Managing the TMN

Stelios Sartzetakis, Costas Stathopoulos, Vana Kalogeraki, David Griffin
Institute of Computer Science, Foundation for Research and Technology - Hellas
PO Box 1385, Heraklion, GR 711-10, Crete, Greece.
tel.: +30 (81) 39 17 27, fax.: +30 (81) 39 16 09.
e-mail: {stelios, stathop, kaloger, david}@ics.forth.gr

Abstract. Metamanagement provides a means to manage the management processes, systems and software comprising a TMN system. In this paper we present the basic requirements of metamanagement, the design and implementation of these TMN metamanagement extensions. The OSI Directory Service is used to store information about the TMN resources, CMIS/CMIP is the basic management service/protocol, and FTAM is used for distribution of software and configuration files. Metamanagement agents communicate with other specialised management processes, providing the TMN metamanagement service, according to the OSI manager/agent model. We show how this model is used to aid the configuration of prototype TMNs and how this can be automated to evolve to a self-managed, self-configured TMN.

1. Introduction

Whereas the scope of the Telecommunications Management Network (TMN) [M. 3010] is to manage networks and services, the scope of metamanagement is to manage the TMN itself. Although the management of the TMN is identified in the standards as one of the basic TMN management services, it is marked as “for further study” [M. 3200]. The purpose of this paper is to describe an architecture that provides metamanagement functionality, initially intended for use by a TMN operator, and when extended, by TMN processes.

It is quite often argued that the TMN is very complex and its implementations based on OSI management technology, despite being very powerful, are difficult to implement because of the complexity of the underlying service/protocol (CMIS/P) and the power and expressiveness of the associated information model. The RACE II ICM project has managed to design and implement large TMN systems [Gri95] and has successfully demonstrated the use of these systems both in ATM testbed experiments and in larger simulated ATM networks. The OSIMIS TMN platform [Pav93b] developed earlier in the project is now enriched with *metamanagement* agents and specialised management processes for manipulating management applications and for accessing them in a distributed fashion. We introduce the term *metamanagement* to refer to the *function of managing the management processes, systems and software comprising the TMN*. This includes the start-up and shutdown of management processes, recovery procedures in case of failures, as well as automatic code and configuration files distribution and updates, so that an easily configured and fault tolerant TMN is provided.

The TMN is a distributed management environment [M. 3010]: a variety of distributed processes interact over a Data Communication Network (DCN) in order to manage the services and resources of an underlying managed physical network. Management information is maintained on a large number of distributed agents. These agents interact with a variety of management applications (managers) over the DCN by using the OSI manager/agent model which forms the basis for both intra-TMN (Qx/Q3 interfaces) and inter-TMN (X interface) communication. The collection of agents and managers (or, with one name, management processes) implement the Operations Systems (OSs), Mediation Devices (MDs), Q Adaptors (QAs), Workstations (WSs) and Network Elements (NEs) of the TMN. The main objective of the TMN is to offer a variety of management services that will provide for effective service and network management.

Within this paper we will assume that the various TMN management processes interact according to the enhanced OSI manager/agent model proposed in [Sta95]. This means that each management process is capable of not only updating the Directory with information about its state, functionality, location etc., but also interrogating the same information about other process via Directory queries. This provides the critical information about which processes exist in a TMN, the management services they provide and a location transparent way for associating with them. This kind of information, also called Shared Management Knowledge, forms the basis for our metamanagement system which should be able to maintain a global view for the various TMN building blocks and the way they interact. The metamanagement system described in this paper is implemented on top of OSIMIS [Pav93a] which supports the mechanisms described in [Sta95].

The TMN building blocks (OS, MDs, QAs, NEs and WSs) are installed on a network of computers which communicate over the DCN. In this paper, we call these computers "*TMN hosts*". These may be independent workstations, like in our testbeds, or special management single board computers inside the network elements themselves. The only requirement for a *TMN host* is to run a multitasking operating system which uses OSI to communicate with other hosts in the TMN. Within certain limitations any TMN building block may be instantiated and run on any TMN host. These limitations are related to:

> *performance issues*, e.g. QAs and some other building blocks may need to be physically close to NEs or other building blocks in order to minimise TMN traffic.
>
> *software availability*, e.g. the latest version of each binary image executable and the necessary configuration files need to be available to the host.
>
> *capabilities of the host*, e.g. RAM size, processing capability, current processing load.

