INTERNET NETWORK SERVICES MANAGEMENT FRAMEWORK

B. Dias(*)<dias@uminho.pt> A. Santos(*)<alex@uminho.pt>, F. Boavida(**) <boavida@dei.uc.pt>

(*)Departamento de Informática, Universidade do Minho
(**)Departamento de Engenharia Informática, Universidade de Coimbra

ABSTRACT

This document presents a new approach to Internet Network Management without changing the basic rules of the interface and encapsulation mechanisms of standard management transport protocols. The **Internet Network Services Management Framework** (INSMF) tries to overcome the most important limitations of the **Internet Network Management Framework** (INMF) by adding a new extension-model to it, using a network service management distributed architecture that provides services management functions with any desired level of functionality. The specification of a generic Domain Name Service (DNS) Management Service is presented, as a way to illustrate the capabilities and potential of the proposed framework.

1. INTRODUCTION

The growing complexity of distributed Internet Services and Applications, supported across various network service providers that use numerous network hardware and software technologies, has reinforced the importance of network management. The hardware devices of a network have increased in number and computational power. Also, a single network device can support numerous protocols, services and applications. All this demands new capabilities from the INMF. Despite the latest improvements, the most important limitations of the INMF model are the excessive centralized agent/manager paradigm [18,22,23,24] and the simplified abstraction [18,19] provided by the Management Information Base (MIB) objects. These limitations are difficult to overcome without a substantial change or extension to the model.

Although there has been continuous evolution in the INMF, the most recent achievements being the group of documents that form the 3rd Version of the Simple Network Management Protocol (SNMPv3), that is, INMFv3 [1,2,3,4,5,6,7,8], and the 2nd Version of the Remote Monitoring Management Information Base (RMONv2) [10], it is recognized that this framework is not providing what its users demand: an *efficient mechanism for global management of Network Services*. IETF efforts [9,10,13,14,15,16,17] try to use indirect mechanisms to provide some of the mid or high-level functionalities lacking on the original framework. The problem with the majority of these mechanisms is the integration with each other and with the low-level functionalities already available leading to an excessive number of indirect objects to manage and complex manipulation procedures. This is also true with respect to the model and mechanisms used to implement and provide security and access control [5,6].

The main limitations of the model and INMF mechanisms are:

- Lack of high-level functionalities available for building Network Management Applications. This is due to the excessive simplicity of the semantics associated with MIB objects [18,19].
- Low efficiency regarding the large amount of non compressed raw data transferred between managers and agents, since the processing of the raw data has to be done mainly on the managers/management application side. Also, object manipulation, like tables, is somewhat complex because managers can only rely on very basic, low-level procedures/methods. So, to accomplish something more complex than trivial manipulation, the manager has to issue a group of related low-level operations, all of them transmitted to the agents. Generally, all these operations have individual responses that are also transmitted through the network, increasing even further the impact on network bandwidth, which tends to be the most expensive resource on a distributed system.
- Scalability problems due to the over-centralized manager/agent model. This client/server architecture is very limited in the creation of various levels of management since the objects represent local resources on an individual network device. The problem has been identified since the INMF creation and some alternative management architectures [21] or SNMP extension mechanisms [24] were proposed to overcome this.
- Complexity of the access control and security procedures of the model created for SNMPv3. Despite being powerful, these mechanisms still use complex chains of small operations and need a large management burden for their extensive list of associated objects.

Some of these problems have been addressed in the past few years with the creation of mechanisms that try to increase the

functionality level of the managed objects and make the model less centralized. Such efforts include *distributed management* [22,23] and *management by delegation* [22] concepts that are partially implemented in the Remote Monitoring [10], Event [15], Notification Log [16], Expression [17], Script [14], Scheduler [13] and Manager-to-Manager MIBs. Nevertheless, the functionality of individual objects within these MIBs is still very limited and their management, from the managers point of view, still complicated and resource consuming.

In the approach defined on this article, these concepts of distributed management and management by delegation are implemented using the same integrated mechanism in the INSMF model and can be provided directly to the user as *Services Management Functions* (SMF).This model is intended for managing network services and distributed applications and can still use the encapsulation protocols and syntax rules of the INMF. The integration with the INMF is simple or, in some cases, straightforward, and all the MIBs can be re-used as *object resources*. So, the most relevant and recent concepts in network management are natively incorporated into the INSMF model.

