Reusable Architecture for Data-Centric Network Management Systems

Rajeev Gopal, David Whitefield
Network Management Group
Hughes Network Systems
11717 Exploration Lane, Germantown, Maryland 20876
USA
{rgopal, dwhitefield}@hns.com

Abstract

This paper describes a reusable architecture that has been used to develop network management systems for data-centric satellite-based networks. The satellite-transport characteristics and networks impose unique constraints on the management systems. Under the data-centric scheme, the manager keeps the network configuration management information that is automatically downloaded by the agents residing on the managed devices. An exclusive use of standards-based off-the-shelf network management platforms does not directly address all requirements for data-centric management of large satellite networks. The architecture described here is based on an integration scheme, called DBMS-based auto discovery. This scheme keeps relational DBMS and NM platforms in synchronization. The integration scheme preserves the desirable development and operational features of both the NM platform and the DBMS and leads to significant savings in development effort and time.

Keywords

Satellite networks, auto-discovery, case studies and experience, standards and frameworks

1. Introduction

Satellites are now playing an ever-increasing role in information distribution and in providing the infrastructure for large-scale multi-media networks. Originating as a broadcast mechanism for television networks and intercontinental telephony trunk lines, satellite-based networks are now being widely deployed for interactive data, voice, video, and Internet applications. Satellites use space, which is a shared medium, to transmit information utilizing time and frequency division multiplexing. Frequency reuse is further enhanced using spot-beams so that a cellular-like architecture can be used with the satellite networks. This mechanism is the corner stone for the next generation of satellite networks. The full potential of satellite networks can be realized by providing an intelligent operations and maintenance

system that allows responsive configuration control for optimal utilization of precious resources in the time and frequency domains.

The power and usefulness of satellite networks, similar to their terrestrial counterparts, require a comprehensive network management system that can be used for operations and maintenance. Traditionally, the Network Management Systems (NMS) for satellite networks have been developed in a proprietary fashion with some use of computing (and not network management) standards and related off-the-shelf software. With the growing emergence of satellite, terrestrial, and hybrid networks, interoperability and standards compliance is rapidly becoming a requirement. However, satellite networks have unique management requirements and not all of them are well aligned with the standardization efforts and the network management platforms that are becoming available. Instead of having a false sense of security that software vendors will provide a complete solution, it is important to identify the deficiencies so that unique solutions can be developed. It is seen that many of the desired requirements are already available in the form of Database Management Systems. An integration of the network management platforms and the DBMS can address many more requirements, as discussed in this paper.

The same physical satellite network can provide several logical virtual networks each tuned to specific requirements based on the type and volume of traffic. Because of centralized maintenance, all pre-configuration and provisioning activities have to be well supported. This requires sophisticated version management support for configuration data so that multiple versions of correct, complete, and consistent network configuration can be maintained within the NMS while operators are experimenting with new scenarios or incrementally adjusting configurations. This centralized handling of configuration data, termed data-centric management, is a key requirement for satellite networks because of the large and complex relationships and configuration rules imposed by satellite networks. It is essential to eliminate the possibility that a single mis-configuration leads to a large non-localized impact on the network.

From a functional point of view, satellite networks share common high-level network management requirements with their terrestrial counter-parts. On the other hand, traditional satellite networks are flat and typically cover large geographical area. Often, their deployment is in remote locations with no local maintenance or upgrade facility. Management links have limited bandwidth availability that requires efficient data coding schemes. To save power and reduce recurring cost for remote stations (thousands to millions in number), processing, memory, and storage capability has severe limitations. This requires that remote device configuration information along with executable software be downloaded from the manager in charge of remote devices scattered over a wide geographic area since there is not enough storage within a device. This configuration management aspect of the satellite networks is described in this paper.

The paper is organized as follows. Section 2 summarizes key facets of network management and the role of network management standards namely, SNMP and CMIP. Section 3 details a data-centric approach for configuration management that aims at handling large, flat networks with (persistent) memory-less managed

.

devices. Section 4 summarizes the requirements. Section 5 presents an overview of the DBMS-based auto-discovery scheme that serves as the integration mechanism for DBMS-resident configuration management and the network management platforms. Section 6 summarizes the application of this reusable architecture in multiple network management systems with cost-benefit analysis. Section 7 summarizes the contributions of this paper and provides a roadmap for future work.