Being another form of management, metamanagement covers all of the five management functional areas: fault, configuration, accounting, performance and security.

> Configuration management of the TMN deals with the instantiation and termination of TMN building blocks on TMN hosts.

Fault management of the TMN is concerned with handling failures of either TMN hosts or TMN building blocks or link failures in the DCN.

Accounting management of the TMN keeps track of the overall TMN activity by maintaining and processing a variety of log data files.

Performance management of the TMN ensures that the allocation of TMN building blocks to TMN hosts satisfies certain TMN performance criteria.

Security management of the TMN certifies the authorised exchange of management information between the management processes and ensures user authentication and authoritative only access to the management processes.

The above areas are closely related. For example, configuration must take into account performance issues and also the current state of failures, subsequent failures in the TMN will result in reconfiguration, faults in the TMN will cause performance degradation and so on.

The system presented in this paper is a first step to this direction, and for the moment aids TMN operators in the tasks of configuration, fault, and performance management of the TMN.

There are two important issues in a distributed TMN environment:

The provision of an initialisation/configuration system to start the TMN management processes

To ensure availability of the management processes, coping with failure cases.

In essence, we are dealing with relocatable TMN application processes (that is, processes that are not constantly bound to some specific TMN host). Having such processes in the TMN is a major advantage, because for example reconfiguration in case of failures is possible, but makes metamanagement a more challenging task.

The fundamental requirement of metamanagement is to provide the mechanisms for the efficient operation of the TMN. The overall functionality of the metamanagement service can be decomposed to the functional management areas described above. The user of the service is not only the TMN operator who has to configure, and occasionally reconfigure, the TMN but also the various management processes that have to report whenever there is a problem in communicating with other processes.

In the first implementation we dealt with the case of the TMN operator who wants to initially configure and start the TMN. We offer a tool to the TMN operator that helps him configure the TMN. After the configuration, the metamanagement system monitors the operation of the TMN and informs the operator of any exceptional conditions. It is the responsibility of the operator to take appropriate action. This initial ‘manual’ mode of operation provides the basic capabilities of metamanagement, and permits an evolutionary path towards its eventual automation by incorporating intelligence into the metamanagement components themselves.

In the next section, we present the metamanagement model for the TMN management processes. Section 3 gives the design details on the implementation that we developed

for the first version of this work. We conclude by discussing the open issues in managing the TMN and our plans for future enhancements.

2. The Metamanagement System: Model and Current Operation

Figure 1 depicts the metamanagement system model. Two new processes are introduced: the metamanagement agent (MMA) and the metamanagement operations system (MMOS). It is assumed that every TMN host runs a MMA acting in the agent role containing a MIB presenting managed objects representing the management processes running on that TMN host. Additionally, the MMA contains an FTAM client that is capable of connecting to an appropriate FTAM server in order to retrieve software and configuration files not present locally on the TMN host.

The MMOS acts in the manager role. It implements an Operations System (OS) equipped with an HCI (Human Computer Interface) through which the TMN operator is capable of performing appropriate management operations on the MMAs and therefore the management processes on the TMN hosts. Both the MMA and the MMOS contain a Directory User Agent (DUA) that performs the Directory updates and queries described in [Sta95].