2. THE INSMF MODEL

The framework of the proposed model tries to overcome the above limitations with a highly distributed architecture. This model is oriented to the management of Network and Application Services, not to the management of individual hardware/software network device resources. While the concept of Network Services Management is not new [20,26], it is not directly applicable on the INMF due to limitations on the management resources (MIBs) definition and usability.

The INSMF architecture *is like an extension* to the INMF architecture, it does not require it, nor replaces it. It can use the same protocol for transferring the management information (SNMP) and it should understand the semantics and syntax of the INMF standard MIB objects used as *management information resources*.

This new extension model should permit defining, on a per network service/application management basis, any level of management functionality and as many management levels as needed. Also, an unique management entity should be capable of implementing a management service with different functionality levels.

Figure A shows a generic management system with some possible interactions between standard INMF entities, INSMF entities and other types of management entities.

Typically, if an INSMF entity resides on a small network device dedicated to a limited number of simple tasks (or, more generally, a *network or application service*) with a limited set of hardware resources, it will implement a group of *Service Management Functions* (SMF) related to those tasks.

Due to resource limitations of this device, the implemented SMFs will have a low level functionality, somewhat closer to the standard INMF MIB objects, but the form in which the management information can be presented to a higher management level entity and the mechanisms available to manipulate it will be of a different nature, more oriented to management of network services than to local resources management. An example of this type of device is a small Ethernet or ATM switch. In these cases, the INSMF entity acts in the role of a **management service agent** – an entity that implements and makes SMFs, available to other remote entities. These remote entities using the SMFs are **management service managers**.

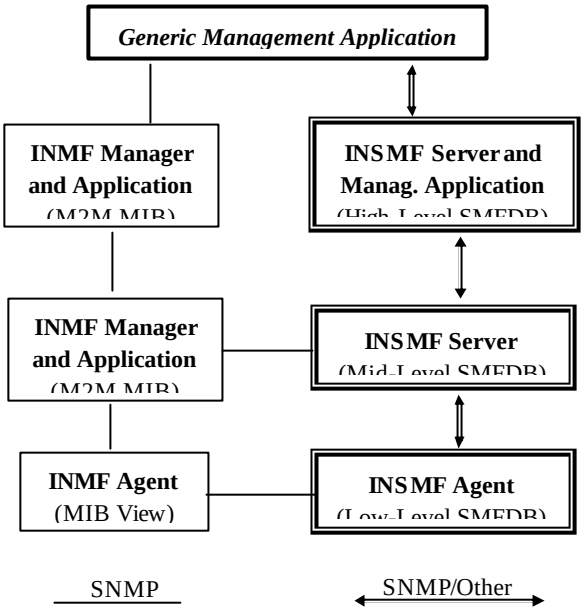

Figure A: The INMF model with the INSMF extension.

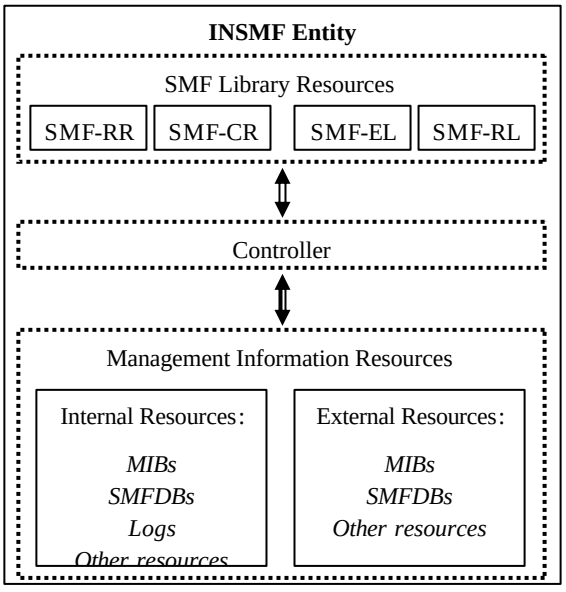

Figure B: The INSMF Entity Architecture.

On a broader network service, like an Ethernet LAN or ATM WAN backbone, we can find powerful devices dedicated to more complex tasks of various network services, linking all the smaller devices. The SMFs implemented on these *bigger* devices will have a higher level of functionality and can use the management information available from the smaller devices through their lower level SMFs, so these devices act in the role of a management service agent and manager, that is, a **management service server**.

An INSMF management service manager becomes an INSMF **management service application** when there is an additional interaction with an entity outside the INSMF framework that uses the management service to perform some other tasks and makes the results available by means of an interface different than SMFs.

