An Integrated and Federative Approach to QoS Management in IP Networks

Daniel Ranc, Jacques Landru, Anas Kabbaj

Institut National des Télécommunications, Rue Charles Fourier, F-91011 Evry Cedex
Daniel.Ranc@int-evry.fr
ENIC, cité scientifique, rue G. Marconi, F-59658 Villeneuve d'Ascq
Jacques.landru@enic.fr

Abstract. This paper focuses on challenges related to the management of Quality of Services in IP networks and proposes a solution relying on a CORBA-based Service Architecture tightly and dynamically linked to the managed network. This architecture relies on Management Layers: EML, NML and SML inherited from the TMN architecture in order to deliver a more deterministic behaviour of the IP network. It makes also use of concepts defined by the TINA architecture as well as the TMF regarding key SML aspects. EML management is taken in charge by in-depth use of JDMK agents hiding network heterogeneity.

1 Introduction

The large increase of IP networks penetration makes it progressively replace legacy technologies such as X.25 and proprietary technologies such as IPX, DSA or SNA. On the other hand the deployment of ADSL and radio at the local loop will introduce new requirements for Services Management, particularly for new types of high bandwidth-related services such as Video on Demand, Multiconferencing or Teleteaching. This shift however brings new requirements at the network level regarding isochronous signal transmission in order to realistically carry e.g. voice data. Other services related to Network Management formerly using deterministic X.25 technology are replaced with IP services exhibiting a much more probabilistic behaviour, even if the overall mean efficiency of the latter is high.

This situation may be taken in charge by two approaches: a first network-oriented approach where the equipments themselves would be in charge of Quality Management – the DIFF-SERV, MPLS or tag switching techniques would represent this approach; a second approach where a Management System would rule the use of network resources and organise the dispatching of services to users – the TINA architecture has been an attempt to do so.

The opinion of the authors is that both approaches are unavoidable: a resilient, QoS-based IP network matching the requirements of a Service Management Layer that delivers high quality on-demand services in near-real time to users.

S. Rao and K.I. Sletta (Eds.): INTERWORKING 2000, LNCS 1938, pp. 196-208, 2000.

The proposed system which is developed by the GET[1] within the GESTICA [GEST] project is an attempt to demonstrate the usefulness of this combined integrated approach. This paper will therefore first briefly recall fundamentals of current service management architectures. The core part will develop the proposed architecture which combines a layered CORBA-based TMN architecture with Service Management components interpreted from the TINA architecture. Finally a last section will provide the author's conclusions.

2 Current Network Management Architectures

2.1 The Telecommunication Management Network Framework

The TMN (Telecommunication Management Network) [8.] architecture defined by ITU-T sets the framework for large-scale, hierarchically layered network management systems.

Four layers are defined apart from the equipments: the Element Management layer, where one single equipment is managed at a time; the Network Management Layer, where the whole network is considered; the Service Management Layer which takes in charge network services; the Business Management Layer, realising network-related strategical tasks. The figure 1 summarizes this classical layered scheme.

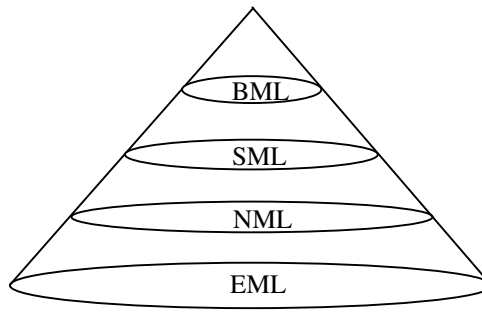

Fig. 1. TMN layered architecture.

Another key feature of the TMN is its dual buildingblock: the manager-agent couple. In this scheme a managing entity, the manager, issues requests to a managed entity: the agent, which replies asynchronously by issuing notifications. 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 [12.] 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

[1] Groupement des Ecoles de Télécommunication

asynchronous fashion. The agent and manager are *roles*, e.g. 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 standardized and consists of publicly available Information Models. The latter are specified using the Guidelines for the Definition of Managed Objects (GDMO) [13.] 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) [X208] 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 seriously challenging the TMN. The following two sections summarise the two main 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 [2.].

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. The remote object is in an implicit distance situation. 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 slightly more sophisticated TMN/C++ API defined by the former NMF, 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 an attractive environment when projecting new TMN systems.