2. Standards-Based Network Management

Network Management as a discipline has gained some maturity and acceptance [5] with the development of key standards such as SNMP [6] and CMIP [7]. The implementation of network management systems also leverages several computing standards such as C++, SQL, HTML, Java, etc. However, the SNMP family is mostly driven by the management of commonly used terrestrial devices such as routers, bridges, workstations, etc. The CMIP family, originating from OSI and now including TMN [4], is aimed at the telecommunications world with complex and large networks. Both of these standards address the management information modeling with Management Information Bases (MIBs) and protocols to exchange information between a manager and the agents residing on the managed devices.

SNMP uses trap-driven polling while CMIP uses notifications (traps) as the primary means of collecting status and event information from agents. Although SNMP (v1) is easy to implement, its scalability, security, bulk data transfer, and manager-to-manager communication capabilities are limited. CMIP addresses these areas, but at the cost of complex and costly implementation. This cost can be justified at a small number of managers but not on the agent side where large number of minimally configured devices are connected to the manager with low bandwidth links. One common deficiency in both SNMP and CMIP family is a lack of behavioral modeling. Both MIBs provide only a place-holder for textual description of device behavior.

In both SNMP and CMIP network management, the agent is expected to know and store the device configuration. The manager can interrogate an agent to get the current configuration or even instruct the agent to modify some attributes. This implies that the managed device has a way of storing configuration persistently using either disk drives or non-volatile memory in order to maintain current configurations. For large satellite networks this model does not work since the remote devices are designed to carry mostly traffic and the cost model does not allow for data persistence support for large number of configuration attributes. In such large, flat networks, the manager keeps the correct and current configuration for each managed device. When a device boots, it requests its configuration (and executable software) from the manager. When the device's configuration or software changes at the manager, the agent is immediately notified of the change to reconcile its processing. This unique feature, termed data-centric configuration management, is not directly supported in either the SNMP or the CMIP model and requires a custom solution.

3. Configuration Management

A major issue in network management has been where to store configuration data. Many approaches are possible and each has its benefits and drawbacks. It is generally accepted that the device has the *current* NM data, but it is unclear who should hold the *desired* NM data. This simple distinction has many repercussions in the entire architecture of an NMS. This is because most configuration management data access is for read-only purposes and satellite bandwidth is so limited.

Satellite network management is simplified if the network manager provides configuration data persistence for the managed devices. This imposes a database-centric architecture for network management systems. This approach is discussed first. For completeness, two other approaches are also briefly described. Several advantages of the database-centric approach in the areas of consistency checking and pre-configuration are also described.

3.1 Database-Centric

A database-centric approach, shown in Figure 1, is where the network manager stores the *desired* configuration. If the agent does not posses the current desired configuration, this need is identified and the manager ensures that configuration is reconciled with the agent with a downline load mechanism. With this approach, the manager has to include a comprehensive data management facility. This is typically provided by a DBMS where all configuration data is stored and indexed. The use of a DBMS facilitates the implementation of the configuration management applications directly on top of the DBMS. The operators use these applications to browse and change configuration data, stored locally in the manager's DBMS, for the network devices. The NMS ensures that the changes are automatically provided to the agent.

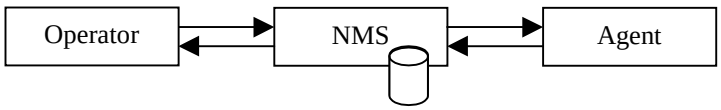

Figure 1: Data-centric network management

The configuration data is converted to files that are then sent to the agent by an entity called reconciler that resides within the NMS. Reconciler publishes all changes in configuration data to the potentially interested agents. An intelligent agent can request and pull a specific piece of data, identified by version numbers or time-stamps The configuration information is sent in files with minimal coding overhead so that the agents can quickly receive and parse the configuration data sent by the manager. Under this scheme, the NMS is not required to poll and push the configuration information (parameters or software files). This kind of intelligence within the agents ensures that the NMS does not have to keep associated state information for disseminating configuration information to large number of agents.

.