Since SNMP can still be used as the management information transport protocol, the use of SMFs by SNMP management entities can be very useful and simple to accomplish.

2.1 The INSMF Entity

Figure B depicts the logical structure of an INSMF generic Entity tha can act both as an agent and a manager, that is, a management service server. The entity is divided into three main logical blocks:

- **Management Information Resources** – This block defines which resources are available/needed in the implementation of the SMF declared in the SMFL. These resources can be classified as *internal* or *external*. An internal resource is a resource available locally, directly accessible to the entity (this doesn't mean that the resource must be in the same device of the entity, only that his access is transparent and direct to the SMF code or controller tasks). An external resource is not directly accessible to the entity, it must be accessed with a non-transparent communication protocol, usually SNMP. The internal resources part has three sub-blocks dedicated to logging: the *External Resources Log*, the *Configuration Log* and the *SMF Execution Log*. The external resources log should register relevant information when accessing all the external resources, meaning, all the information needed to implement the SMFL. The configuration log should register the information about the INSMF entity working configuration/status. This information is essential to the implementation of the SMFL and the *Management Service Backup* (MSB) mechanism. The SMF execution log registers, when required, pertinent information from each SMF execution on the entity. This type of information is important for the implementation of the SMF dedicated to the management service itself (just like the SNMP Objects Group of the standard INMF MIB).

- **Controller** – This logical block represents the generic hardware and software resources (RAM, CPU, dispatcher, etc) of the device allocated to the execution and control of the scheduled SMF, to the implementation of the MSB, security and access control mechanisms and all the tasks related to an INSMF entity (transport mapping, logs management, etc) implementation. The controller is the *brain*, super visioning all the activities of the entity.

- **Services Management Functions Library Resources (SMFLB)** - This block represents the logical resources directly involved in the implementation of all the SMF that the entity is capable of and is divided in to four sub-blocks:

  The *SMF Reference List* (SMF-RL) - the list of all the SMF that the entity is capable of implement. The SMF-RL is further divided in three types of SMF references:
  - SMF dedicated to the management of generic Network or Application Services (IP Routing, DNS, NFS, for example); these SMF should be defined in standard SMF Definition Bases (IP Routing SMFDB, DNS SMFDB, NFS SMFDB, for example);
  - SMF dedicated to the management of the Generic Management Service itself; these SMF should be defined in a special SMFDB named Generic Management Service Functions Definition Base (GMSFDB); and the
  - delegated SMF, usually, by a higher level entity; the code for these SMFs is transferred from one entity to another by request (as part of the code of a another SMF) or by normal delegation from an entity acting as a manager to another entity acting as an agent; the references for these delegated SMF are, then, included in the SMF-RL of the entity; these delegated SMF can be defined on a standard SMF Base or created as needed by a Management Application and should use a very simple and limited programming language, named SMF Programming Language (SMF-PL).

The *SMF Execution List* (SMF-EL) - the list of all the SMF that are scheduled to execute on the entity. The scheduling attributes should permit conditional execution and repetition.

The *SMF Code Repository* (SMF-CR) - where the code for the delegated SMF is maintained. Note that the code really recorded on the SMF-CR can be in any compiled form or in the original SMF-PL. It's up to the entity software implementation creators to decide in which form to record and execute the code (compiled, interpreted or in between). Depending on certain code recording parameters and later execution parameters, the code for a certain delegated SMF can be maintained indefinitely on the repository (in these cases, the compiled may be preferable) or for a conditional amount of time or number of executions (for example, the code for a SMF can decide itself its own maintenance on the repository).

The results from the execution of the SMF from the SMF-EL are recorded on the *SMF Results Repository* (SMF-RR). The results to record depend on the default SMF definition and on extra results recording parameters defined on a per SMF execution basis. Results can be immediately transferred to the calling entity, maintained for later retrieval or just as a temporary place for passing information among SMFs.

2.2 Services Management Functions

The concept of SMF is the heart of this new service management approach. This new concept permits the implementation of various distributed management concepts and other important network management mechanisms that are not directly implemented in the original INMF. Also, it can be seen as an evolution of the Open Systems Interconnection (OSI) *Systems Management Functions* concept [25].

In addition, the SMF concept makes possible the coexistence of various levels of management functionality, ranging from the low level functions implemented on small devices to high level functions implemented on mid-level managers, available to other managers or management applications.

