapman, F. Dupuy, G. Nilsson, An Overview of the Telecommunications Information Networking Architecture, in Proc. TINA'95, Melbourne, 1995

[5] NMF - X/Open, Joint Inter-Domain Management (JIDM) Specifications, Specification Translation of SNMP SMI to CORBA IDL, GDMO/ASN.1 to CORBA IDL and IDL to GDMO/ASN.1, 1995

[6] T. Rutt, ed., Comparison of the OSI Systems Management, OMG and Internet Management Object Models, Report of the NMF - X/Open Joint Inter-Domain Management task force, 1994

[7] G. Pavlou, A Novel Approach for Mapping the OSI Systems Management TMN Model to ODP / OMG CORBA, in Integrated Network Management VI, Proc. IM'99, Boston, pp. 67-82, IEEE, May 1999

[8] ITU-T Rec. X.710, Information Technology - OSI, Common Management Information Service Definition and Protocol Specification, Version 2, 1991

[9] ITU-T Rec. X.722, Information Technology - OSI, Structure of Management Information - Guidelines for the Definition of Managed Objects, 1992

[10] ITU-T Rec. X.208, Information Technology - OSI, Specification of Abstract Syntax Notation One (ASN.1), 1988

[11] ITU-T Draft Rec. X.901-904, Information Technology - Open Distributed Processing, Basic Reference Model of Open Distributed Processing, 1995

[12] D. Ranc, G. Pavlou, D. Griffin, A Staged Approach for TMN to TINA Migration, in Proc. TINA'97, pp. 221-228, IEEE Comp. Soc., Santiago, 1997

[13] Object Management Group, CORBA Services: Common Object Services Specification (COSS), Revised Edition, 1995

[14] G. Pavlou, G. Knight, K. McCarthy, S. Bhatti, The OSIMIS Platform: Making OSI Management Simple, in Integrated Network Management IV, A. Sethi, Y. Raynaud, F. Faure-Vincent, ed., pp. 480-493, Chapman & Hall, 1995

[15] ITU-T Rec. X.500, Information Technology - OSI, The Directory: Overview of Concepts, Models and Service, 1993