3.2 Agent-Centric

An agent-centric approach, shown in Figure 2, is where the agent stores the *current* and *desired* NM data, using Non Volatile RAM or even disk. If an operator-level application needs to show configuration data, the manager has to retrieve this information from the agent. The manager may temporarily cache this information, but it is the agent that has the most current data. The data transfer from the agent to the manager could be time consuming because of the limited bandwidth and satellite delay between a manager and the agents that are not always geographically co-located. If the agent has a persistence problem, such as NV RAM failure, then all of its configuration information will be lost and must be reentered from an external source.

Almost every major network management platform assumes an agent-centric approach with open protocols such as SNMP and CMIP. Thus, third party and satellite devices conforming to agent-centric approach can be operated easily with off-the-shelf platforms supporting open protocols. Some of these platforms may provide persistence, but it is typically used for local objects residing on the network management host such as event logs.

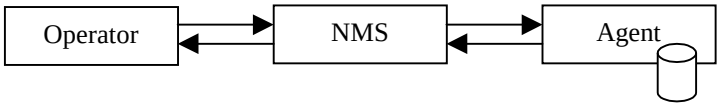

Figure 2: Agent-centric network management

3.3 Manager and Agent Stash

Another approach is to store the desired NM data in a private *stash* largely independent of the manager or the agent. This allows for slightly better redundancy if either the manager or agent fails, but introduces consistency problems for the NM data. Either the manager or the agent can write to this stash and either of them can read from the stash when this backup data is needed. Stashing is just for illustration purposes and is less likely to be used in satellite products since stashing is not transparent. Each application is required to implement its own reconciliation scheme. Consistency between stashes and versions of stashes used to acquire desired NM data for agents is not easily enforceable

3.4 Consistency Checking

Consistency checking is an important issue for any approach to configuration data persistence. In order to prevent an operator from mis-configuring a complex network, the system provides double checks. These checks are performed at the manager before any data is sent to the agent. Many third-party devices that use SNMP let the agent do all consistency checking. In that case, the manager resends requests repeatedly over the network until a positive response is received from the agent.

It is not easy to ensure consistency across several agents because of the difficulty in implementing transaction and locking control in such a distributed scenario. Consistency checks, especially that involve more than one object, are best implemented at the server level so that client applications are not expected to make these checks individually. This minimizes redundant code and can also improve performance since all data is available at the server level on the manager.

A data-centric approach where the manager provides agent data persistence is found to be most suitable for ensuring consistency. In this approach it is possible to fully leverage the distributed locking, triggers, and transaction features of modern databases. All configuration-related consistency constraints and transformations can be coded within the DBMS as internal procedures. These software modules are invoked automatically using the DBMS triggering mechanism when an object is created, deleted, or modified. A trigger can thus reliably run all checks and also make related secondary changes in the same or other tables transparent to the various management applications using the DBMS.

3.5 Pre-Configuration

The database-centric approach is also needed for providing pre-configuration support. Often a network operator wishes to pre-configure the network before the actual hardware arrives or is installed. This allows the network operator to decide when they wish to perform all of the data entry necessary to configure a network rather than doing it only after the hardware has been installed and while it is turned on. This also allows the network configuration to be entered in the factory. This value-added feature is only possible if operators can configure a network without having to interact with a real network agent.

3.6 Multiple Versions

Most networks continue to evolve with on-going change, addition, and deletion activities. Multiple operators may be involved in a large network dealing with the various phases of planning, installation, test, and commissioning activities. A data-centric manager requires a version management facility where the database can store configuration information related to these specific stages and provide a robust environment for defining and deploying configuration data. A DBMS simplifies the management of multi-version configuration data by using a suitable granularity at the database (network), table (device type), or the row (specific device) levels.

4. Requirements

The requirements for satellite network management networks are presented under three categories: business, architectural, and functional.

4.1 Business Requirements

Some next generation network management system (NMS) requirements are driven by market forces including sharing of common reusable software by multiple product lines, low cost development, and standards-compliance.

.

4.2 Architectural Requirements

The architectural requirements, shown in Table 1, also include those derived from major business requirements. For each architectural requirement, we also show its implementation with the following: Database Management System (DBMS) and Network Management Platform (NMP). Note that all network management platforms may not necessarily meet all of these requirements.