Because the INSMF model has a new overall object of management - network or application services – the entities in the service management system access management information by means of functions defined in a definitions base or through code delegation. Each network or application service should have an individual SMFDB and each SMFDB should have the SMF definitions divided by levels of functionality and type of management (like *Monitoring*, *Configuration*, *Accounting*, *Performance* and *Security*), when applicable. There is a special group of functions dedicated to the management of the Network Management Service itself. These funtions are defined on the Generic Management Service SMFDB.

The use of INSMF functions instead of INMF objects will simplify the manipulation of management information. The execution of complex manipulation tasks can be shifted to the entity acting as an agent or to the entity acting as a manager, depending on the functionality level available from each entity. As a result, the number of interactions and quantity of intermediate data transferred will relevantly decrease, increasing the efficiency of the model. Also, all the important distributed management mechanisms can be directly implemented and its use is simple and transparent to the manager.

- **Events/Alarms** – this mechanism is easily implemented on the INSMF through the use of a SMF, delegated or not, with conditional execution parameters; or on the SMF code itself. The first approach permits using any SMF as an event/alarm handler, while the second is preferable when creating dedicated event/alarm handlers with more complex trigger conditions.
- **Expression Evaluation** – this is done in a completely transparent way on the SMF definition or explicitly on a delegated SMF code by means of the use of SMF-PL commands and parameters.
- **Operations Scheduling** – all types of conditional execution are available with the SMF concept, either by means of direct conditional execution parameters or SMF code definition.
- **Script Delegation**, or more generally, **Management Delegation**, is obtained through the use of delegated SMF. In this case, there's only one possible programming language to delegate the SMF code, but the actual form of execution (compiled or interpreted) is not defined. The SMF-PL is intentionally simple but powerful enough for efficient delegation of management code. This pragmatic approach favors the ease of implementation and the creation of various levels of management.

2.3 SMF Definition

The code definition of an SMF, including his prototype definition is usually done in an SMF Definition Base, or less frequently, by an entity when delegates non standard SMF code to other entity.

The tasks that a standard SMF, that is, a SMF defined on a SMFDB, is expected to execute can be defined using any language, including human-oriented languages, except for the SMF prototype. The chosen language depends on the SMF definition author. Usually, the author of the SMF definition is not the future implementer on the network device, so the authors of this type of definitions tend to use human languages to describe the SMF behavior because it is harder to use formal languages when

the SMF denotes an high functionality level. The author of the SMF definition must be careful and explicitly and with accuracy describe all the tasks intended to be performed by the SMF and all the execution and results parameters associated semantics. There must be no room to ambiguity or unknown behaviors.

The SMF prototype definition must be written using a group of syntactic rules defined on the SMF-PL. This prototype definition should indicate the data type and default values of the execution and results parameters, and the SMF identification value.

So, the SMFs definition on the SMFDBs must include the prototype definition followed by the function abstract, the parameters description and the complete function description and, optionally, the SMF-PL code.

functionID(execution-parameters; results-parameters)
{ *function abstract* }
{ *execution-parameters description* }
{ *results-parameters description* }
{ *reserved-parameters description* }
{ *function complete description* }
{ *function SMF-PL code* }

To make this approach more flexible, when an entity calls a SMF does not need to use all the parameters and can use a parameter with its default value. Here is the generic SMF call definition:

FunctionID.call (reserved-parameters; execution-parameters; results-parameters)

The reserved parameters are needed for implementation of labeling, execution and access control and security mechanisms and are defined on a framework basis, that is, these parameters are defined for use on all SMFs, some being mandatory others being optional. They are not defined on a SMF basis, but the author of the SMF can further describe those reserved parameters behavior with a greater importance for his SMF implementation.

The entities running the called SMF always issues, at least, one SMF complete response.

ResponseID.call (reserved-parameters; results-parameters)

A powerful feature is that it is not necessary to end the execution of a SMF to inspect the results parameters values. Results parameter inspection is possible on defined check points of the SMF code. In the same way, it's possible to change SMF execution parameters on defined check points, if that's the intention of the SMF definition. This is very useful, for example, on the creation of SMF dedicated to gathering indirect statistics on a certain period of time or to monitor the status of a given group of objects or resources.

2.4 SMF Code Delegation

A manager can delegate a SMF code definition written on SMF-PL. This delegated SMF will be part of the SMFLB of the entity receiving the code for a period of time defined by the delegating manager.

2.5 SMF SNMP-PDU Encapsulation

SMF calls and responses can be encapsulated on any management transport protocol. There's no standard choice here, although SNMP seems to be a logical one, since it's a standard Internet protocol and the ISNM first objective is to become an extension of the INMF, despite its capacity of being a standalone network management system.