This decision-making point however hides some key questions regarding the TMN architecture. CORBA does not provide 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 fundamental Systems Management Functions (SMFs) which support generic Fault, Configuration, Accounting, Performance and Security (FCAPS) functionality, and which are not covered suitably (in a TMN requirement perspective) by COSS [3.] services.

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, which is a strategic issue.

2.3 The TINA Architecture

The Telecommunication Information Networking Architecture Consortium (TINA-C) [1] aimed 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 sessions 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.

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 the Open Distributed Processing (ODP) [14.] methodology in its specifications.

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

A discussion on the architectural relationship and potential migration of the TMN to the TINA architecture can be found in [9.], 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 The JDMK Framework

In the belief that new management architectures should be more decentralized, more adaptive and more dynamic, system designers created a particular flavour of JavaBean for network management purposes. Such objects can be used to develop modular open network management agents [7.]. Sun Microsystems has created specific management components called *managed beans* (MBean) [4.] which are Java objects featuring management capacities (get, set internal values or event notification) in addition to the properties of normal JavaBeans (modularity, dynamic loading, …).

An MBean instance is manageable as soon as it is registered within a framework..Associated with this framework several protocol adaptors can be used. MBeans can then be accessed in an open manner through several protocols (SNMP, CORBA, RMI, HTTP, …). The MBean concept is the core technology to build modular open multi protocol manager/agent components.

Furthermore, the Java Dynamic Management Kit [4.] is a set of tools enabling the development of Java-based managers or agents as well as a number of useful classes based on the MBean concepts and delivering a high-level API for communication.

Although property of a specific vendor, Sun Microsystems Inc., this framework will probably gain a rising attention and its status is likely to evolve in the future. Sun Microsystems has started the standardization process to produce a universal open management extension of the Java programming language : the Java Management eXtensions JMX [6.]. These extensions can represent the transition from current management technologies (static agents and protocols) to an open model where:

- management agents are plugged in dynamically and immediately available
- management applications are freed from dependancies on a fixed information model or specific communication protocols,
- new management services can be fetched through the network and plugged dynamically into the manager.

The JMX architecture is divided into three levels:

- the instrumentation level: implements the Mbean, the Java object that represents a manageable resource;
- the agent level: provides management agents, the containers that provide core management services which can be dynamically extended by adding new resources. An agent is composed of a Mbean server, a set of Mbeans representing managed resources, and at least one protocol adaptor. An agent may also contain management services, also implemented as a Mbean.
- The manager level: implements management components that can operate as a manager or an agent for distribution and consolidation of management services. It

provides an interface for management application to interact with the agent, distribute or consolidate management information. Additional management protocol APIs provide a standard way to interact with legacy management technologies (SNMP, WBEM, TMN, …).
The latter level is used to implement the EML layer in the GESTICA architecture.

3 The GESTICA Architecture

3.1 Project Objectives

The GESTICA project aims at providing an integrated quality control over high bandwidth services such as Video on Demand, Multiconferencing or Hi-Fi Audio. The main concern in this context is that available protocols (e.g. IP) are not oriented towards isochronous transmission, overlaying applications having the burden of smoothing the unpredictable behaviour of underlying resources.

The challenge of the proposed structure is to combine the probabilistic nature of IP and ethernet with a management environment controlling the feasibility and quality of service criteria in a session-based manner. A TINA-inspired session management layer is then be in charge of the control of the availability of network resources in order to insure the required functionality.

The SML components are tightly linked to realistic network information through a network data repository which is constantly updated.

This section presents the GESTICA architecture starting with a global overview, after which it will detail the different Management Layers SML, NML and EML.

3.2 Architecture Overview

The GESTICA architecture is mapped on the layering of TMN:
As can be seen in figure 2, GESTICA combines the three classical TMN layers:

- The Service Management layer which is in charge of the user session access and its parameters, as well as service maintenance; this layer is based on a series of components distributed over CORBA.
- The Network Management Layer which delivers on demand connectivity to SML, as well as playing the role of network information repository regarding the network topology, the status of network resources, performance statistics etc. This layer makes use of a particular CORBA-based Management Agent: the TRAC agent.
- The Element Management Layer which manages equipments regarding their configuration on behalf of the NML, and track equipment status and network information in order to be delivered to NML. This layer makes use of Java Mbeans which hide the heterogeousness of equipments and protocols.