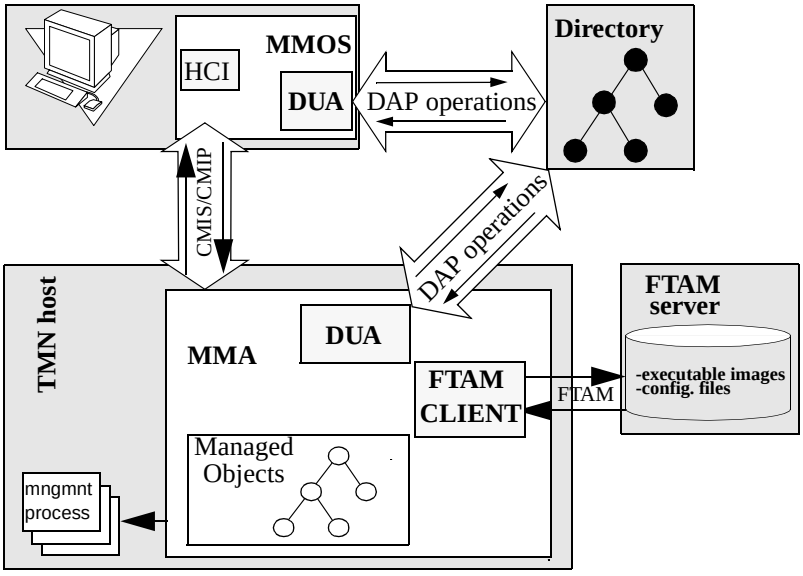

FIGURE 1. The OSI Metamanagement Model

The MMA and MMOS are critical for the operation of the TMN and are therefore implemented as fault tolerant processes as discussed later in this section.

The MMOS obtains information concerning the running MMAs, and therefore the available TMN hosts, through Directory search operations. Additionally it keeps the current load of each TMN host and an estimation of the load of the DCN links interconnecting the MMOS to that particular MMA. The load of each TMN host is

retrieved from the corresponding MMA's MIB. The load of the DCN links is currently measured by periodic round trip time estimations for MMOS messages to MMAs. In the future more sophisticated measurements based on performance monitoring of the TMN will be used.

The TMN operator, using the MMOS HCI, determines the appropriate TMN host on which to run a new management process. This decision is based on the information about MMAs displayed on MMOS HCI, on the knowledge that the operator has about the topology of the managed network, the topology of the TMN DCN, and the points where the TMN connects to the underlying managed network elements.

After an appropriate MMA has been identified, the new management process is started via a CMIP "create" operation from the MMOS to the chosen MMA, i.e. the MMOS creates a managed object within the MMA's MIB. The attributes of the new managed object (MO) include the Distinguished Name (DN) of the process that this MO represents and the DN of the software package that the new process should run. The process's software package consists of the necessary executable, configuration and data files and this information is stored in the Directory (described in Section 3.)

The MMA reads from the Directory the entry corresponding to the process's software package and checks whether the necessary files are available locally. If the software package is already available, the MMA executes the process and updates its MIB with the newly created MO. After receiving the appropriate object creation message the MMOS updates the Directory with information about the started process. This information includes the MMA that is responsible for the process, the software package that the process runs and the execution vector used to start the process. Note that the MMOS updates the directory rather than the newly instantiated management processes since these are not usually aware of all the required update information.

On the other hand, if the software package is not available on the TMN host (or it is out-of-date), the MMA retrieves the files from the TMN FTAM server located in one of the TMN hosts (for increased availability, more than one FTAM servers could be available, mirroring the TMN data). After retrieving the necessary files the MMA proceeds as above.

As well as creating new management processes, the TMN operator may need to terminate already running management processes (e.g. reconfiguration of the TMN). A prerequisite for this is that the MMOS should know which processes run and on which TMN hosts. Again, this information can be retrieved from the Directory. To terminate a management process, the MMOS issues a CMIP "delete" operation on the appropriate managed object in the appropriate MMA.

It is critical that the MMOS and the MMAs are implemented in a way that ensures TMN reliability and fault tolerance. Fault management of the TMN refers to the mechanisms that ensure that faults in the TMN (failures in management processes, TMN hosts and the DCN) are detected and appropriate actions are taken by the metamanagement system. One of the important issues here is that the metamanagement system has to manage itself, that is the MMAs and MMOS have to deal with their own failures. In order to achieve this, each MMA communicates with a

lightweight process[1] (running in the same TMN host as the MMA) by periodically sending “I am alive” messages. If the MMA does not respond for some time, the lightweight process assumes that it has died, and is responsible for restarting it. There is provision for a log file that keeps checkpoints describing the latest state of the MMA. This log file is read by the MMA on start-up, ensuring that it recovers its latest state to maintain consistency. Conversely, if the lightweight process dies, the MMA will restart it. A similar approach can be followed for the MMOS, although for the time being MMOS faults are simply detected by the TMN operator.