Each SMF call or response is encapsulated in an SNMP PDU. In this case, an SMF call will be mapped into one or several set-request operation PDUs and a SMF response will be mapped into one or several set-response operation PDUs. The mapping mechanism is simple: the *entire SMF call or SMF response* (or parts, if needed) is to be inserted on the variable bindings part of the PDU. The SMF call or response is, prior to be encapsulated, encoded using some SMF Rules of Encoding (SMFROE). There are various options to use as SMFROE, one of them being based on the Basic Encoding Rules (BER) used on the INMF framework. Each SMFROE option can be chosen on a per Management Session basis.

2.6 Security and Access Control

Despite major advances in the INMF security and access control model conveyed in version 3 of SNMP, it has to be said that the mechanisms needed to implement it are complicated and some argumentation has being made, mainly on the IETF SNMP mailing lists, in the sense that it lacks configuration flexibility since it can't allow the use of new, or most recent encryption mechanisms, like 3DES or AES. There is some draft work done in this area in order to enable the use of AES on the INMF.

Also, security mechanisms are configured on an agent/manager pair basis and the access control can be done on a per object granularity (or groups of objects), the object being an abstraction of an agent's resources (hardware, software or data).

In the INSMF model, it is not allowed to directly access these resources. The only way to control the behavior of the agent is issuing SMF calls. The code of these SMFs will manipulate a set of internal resources, most of them unknown to the manager. So, what's relevant here is *"what can a manager do?"*, that is, *"which SMFs"* and *"in what way the manager is authorized to use"*, and not *"what resources the manager can directly manipulate."* It's much harder to define a security and access control policy based on the resources than based on the actions that manipulate those resources.

2.7 INSMF User Access & Security Model

The INSMF User Access & Security Model (UACM) is divided in two parts:

- U**ser Access Control** - management of the users access, including *User Identification*, *User Quotas (Execution Time, Bandwidth and Memory)* and *Logs and SMF Library Resources Quotas*. This part must be completely implemented by any INSMF entity.

- **User Security** - management of all the security mechanisms that implement confidentiality, integrity and authentication. This part of the model has several options to be implemented, including a *minimum-security* option that could be used if it's used a secure management transport protocol, like SNMPv3.

2.8 Other features

The ISNMF Model and the SMF concept convey other important features that aim to create a vast array of high functionality management mechanisms available to Network Management and Systems Applications:

- *Management sessions* – A management session delimits an exchange of SMFs calls and responses. This is useful for time synchronization, time execution delimitation, error recovery and resources control.
- *Execution times and conditional execution* – the capability to delimit a SMF execution in terms of time consumed and to indicate a list of external conditions to the SMF definition itself (these conditions can be checked before or during the SMF code execution) facilitate the implementation of event/alarm handlers and any type of management monitors.
- *Generic SMFDBs* – Since any SMF code definition can use other SMFs already defined, there're some groups of base SMFs definitions on generic SMFDBs. These base SMFs are dedicated to generic network management procedures (INMF MIB object instance manipulation, INSMF entity configuration, etc) or just generic programming functions (math/statistics, flow control, conditions, code delegation, data manipulation, database systems interface, etc). This way, it's possible to re-use SMF definitions and define new libraries of SMFs with higher levels of management functionality.
- *Remote recording and manipulation of SMF results parameters on standard Database Systems* – To overcome possible problems of entity resources consumption there're some mechanisms that permit to record and manipulate management data outside the INSMF entity (agent or manager) using standard Database Systems to process the management data.
- *Pre or post processing of management data* – The possibility to process the management data with external methods to the SMF code itself augments the portability and adaptability of the SMF definitions and decreases the network resources consumption.
- *INSMF Entity Backup* – This mechanism is inspired on the DNS backup mechanism and uses the concepts of primary and secondary INSMF Entities per level of *Network Management Domains*.

3. DNS MANAGEMENT SERVICE

As an illustration of the proposed management model, this section presents an example of how a DNS Management Service could be defined and installed. This service is to be structured in two levels of functionality:

- A low level management service (LLMS) with separate management functions for DNS resolvers and for DNS servers. These low level SMFs should, whenever possible, access/manipulate local or remote DNS Resolver [11] and DNS Server MIB objects [12]. The INSMF entities implementing this LLMS could reside on the DNS resolver or DNS server but it is not necessary if the SMFs of this LLMS could be implemented using only MIB variables manipulation (local or remote). The simplest way is to let each implementer of the SMFs to decide whether to use MIB variables manipulation or some other management resources and manipulation processes. This last option would be more efficient and secure if the only management resources used were local and the implementer of the DNS software would integrate the SMFs of this LLMS just like they do with the DNS MIB modules. If the LLMS implementation is done independently of the DNS software, probably, depending on the Operating System, the only way of accessing some important DNS management information is through the MIB variables/SNMP interface since the DNS MIB modules are always integrated in the DNS software. Let us assume, in this example, both situations.