In other terms, the EML layer uses specific protocols (telnet, snmp etc.) to access equipments, while all the rest of the system runs over CORBA.

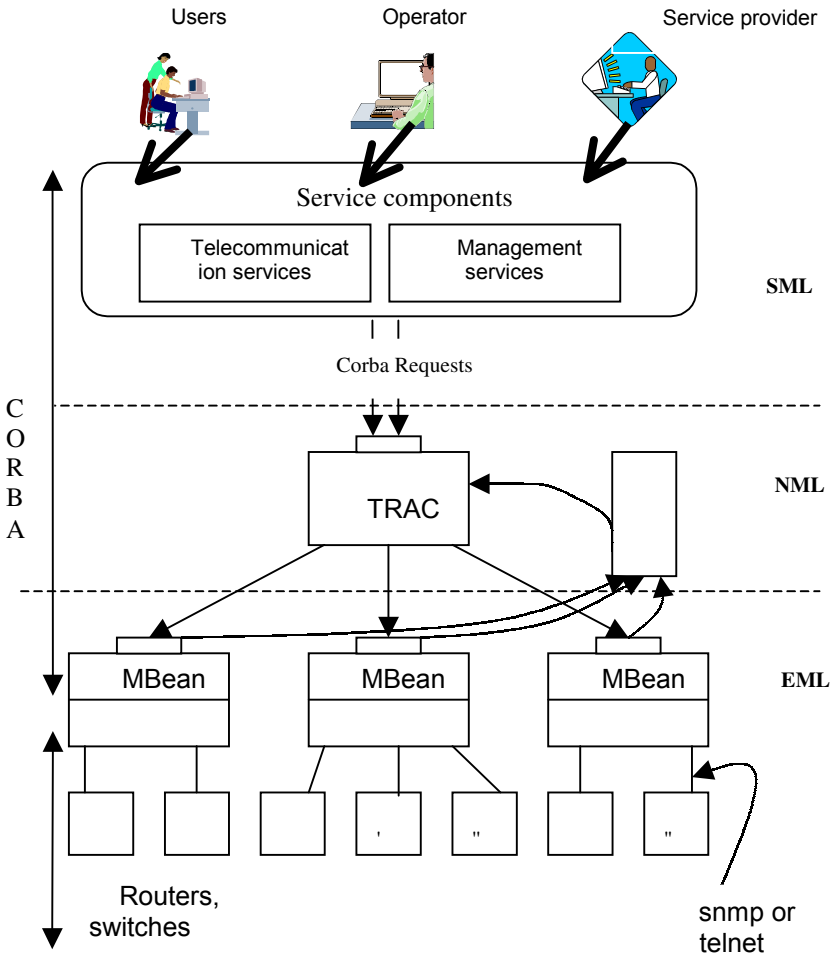

Fig. 2. GESTICA overview.

From a functional point of view the GESTICA system runs simultaneously in two modes:

- A planning mode where the users – in a session-oriented way mimicking the TINA approach - connect to the system at the SML level which then retrieves their user profile, connection classes, tariff selection etc. and allows them to choose the desired service with its parameters, notably the desired Quality of Service. Furthermore the session verifies if the requested service can be delivered in the required quality, given the status of the corresponding resources.
- A supervision mode where network information and statistics are constantly gathered and updated, keeping track in near realtime of e.g. ethernet collision statistics etc., in order to keep the NML layer informed of the status of the network. It is the SML layer which, regularly, interrogates the NML layer to verify the soundness of the network, and if it is not the case, makes the appropriate decisions.

3.3 EML Layer

The EML layer is composed of JDMK-based modules insuring a uniform presentation of equipments to the NML layer, disregarding the heterogeneity of routers, switches etc. of different vendors. They take in charge the conversion of high level requests issued by NML, to the specific protocols (telnet, snmp or even sockets) of the particular equipment. Reciprocally, they poll equipment status information regularly (again in the particular protocol of the attached equipment) in order to transmit it to NML.

The available implementation makes use of SNMP mainly for configuration purposes, as well of the `telnet` protocol. It allows to build on demand end-to-end VLANs in the available platform (two routers, 6 switches) as well as to accept requests to upload relevant network statistics such as collision percentages or packet flow measures.