Table 1: Architectural requirements for network management systems

Feature	*Description*	*Implmented With*
Proprietary Protocol Translation	The NMS should be able to manage devices utilizing proprietary protocols for configuration, events, command/response, test, diagnostics, and performance data.	NMP
Data-Centric Management	The NMS should be able to provide data-centric management where configuration data is kept in DBMS facilitating centralized configuration.	DBMS
Scalable Architecture	The NMS should be scalable to handle tens of thousands and potentially millions of geographically distributed managed devices	DBMS and NMP
Distributed Software Architecture	GUI, server and network interface applications to run on different computers over IP protocols.	DBMS and NMP
Rapid GUI Generation	Operators should be able to configure devices using easy-to-use GUI applications. Their development and modification should be low cost.	DBMS tools
Object-Oriented APIs	The APIs should allow the use of object-oriented implementation tools such as C++ and Java.	DBMS and NMP
Multi-User Operation	Multiple operators should be able to use, without jeopardizing data integrity, the same NMS and share common network configuration and monitoring data.	DBMS
Transaction Processing	The network configuration should be created and modified under well-defined transactions in a multi-user scenario.	DBMS
Version Management	Multiple versions of network configuration should be available to the operator.	DBMS

Pre-Configuration	Operator should be able to pre-configure non-existent devices and store the configuration until the devices become on-line and fetch their configuration from the NMS.	DBMS

4.3 Functional Requirements

The following functional requirements coupled with the business and architectural requirements listed earlier provide the necessary structure to analyze our approach. The functional requirements are listed under five areas. There is a comment at the end of each area that summarizes whether the implementation is database or network management platform based.

1. **Fault Management** - Encompasses fault detection, isolation and the correction of abnormal operation of the network. Faults cause systems to fail to meet their operational objectives and they may be persistent or transient. This involves collection of events and alarms, suitable applications for viewing and managing these logs and iconic displays, and performing actions.

   The fault management requirements are met with the network management platform.
2. **Performance Management** - Monitors the behavior of resources in the environment and the effectiveness of communication activities to be evaluated. This involves collection of performance data (notification and/or polling) and viewing applications.

   The performance management requirements are only partially met with off-the-shelf network management platform features. Historical performance data (logging and visualization) can benefit from DBMS.
3. **Configuration Management** - Identifies, exercises control over, collects data from and provides data to systems for the purpose of preparing for, initializing, starting providing for the continuous operation of, and terminating interconnection services. This includes auto-discovery, SNMP and CMIP MIB browsers and rules engines for data-centric configuration consistency and downline load generation rules. Also important is the modeling scheme for configuration data.

   Except for SNMP and MIB browsing, these requirements are not suitable for the network management platform and warrant the use of DBMS.
4. **Accounting Management** - Enables charges to be established for the use of resources in the environment, and for costs to be identified for the use of those resources. This includes collection and processing of accounting data.

   The accounting management information is collected using the platform functionality (similar to performance and fault) for file transfers. Historical data is stored in the DBMS and visualization applications are easily built on top of the DBMS.
5. **Security Management** - Supports the application of security policies by means of functions including the creation, deletion and control of security services and

.

mechanisms, the distribution of security-relevant information, and the reporting of security-relevant events. This includes operator access control and managed element security.

Object-level security, especially for configuration, requires granularity that is typically missing from SNMP (v1) platforms. CMIP platforms do provide security. Configuration-based security is provided by the DBMS.

6. **Navigation –** Navigation provides an operator interface (often with geographic map features) to view the network, its components, and current status.

The navigation features are provided by the network management platform.

4.4 Discussion of the DBMS Requirements

The requirements can be grouped into three broad categories. The first is the set of functions that are easily mapped to the relational DBMS. The second set of functions can be hosted by network management platforms. The final set of remaining functions requires in-house software development.