- A high level management service (HLMS) that uses those low level SMFs. Each INSMF Entity implementing these high level SMFs could manage one or several DNS domains on behalf of human administrators or other network services or applications. In the first case, the interface between the INSMF entity and the human user is done through a DNS Management Service Application. Regardless of its location, this application would let the DNS

administrator to manage the DNS service using high level tools that would permit:

- Remote configuration by means of application forms (without the need to know each operating system or DNS software syntax configuration) and immediate data validation;
- Graphic visualization of domain names hierarchy and some configuration automation of primary and secondary servers and resolvers;
- Maintenance of a user database for access control to the DNS management service application;
- Automatic zone transfers, if desired, after Resource Records actualization;
- Capability of automatic configuration of reverse mappings domains from address data on regular type A resource records;
- Reports of servers and resolvers statistics and network usage followed by automatic re-configuration of DNS server parameters;
- Automatic detection of anomalous situations, like DNS loops, server unavailability, etc;
- Easy configuration of a list of Automatic Management Procedures to be issue after a certain alarm has been produced (there should be a list of pre-defined/default procedures);
- Maintenance of a management backup system; this mechanism permits the automatic substitution of a faulty INSMF server (not the management application).

These generic requirements for a DNS management application are independent of the management framework used. But, if the management framework lacks high functionality native procedures, it must be implemented by the management application itself, which increases the development time and costs for network services management applications, increases network traffic and the availability of the high level management procedures to others management services or applications is very limited (at least through a standard management protocol). Using the INSMF model the management application can be dedicated to the user interface and to the implementation of the strategic network service management decisions.

3.1 DNS Management Service Architecture

Figure C represents a generic architecture for a DNS Management Service. Note that all the SMFs on the HLMS can be accessible through SNMP. If it's used a secure version (like SNMPv3), there's no need for extra security integrated on the INSMF portion of the system. If the SNMP version used is not secure (like SNMPv2), there will be the need for implementation of the security features possible on the INSMF mode. These considerations are also applicable to any other management transport protocol other than SNMP, when desired.

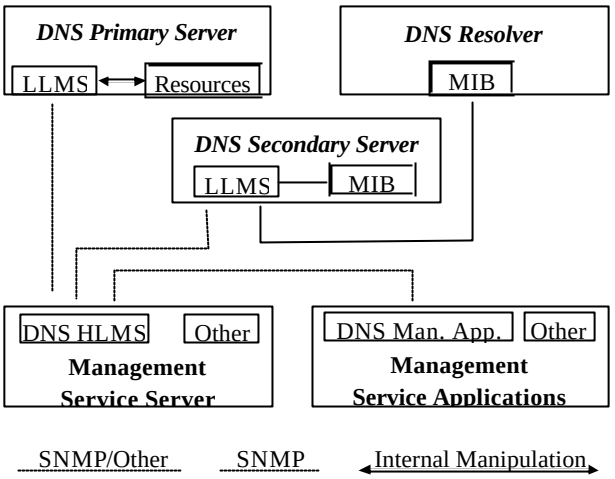

Figure C: DNS Management Service Architecture.

4. CONCLUSIONS AND FUTURE WORK

This article describes the overall architecture and the most important concepts behind the INSMF model, which can be regarded as an extension model to the INMF model since it tries to overcome its major limitations, but also as a standalone management framework that can use a management information transport protocol other than SNMP.

The specification of a generic DNS management service is presented as an illustration of a complete management framework that uses standard SNMP and DNS MIBs, as well as other specific management resources, as a basis for the definition of low level SMFs. Using these low level SMFs, other SMFs, with a higher level of management functionality, are defined that, in turn, should be used by other management services (like routing management) or end user management applications. By adding any desired level of functionality we can increment the usability of standard MIB objects and, at the same time, hide their complex manipulation procedures and save network bandwidth.

The implementation of the referred DNS management service is being done as part of the global project, that is, the INSMF framework complete specification. This includes the definition of the SMF Programming Language, the User Access Control model and some SMFDBs for generic management and other basic programming functions (math, flow control, conditional execution, etc).