Since the MMOS periodically requests the load status of each TMN host by issuing CMIP requests to every MMA, it has a simple way for detecting either a faulty TMN host or a DCN failure whenever a MMA does not respond (bearing in mind that MMAs are reliable processes as described above). In this case, the TMN operator has to take appropriate action by either reconfiguring the TMN (in case of a faulty TMN host) or fixing any failing DCN links. Note that in order for the MMOS to distinguish among the above two failures special functionality should be implemented (e.g. alternate routes for connecting to MMAs) that help determining whether an MMA is unreachable because of a host crash or a link failure.

An alternative approach to the polling method, is for the MMAs to provide event reports such as ‘overloaded TMN host’. In this case other management processes required to communicate with failed management processes on a crashed TMN host would be responsible for informing the MMOS that the required management process is not available. This way the MMOS will be informed of TMN host crashes.

Each MMA periodically checks whether the management processes running on its TMN host are alive (e.g. by issuing a CMIP get request for the top - namely, the system - MO). In case of a process failure, the MMA sends an appropriate notification to the MMOS. The TMN operator (through the HCI of the MMOS) is always able to have a complete view of the TMN. He can query the directory for TMN hosts, the processes that run or are available, where they run and their state. As soon as the metamanagement system informs him that a process is dead (or unreachable), he can decide to restart it. Additionally, he may decide to reconfigure the system, so to improve performance, by terminating and restarting a management process in another TMN host. This will usually involve complicated recovery procedures, especially when a TMN process has to recover its previous state. As mentioned above, other management processes also report to the MMOS whenever they fail to associate with a peer management process. For the present, the TMN operator has to track the reason for that failure.

---

1. In this context, lightweight process does not refer to a thread but to a special separate process that implements some simple function without supporting any “heavyweight” OSI application protocol (e.g. CMIP). It is achieved via a local communications method (internal to the host).

3. Design Considerations

3.1 Directory Objects for Metamanagement

The MMOS is an OSI application process, hence it will be stored in the OSI Directory just like other management processes [Sta95]. The communication with the MMOS is realised through one application entity. The same holds for the MMA. The attribute that distinguishes a metamanagement process is the management service that it implements (namely, “metamanagement”). This is the service that both MMAs and MMOSs provide, the difference being the management functions they performs.

One question here is to which TMN building block the MMAs belong. Although this is still open to discussion, the MMA can be considered a QA since it provides a Q interface to the operating system dependent process invocation mechanisms.

Two extra attributes will be used in the directory object of each management process. The first points to the MMA containing the MO representing the management process:

```
metaManager ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DistinguishedNameSyntax
SINGLE VALUE
```

The above attribute is an optional attribute of the *managementProcess* object class (see [Sta95]). The value of this attribute is chosen during the initial configuration procedure of the TMN and is updated by the MMOS as described in the previous section.

The second attribute points to the software package necessary to run the process:

```
usesSoftwarePackage ATTRIBUTE
WITH ATTRIBUTE-SYNTAX DistinguishedNameSyntax
SINGLE VALUE
```

The software package is represented by a Directory entry containing attributes for the required executable, configuration and data files and the generic argument vector that should be supplied on starting the process. Additional attributes will contain the version of the software package, description of the purpose of the process, operating system requirements, hardware requirements etc. The ASN. 1 definitions for the above entry follows. A new object class is defined as:

```
icmSoftwarePackage OBJECT CLASS
SUBCLASS OF top
MUST CONTAIN{execVectorFormat, binaryImage, configurationFile}
MAY CONTAIN{spVersion, description, OSrequired, hardware}
```

The definitions for the attributes defined in the *icmSoftwarePackage* object class are:

```
execVectorFormat ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreString
SINGLE VALUE
binaryImage ATTRIBUTE
WITH ATTRIBUTE-SYNTAX unixfileSyntax -- a unix file path for first
version
MULTI VALUE
configurationFile ATTRIBUTE
WITH ATTRIBUTE-SYNTAX unixfileSyntax -- a unix file path first version
```

```
MULTI VALUE
spVersion ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreString
SINGLE VALUE
OSrequired ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreString
MULTI VALUE
hardware ATTRIBUTE
WITH ATTRIBUTE-SYNTAX caseIgnoreString
MULTI VALUE
```

Additionally, the QUIPU *execVector* attribute [Kil91] will be used in every management process entry keeping a character string with the actual arguments that were used at process invocation.