There is an advantage in using relational DBMS as opposed to object-oriented DBMS for network management applications. Although functionally rich (with seamless mapping to object-oriented language primitives), object-oriented DBMS is still not as commonly used as its relational counterpart. With the availability of layering software that facilitate the use of object-oriented language, the relational DBMS now provide best of the both worlds, as enumerated below:

- Relational DBMS, as opposed to object-oriented DBMS, is well entrenched in modern computing. It offers a wide variety of interfaces (SQL, C++, HTML, Java) and mature features such as third-party applications, scalability, stability, replication, distribution, etc.
- Relational DBMS and associated rapid generation tools are increasingly useful in developing GUI applications and web interfaces.
- Relational DBMS provides engines for configuration and downline load rules and is available on all major hardware/software platforms.

5. DBMS-Based Auto-Discovery

Many requirements listed above for complex multi-protocol satellite devices can easily be met with network management platforms. However, most of the device specific work is related to the implementation of configuration applications ensuring correct and consistent behavior. The need for storing the entire network configuration in a DBMS is aligned with transaction, multi-user, and pre-configuration requirements. It is also clear that substantial development is with respect to information stored in the DBMS. The major objective for a suitable integration scheme (reusable architecture) is to somehow keep both the platform and the DBMS in synchronization so that various applications can be seamlessly developed and used without precluding the use of the desirable features of either the platform or the DBMS. Two additional possibilities are discussed first, followed by the scheme actually used in the reusable architecture.

5.1 NM Platform API with Embedded DBMS

One possible implementation could be that the DBMS is embedded within the network management platform. All custom applications are developed using the API provided by the platform. The model for the network (full FCAPS) is located in the platform. Applications that require persistence ultimately use the DBMS features as seen through the platform API. Though elegant, this scheme has two major disadvantages. First, one cannot benefit from the large selection of application development tools that are available for the DBMS (and not the platform). Second, essential features such as transaction management that are central part of the DBMS are likely to be missing from the platform. Any DBMS embedding essentially precludes the use of these desirable features because of current inadequate NMP APIs.

5.2 DBMS API with Embedded Platform

Taking the reverse approach where the DBMS is layered above the platform introduces its own problems. In this scheme the DBMS hosts the network model (all FCAPS) and the platform becomes a client for the DBMS. The DBMS APIs are generally available for data use and are less likely to have the message definition and processing capabilities that are required by fault and performance management applications. It becomes easier to develop GUI applications with the rapid generators, but these applications will not be able to benefit from the message modeling (such GDMO) besides suffering from poor message handling performance. Moreover, many off-the-shelf applications such as alarm manager and event manager bundled with the platform may need to be developed again on top of the database. Command-response types of management protocols are problematic to implement in a transaction-based DBMS.

5.3 DBMS and NM Platform at the Same Level

By keeping both the DBMS and the NM platform at the same level, as shown in Figure 3, we can benefit from their respective attractive features. The configuration applications can be developed on top of the DBMS, benefiting from the availability of rapid generator tools. The respective APIs are used to develop custom applications. The first challenge is to make sure that the NM platform and the DBMS are synchronized with respect to the network model. Secondly, the user interface should seamlessly provide access to all NM applications irrespective of how they are hosted.

Architecturally, the NMS can be divided into three horizontal layers. The topmost layer includes all user-level applications that are used by the operator. Currently GUIs use Motif, but this layer is fast migrating to the Web-based standards (HTML and Java). A visual display of entire network in terms of device containment hierarchy is provided by the platform viewer application. The second layer is the server layer which contains the following: models for network components, network containment hierarchy, APIs for user-applications, distribution function, routing function, various logging services, APIs for the network interface layer, etc. The bottom-most layer provides the interface to the network. It can

.

directly support the standard protocols such as SNMP and CMIP or provide libraries for converting legacy protocols. The integration of the DBMS and NM should be transparent from all user applications.