The mechanisms that will need major developments in their definitions are the User Security and the Management Service Backup.

5. APPENDIX: DNS SMFDB

Here're some extracts of some LLMS SMF definitions:

```
DNS.LLMS.SOA.Configuration.1(
*'ID:SNMP-RESOURCE' snmp-resource,
'CHAR:DNS-DOMAIN' mname,
*'CHAR:E-MAIL' rname,
*'INTEGER:DNS' serial,
*'TIME:DNS' refresh,
*'TIME:DNS' retry,
*'TIME:DNS' expire,
*'TIME:DNS' minimum;
'LIST:SOA-REPORT' report 'RATE'/'DWR'/'BIR' )
{ Function to configure a SOA resource record [...] }
{ The first optional parameter indicates a local or remote SNMP DNS MIB
resource [...] }
{ The results parameter is a list of status records indicating the result of each
parameter configuration.
   RATE: Return At The End
   DWR: Delete When Returning
   BIR: Big Relevance [...] }
{ The reserved execution parameters should be used in a normal
way [...] }
{ [...] complete description of code behavior [...] }

DNS.LLMS.RR.Configuration.1(
*'ID:SNMP-RESOURCE' snmp-resource,
(0,6) 'CHAR:DNS-NAME','INTEGER:DNS-RR-
TYPE','INTEGER:DNS-RR-CLASS',
 'TIME:DNS', 'INTEGER:DNS-RR-DATA-LENGTH', 'DATA:DNS-RR'
rr (name, type, class, ttl, length, data);
 'LIST:RR-REPORT' report 'RATE'/'DWR'/'BIR')
{ Function to configure a list of generic resource record different than SOA
[...] }
{ The first optional parameter indicates a local or remote SNMP DNS MIB
resource [...] }
{ The results parameter is a list of status records indicating the result of each
resource record configuration [...] }
{ The reserved execution parameters should be used in a normal away [...]
}
{ [...] complete description of code behavior [...] }

DNS.LLMS.Server.Statistics.1(
*'ID:SNMP-RESOURCE' snmp-resource,
'CHAR:DNS-NAME' name;
 (0) 'CHAR:DNS-SERVER-STATS' statistics 'ROD'/'DWR'/'SOR')
{ Function to gather a list of statistics for DNS server. This function can be
executed like a monitoring procedure. [...] }
{ The first optional parameter indicates a local or remote SNMP DNS MIB
resource [...] }
{ The results parameter is a list of status records indicating the result of each
resource record configuration.
   ROD: Return On Demand
   SOR: Some Relevance [...] }
{ The reserved execution parameters should be used in a normal
way [...] }
{ [...] complete description of code behavior [...] }
```

And some examples of SMF prototype definitions for HLMS:

```
DNS.HLMS.Domain.Configuration.1(
'CHAR:DNS-DOMAIN' domain_name,
'CHAR:DNS-HOST' primary_server,
(0) 'CHAR:DNS-HOST' secondary_servers,
*'CHAR:E-MAIL' rname,
*'INTEGER:DNS' serial,
*'TIME:DNS' refresh,
*'TIME:DNS' retry,
*'TIME:DNS' expire,
*'TIME:DNS' minimum;
'LIST:SOA-REPORT' report 'RATE'/'DWR'/'BIR',
 'LIST:DNS-SERVERS-REPORT' report 'RATE'/'DWR'/'BIR')
{ Function to inicialize or re-configure a complete DNS domain;
  Except for RRs different than SOA [...] }
{ The first parameter indicates the Domain Name [...] }
{ The first results parameter is a list of status records indicating the result of
each SOA parameter configuration. The second is a list of status records wit
the result for each server indication.
   RATE: Return At The End
   DWR: Delete When Returning
   BIR: Big Relevance [...] }
{ The reserved execution parameters should be used in a normal
way [...] }
{ [...] complete description of code behavior [...] }

DNS.HLMS.RR.Configuration.1(
 'CHAR:DNS-DOMAIN' domain_name,
 'CHAR:DNS-HOST' primary_server,
 *'FLAG:FORCE-ZONE-TX' z_tx_flag,
 (0,6) 'CHAR:DNS-NAME','INTEGER:DNS-RR-
TYPE','INTEGER:DNS-RR-CLASS',
'TIME:DNS', 'INTEGER:DNS-RR-DATA-LENGTH', 'DATA:DNS-RR'
 rr (name, type, class, ttl, length, data);
 'LIST:RR-REPORT' rr_report 'RATE'/'DWR'/'BIR',
 *'LIST:ZONE-TX-REPORT' z_tx_report 'RATE'/'DWR'/'BIR')
{ Function to configure a list of generic resource records different than SOA.
If indicated, there'll be an automatic forced zone transfer. The actualization of
the serial number is also automatic [...] }
{ The first results parameter is a list of status records indicating the result of
each resource record configured.
  The second results parameter is a list of status records indicating the result
of each zone transfer
 [...] }
{ The reserved execution parameters should be used in a normal
way [...] }
{ [...] complete description of code behavior [...] }
```