The file attributes, mentioned above, can also be pointers to entries below a subtree of the DIT that keeps file information. This will not be the case for our first implementation but it will be used in the future. A general description of the directory objects representing files can be found in [Bar94]. Based on the information contained in this subtree an FTAM client will be able to retrieve the files from an appropriate FTAM server. For our metamanagement system this FTAM server will make available all the software packages needed for the TMN management processes. Security and access control are currently under design as we briefly mention in the open issues section.

3.2 Definition of Managed Objects

The Metamanagement Object Class definitions used in our first implementation are the following:

Managed Object Class definition

```
ManagementProcess        MANAGED OBJECT CLASS
DERIVED FROM             top;
CHARACTERIZED BY         ManagementProcessPackage;
REGISTERED AS            {icmManagedObjectClass 1};
```

Name Binding definition

```
ManagementProcess-system NAME BINDING
SUBORDINATE OBJECT CLASS ManagementProcess AND SUBCLASSES;
NAMED BY
SUPERIOR OBJECT CLASS    system AND SUBCLASSES;
WITH ATTRIBUTE           ManagementProcessTitle;
BEHAVIOUR                MngmtCreateBehaviour;
CREATE                   WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE                   ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS            {icmNameBinding 1};
```

Package definition

```
ManagementProcessPackage PACKAGE
BEHAVIOUR                MngmtProcessClass;
ATTRIBUTES
ManagementProcessTitle   GET,
```

```
ManagementProcessExecVector GET;
NOTIFICATIONS
objectCreation,
objectDeletion;
REGISTERED AS            {icmPackage 1};
```

Attribute definitions

```
ManagementProcessTitle ATTRIBUTE
WITH ATTRIBUTE SYNTAX
UCLAttribute-ASN1Module.SimpleNameType;
MATCHES FOR EQUALITY, SUBSTRINGS, ORDERING;
BEHAVIOUR                MngmtProcessTitleBehaviour;
REGISTERED AS            {icmAttributeID 100};
ManagementProcessExecVector ATTRIBUTE
WITH ATTRIBUTE SYNTAX
UCLAttribute-ASN1Module.GraphicString;
MATCHES FOR EQUALITY, SUBSTRINGS, ORDERING;
BEHAVIOUR                MngmtProcessExecVectorBehaviour;
REGISTERED AS            {icmAttributeID 101};
```

Behaviour definitions

```
MngmtCreateBehaviour BEHAVIOUR
DEFINED AS
“The objects could be created either when initializing the agent or
during the program execution as required by the manager. There is no
limit on the number of object instances that can be simultaneously
generated.”;

MngmtProcessClass BEHAVIOUR
DEFINED AS
“This is a class of managed objects that represent running processes.
It includes the attributes of these objects, the actions that are
performed by them and the notifications that are generated upon object
creation and deletion.”;

MngmtProcessTitleBehaviour BEHAVIOUR
DEFINED AS
“The ManagementProcessTitle is an attribute type whose distinguished
value can be used as an RDN when naming an instance of the
ManagementProcess object class.”;

MngmtProcessExecVectorBehaviour BEHAVIOUR
DEFINED AS
“This attribute type represents an execution vector. It contains the
actual arguments used when starting the process.”;
```

The *ManagementProcessExecVector* attribute has the same value as the *execVector* directory attribute. Although in the above definitions we do not have any actions defined it is expected that actions like process recovery will be identified in the future.

4. Open Issues

As we stated in the introduction, the metamanagement service has yet to be formally defined in the standards. Until then, a continuous evolution of the described system is under way implementing many features that, from our experience, we consider useful.