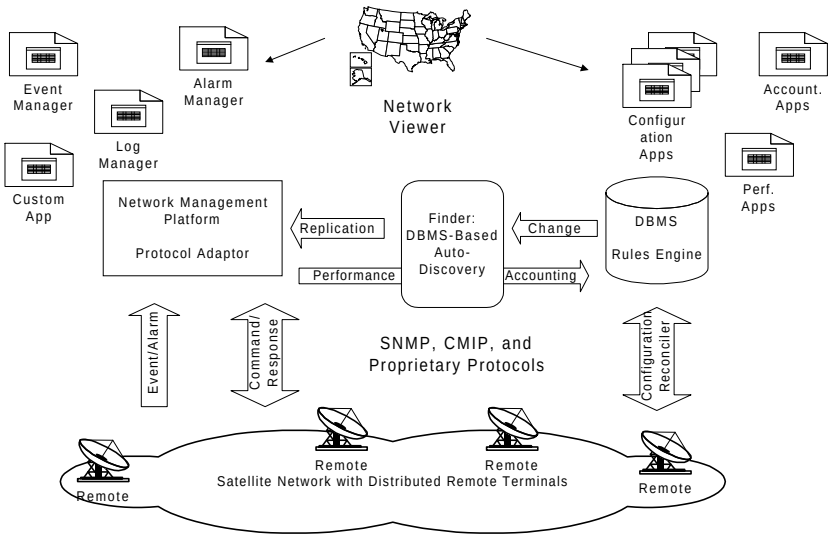

Figure 3: Reusable architecture for network management systems

The detailed configuration data for each device, represented by an icon on the viewer, is stored in the DBMS. The containment hierarchy should be in sync with the database representation. The configuration applications are developed using DBMS tools (rapid application generators). Several configuration constraints [1] have to be addressed to ensure correct and consistent configuration. The configuration rules are coded in a high-level DBMS internal procedure language. When a GUI (or a command line interface for that matter) configuration application creates or adds a device in the database, then the database triggers get activated and eventually the containment hierarchy gets replicated to the NM platform. Thus, the network management platform *automatically discovers* the managed devices through the DBMS (that is storing configuration data for the managed network). Similar discovery in the SNMP world requires that the manager poll all managed devices in the network.

The platform itself collects traps and command-response information and changes the icon color based on the status. The operator can launch any fault, performance, or configuration application from menus associated with the icon via a common launching procedure. The configuration changes are converted to the download files using rules that are also coded in the same DBMS internal procedure language. Both configuration checks and download file generation thus share a common rules language and processing engine.

This scheme preserves the distribution and scalability features of the DBMS and network management platform. For larger networks it is possible to partition and

map the network into multiple instances of the platform and DBMS. Each DBMS instance synchronizes, using a dedicated Finder module, with its counterpart platform instance for the managed devices under its domain. The platform instances can perform command/response handling and event propagation within their hierarchy.

Another possibility where both the DBMS and the NMP are at the same level could have been that a management application uses two-phased commits. This scheme is not desirable since it exposes each application to two APIs (DBMS and platform) and also because there is a general lack of transaction support in NM platforms.

6. Application of Reusable Architecture

Two network management systems have already been developed using this reusable architecture. Many more are under various development stages.

6.1 VSAT Telephony

A Very Small Aperture Terminal (VSAT) telephony network management system was the first application of the reusable architecture. The network is based on Geo-synchronous satellite and can include thousands of remote VSATs. The network management system monitors the remotes for their status and performance characteristics. The configuration data is stored in the database.

The Finder module replicates the containment hierarchy to the network management platform (Sun Solstice Enterprise Manager). Fault management applications are off-the-shelf, while custom performance application has been developed on the network management platform. A web-based version of these applications is also available [2].

6.2 Mobile Satellite Network

A network management system for managing satellite base stations of a medium-earth-orbit (MEO) satellite network has been developed. The network uses MEO satellites for providing global mobile telephony and related applications. The satellite base station is a key component that provides the functionality of cellular Base Station Controller (BSC) and Base Transmission Station (BTS). In addition, it includes a resource management system that dynamically manages the frequency and time slots for multi spot-beam satellites.

The NMS manages a variety of equipment using proprietary (Message Protocol Adapters) and SNMP protocols. The software and device configuration is automatically down loaded from the NMS. Accounting data is collected using FTP. A viewer provides a graphical display of the network hierarchy with color icons representing the status of respective devices. All applications can be started using context-sensitive menus associated with the icons.

6.3 Cost-Benefit Analysis

By using off-the-shelf software (DBMS and platform), the amount of custom software has been reduced by about 50%, leading to substantial savings in multi

.

person-year projects. The reduction in software size was estimated by comparing the NMS developed using this approach with a similar NMS that was developed earlier using proprietary techniques. The biggest savings were observed because of the applications that are bundled with the NM platform and with the use of configuration and download rules that could now be coded within the DBMS.