6. REFERENCES

[1] J. Case, R. Mundy, D. Partain, B. Stewart, *Introduction to Version 3 of the Internet-standard Network Management Framework*, RFC 2570, April 1999.

[2] D. Harrington, R. Presuhn, B. Wijnen, *An Architecture for Describing SNMP Management Frameworks*, October 2001.

[3] J. Case, D. Harrington, R. Presuhn, B. Wijnen, *Message Processing and Dispatching for the Simple Network Management Protocol, Version 3*, October 2001.

[4] D. Levi, P. Meyer, B. Stewart, *SNMP Applications*, RFC 2573, November 2001.

[5] U. Blumenthal, B. Wijnen, *The User-Based Security Model (USM) for Version 3 of the SNMP*, RFC 2574, November 2001.

[6] U. Blumenthal, B. Wijnen, *The View-Based Access Control Model (VACM) for Version 3 of the SNMP*, RFC 2575, November 2001.

[7] K. McCloghrie, R. Presuhn, J. Case, M. Rose, S. Waldbusser, *Transport Mappings for the Simple Network Management Protocol*, October 2001.

[8] K. McCloghrie, R. Presuhn, J. Case, M. Rose, S. Waldbusser, *Version 2 of the Protocol Operations for the Simple Network Management Protocol*, October 2001.

[9] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, M. Rose, S. Waldbusser, *Textual Conventions for SMIv2*, RFC 2579, April 1999.

[10] S. Waldbusser, Remote *Network Monitoring Management Information Base, Version 2 using SMIv2*, RFC 2021, January 1997.

[11] R. Austein, J. Saperia, *DNS Resolver MIB Extensions*, RFC 1612, May 1994.

[12] R. Austein, J. Saperia, *DNS Server MIB Extensions*, RFC 1611, May 1994.

[13] D. Levi, J. Schoenwælder, *Definitions of Managed Objects for Scheduling Management Operations*, RFC 2591, May 1999.

[14] D. Levi, J. Schoenwaelder, *Definitions of Managed Objects for the Delegation of Management Scripts*, RFC 3165, August 2001.

[15] D. Levi, J. Schoenwaelder, *Event MIB*, RFC 2981, August 2000.

[16] B. Stewart, R. Kavasseri - Editor, *Notification Log MIB*, RFC 3014, November 2000.

[17] B. Stewart, R. Kavasseri - Editor, *Distributed Management Expression MIB*, RFC 2982, October 2000.

[18] B. Dias, *Gestão de Redes Internet*, PAPCC, Universidade do Minho, 1996.

[19] A. Brites, P. Simões, P. Leitão, E. Monteiro, F. Fernandes, *A High-Level Notation For The Specification Of Network Managament Applications*, Proc. INET'94/JENC95.

[20] T. Saydem, T. Magedanz, *From Networks and Network Management into Service and Service Management - Guest Editorial*, pp. 345-348, Journal of Network and Systems Management, Vol.4 – N.4, December 1996.

[21] F. Stamatelopoulos, T. Chiotis, B. Maglaris, *A Scalable, Platform-Based Architecture for Multiple Domain Network Management*, National Technical University of Athens.

[22] G. Goldszmidt, Y. Yemini, *Distributed Management by Delegation*, Proc. 15th Int. Conf. On Distributed Computing Systems, June 1995.

[23] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, *Decentralizing Control and Intelligence in Network Management*, Proc. 4th Int. Symposium on Integrated Network Management, May 1995.

[24] M. Siegl, G. Trausmuth, *Hierarchical Network Management – A concept and its Prototype in SNMPv2*, 1996.

[25] Aiko Pras, *Network Management Architectures*, PhD Thesis, February 1995.

[26] N. Freed, S. Kille, *Network Services Monitoring MIB*, RFC 2788, March 2000.