Although the ultimate responsibility for managing the TMN processes belongs to the TMN operator, some responsibility may be delegated to the MMAs. This means that, if one of the management processes running on a host die abnormally, the host's MMA might be able to restart it itself.

Additionally, whenever a new version of a software package is available in the TMN FTAM server, the related directory entries are updated from the MMOS. It is the TMN operator's responsibility to find which management processes use this data, decide whether existing processes need to be restarted with the new version of the software (or data), and instruct the appropriate MMA to retrieve the new version of the software package and restart the processes.

Another area of further work is in the application of security and access control mechanisms. It is important to restrict the access of managers to the MMAs to ensure that only the authorised MMOS is able to initiate new processes and, probably more importantly, to terminate existing processes. Directory access control lists will ensure that only the metamanagement processes will be able to retrieve the relevant information from the Directory. Authentication information for the FTAM sessions will be also kept in the Directory. At the time of writing security mechanisms are being incorporated into the OSIMIS platform, when they are available, they will be used by the metamanagement system.

Other remaining open issues are related to the inclusion of more intelligence into the MMOS. Currently it is assumed that a human TMN operator will decide which TMN hosts should run the management processes. But, by providing algorithms and heuristics for choosing appropriate TMN hosts according to: the TMN topology, loading of the TMN hosts and the DCN, the specific requirements of individual management processes; the MMOS will be able to make TMN configurations with little or no human intervention.

Once the MMOS is able to make automatic decisions on the placement of new management processes we are presented with the opportunity for the automatic, self-configuration of the TMN. In this way, only a single management process needs to be instantiated "manually", from that point, whenever this process needs to associate with peer management processes, and it is discovered that the required management process is not running, it will contact the MMOS, who will then decide where it should run and instantiate it via the appropriate MMA.

5. Conclusion

Starting from the fact that a management system for the management system itself is needed, we designed the system described in this paper. We implemented it, and it is currently in operation in experimental TMNs consisting of 10s of management processes. We have shown that a management model based on OSI systems

management concepts is rich enough to support the requirements of a metamanagement system. This type of management is typical in distributed systems contexts but we have shown that the OSI management and directory models provide a very good solution in an OSI management environment.

6. Acknowledgements

The original idea for the management of TMN components using specialised agents was contributed by George Pavlou. The work was carried out under CEC RACE II project R2059 ICM (Integrated Communications Management). The authors would like to thank all the ICM members for their feedback and support (especially the UCL OSIMIS team).

7. References

[M. 3010] ITU/CCITT Recommendation M. 3010, *Principles for a Telecommunications Management Network*, Geneva, October 1992.

[M. 3200] ITU Recommendation M. 3200, *TMN Management Services: Overview*, Geneve, October 1992.

[D10040] ISO/IEC DIS 10040, *Information Technology - Open Systems Interconnection - Systems Management Overview*, June 1991

[X. 500] ITU/CCITT Recommendation X. 500, *The Directory - Overview of Concepts, Models and Services*, December 1988.

[Pav93a] Pavlou, G., S. Bhatti and G. Knight, *OSIMIS User Manual Version 1. 0 for System Version 3. 0*, 02/93

[Pav93b] Pavlou G., *The OSIMIS TMN Platform: Support for Multiple Technology Integrated Management Systems*, Proceedings of the 1st RACE IS&N Conference, Paris, 11/93

[Gri95] Griffin D., Georgatsos P., *A TMN system for VPC and routing management in ATM networks*, Proceedings of the fourth international symposium on integrated network management, ISINM95, ed. Sethi A., et al., Chapman & Hall 1995. p356

[Sta95] Stathopoulos C., Griffin D., Sartzetakis S.: *Handling the Distribution of Information in the TMN*, Proceedings of the fourth international symposium on integrated network management, ISINM95, ed. Sethi A., et al., Chapman & Hall 1995. p398

[Bar94] Barker Paul, *Implementing FTP archive searching using X. 500*, Internet Draft, (aka OSI-DS 40), version3, October 1994.

[Kil95] Kille, E. S. (1991) *Implementing X. 400 and X. 500: The PP and QUIPU Systems*, Artech House, Boston MA.