Figure 2 summarizes the situation of the Mbeans in the Gestica architecture.

3.4 NML Layer

3.4.1 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.
- 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.

3.4.2 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 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.

3.4.3 Implementation Notes

The implementation of the agent is 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 emphasized. 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.

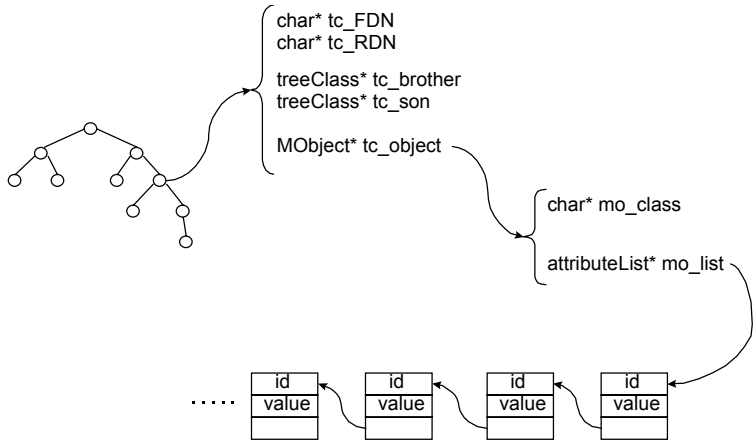

Fig. 3. Object relationships within the agent.

The agent's IDL interface contains all the information and structures needed to 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) and so forth. Other classes defined in the NRIM information model that were used are `link`, `topologicalLink`, `linkTerminationPoint` and `topologicalLinkTerminationPoint` (see figure below).

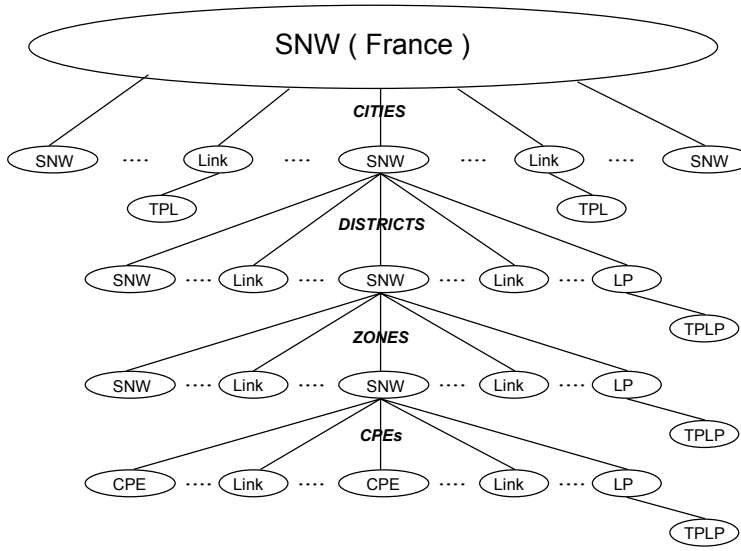

Fig. 4. Simulated national backbone.

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.

3.5 SML Layer

The Service Management Layer implements the User Session Manager (USM) and the Network Quality of Service Supervisor (NQSS).

3.5.1 The User Session Manager

The USM takes in charge the management of the user session in two steps. First, the USM realises the function of session access portal : to invite the user to identify himself and to choose a particular service including its parameters if necessary. Second, it verifies that the required service can be insured properly given the actual status (their load, in particular) of the corresponding resources as viewed at the NML layer. A service that would endanger the quality of already running services would be refused : this way, existing services are guaranteed to remain stable all over their session (in the hypothethis that the network carries only GESTICA-managed bandwidth).

3.5.2 The Network Quality of Service Supervisor

The NQSS implements the function of quality maintenance over the network. It constantly polls the NML layer in order to gather load and status information. In the case of node criticality (e.g. more than a certain percentage of collisions at EML, an information which is mapped into NML information in the form of load statistics), the NQSS makes appropriate decisions, for instance rerouting of certain services into nodes that are less critical. In certain extreme cases, the NQSS would even interrupt certain (less important) services in order to relieve the network. The priority of a service is a parameter which is a session characteristic and related to tariff classes.

The figure 5 shows how these components run over NML.