The use of off-the-shelf software introduces recurring software licensing costs for each deployed network. This cost, however, is a small fraction of the development labor costs that are saved with the off-the-shelf software. The number of NMS nodes in a satellite network is rather small compared to the nodes that carry traffic. Thus the recurring costs because of the use off-the-shelf software in the NMS is manageable.

The benefits for the various phases are summarized below. All these savings can ultimately be traced to a simple application of the systematic reuse initiative. A reusable architecture facilitates more functionality with less software!

- **Marketing**: The use of an NM platform with SNMP and CMIP compliance provides an additional marketing edge to the networks.
- **Prototyping**: New networks typically take a few weeks for a demonstrable prototype with significant fault, performance and configuration management features. This used to require several months where large number of source code files had to be modified.
- **Time-to-Market**: A data-driven approach and significantly less software development has allowed us to deploy a networking product faster.
- **Testing**: Testing is reduced since several application such as alarm manager, event manager, and viewer with features such as rules engine, transactions, multi-user support, caching, replication etc. are off-the-shelf.
- **Maintenance**: An architecture-driven approach allows a common maintenance of product-independent features of the NMS.

7. Conclusions

The network management systems for satellite networks can directly leverage computing standards in the following areas: operating system, GUI, databases, inter-process communication, object architectures, and programming languages. These managers can use advanced network management platforms as well. However, the data-centric configuration management requirements require the use of DBMS. A replication-oriented integration of the platform with a DBMS allows leveraging both sets of technologies for significant cost and time savings. We have validated our approach by accommodating two network management platforms (Sun Solstice Enterprise Manager [3] and HP OpenView [8]). The DBMS selection has been Oracle, but with our use of the CORBA and SQL standards for the integration software, other DBMSs can easily be accommodated.

It is likely that gradually the platform vendors may appreciate the importance of lean protocols, data-centric configuration management, and versioning, and ultimately may even incorporate them in the platforms. Recent initiatives undertaken by the IP networking equipment manufacturers, termed Directory Enabled Networks

(DEN), have some similarities to data-centric management and are promising under this context. Some networking and software vendors have shown interest and are slowly migrating towards a scenario where configuration data is obtained from centralized places. New protocols are being developed so that management systems can use such directory-based data. A wide spread use and subsequent incorporation into network management platforms may still take some time.

For foreseeable future custom satellite network management systems will continue to be developed because of the deficiencies in the NMS platforms. However, the development of such systems can benefit from the availability of off-the-shelf platforms and other computing technologies such as DBMS. Besides the use of DBMS, other areas such as visualization and intelligent processing look more promising since the ongoing improvements in computing technology can also be leveraged. General purpose visualization tools, Web technologies (Java, VRML, XML, HTML, HTTP) for universal GUI and remote access, CORBA services, and expert system toolkits, etc. are available today for their incorporation into network management systems. An integration of off-the-shelf software instead of development from scratch is in itself a promising start with significant cost and time-to-market advantages.

Acknowledgments

This paper is based on the work done by the past and current members of the Software Technology department and the various product lines of HNS.

References

[1] Goli, S. K., Haritsa, J., Roussopoulas, N., ICON: A system for implementing constraints in object-based networks, Integrated Network Management IV, Chapman and Hall, 1995.

[2] Gopal, R., Web interface for network management systems, Network Management Forum Bulletin, Fall 1996.

[3] Huntington-Lee, J., Terplan, K., Gibson, J., HP Open View: A manager's guide, McGraw Hill, 1997.

[4] ISO/ITU, Principles for a telecommunications management network, International Telecommunication Union Recommendation M.3010, 1992.

[5] OMNIpoint documents (Overview, procurement guide, development guide, conformance requirements), Network Management Forum, 1994.

[6] Rose, M., The Simple Book: An introduction to Internet management, Prentice Hall, Englewood Cliffs, NJ, 1994, 2nd edition.

[7] Stallings, W., SNMP, SNMPv2, and CMIP: The practical guide to network-management standards, Addison-Wesley, Reading, Massachusetts, 1993.

[8] Sun Solstice Enterprise Manager Overview, SunSoft, 1996.

.