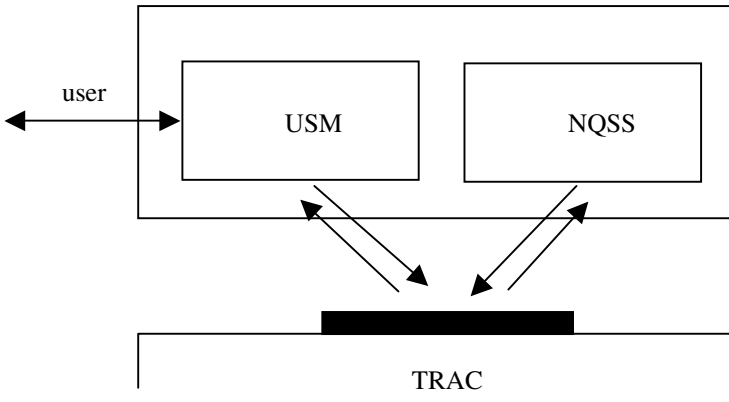

Fig. 5. The Service Management Layer.

3.6 Overall considerations

Two remarks can be made on the GESTICA system :

- The proposed approach is largely technology-independant because 1) the EML components hide the underlying equipments and 2) the Information Model that is

used at the NML layer is the NRIM which is technology transparent. In principle, GESTICA could insure the same services over ATM networks for example, or even mixed networks.

- GESTICA is largely independant of the actual Service as well. The proposed Service, for economical reasons, is VPN configuration but the architecture would not have changed if it would have been vIP or VoD.

4 Conclusions

The work presented in this paper is an attempt to integrate in a single system all components involved in Services Management as well as to demonstrate the feasibility of this approach. At the same time, the system shows how the TMN architecture may be used in CORBA-based systems, making profit of the features of both frameworks. This aspect could be the base for new, enhanced management systems both easier to develop and to deploy than CMIP-based TMN.

On the other hand this system demonstrates also how an IP network can be managed to exhibit a more deterministic behaviour, and this is a key feature for many users which appreciate this technology, but are reluctant to accept its low determinism.

Last, the proposed approach which implements LAN-based VPN management, can be easily extended to other services such as VoD or VoIP because the actual architecture remains independant of the managed service.

References

1. M. Chapman, F. Dupuy, G. Nilsson, *An Overview of the Telecommunications Information Networking Architecture*, in Proc. of the Telecommunications Information Network Architecture *(TINA'95)* International Workshop, Melbourne, Australia, 1995
2. Object Management Group, The Common Object Request Broker: Architecture and Specification (CORBA), Version 2.0, 1995
3. Object Management Group, CORBA Services: Common Object Services Specification (COSS), Revised Edition, 1995
4. Java Dynamic Management ™ Kit (JDMK) SUN Microsystem Inc., Dynamically Extensible Management Solutions for today and tomorrow, http://www.sun.com/software/java-dynamic/
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. Java Management Extensions (JMX) SUN Microsystems, Inc. http://www.sun.com/products/JavaManagement/index.html
7. J.Landru, H.Mordka, P.Vincent *M*odular *O*pen *N*etwork *A*gent for *C*ontrol *O*perations IEEE Noms'98 New Orleans 15-20 february 1998 http://www.enic.fr/people/landru/publications/ieee-noms98/index.htm
8. ITU-T Rec. M.3010, Principles for a Telecommunications Management Network (TMN), Study Group IV,1996

9. D. Ranc, G. Pavlou, D. Griffin, *A Staged Approach for TMN to TINA Migration*, in Proc. of the Telecommunications Information Network Architecture (TINA'97) International Conference on Global Conference of Telecommunications and Distributed Object Computing, pp. 221-228, IEEE Computer Society, Santiago, Chile, 1997
10. 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
11. ITU-T Rec. X.701, Information Technology - Open Systems Interconnection, *Systems Management Overview*, 1992
12. ITU-T Rec. X.710, Information Technology - Open Systems Interconnection, Common Management Information Service Definition and Protocol Specification (CMIS/P) - Version 2, 1991
13. ITU-T Rec. X.722, Information Technology - Open Systems Interconnection, Structure of Management Information - Guidelines for the Definition of Managed Objects, 1992
14. ITU-T Draft Rec. X.901-904, Information Technology - Open Distributed Processing, *Basic Reference Model of Open Distributed Processing,* 1995