Abstract

This paper presents an Event-Driven Service-Oriented Architecture (EDSOA) for an agile and scalable Network Management System (NMS). Dynamic business requirements, interoperability with Operations Support Systems implemented on disparate platforms, and requirements to interact with Network Elements using a wide range of protocols are the current market drivers in the NMS space. This has necessitated the need for an agile, flexible and loosely coupled Network Management Framework. We propose the EDSOA approach which will fulfill these requirements and also provide for a gradual evolution of the network management platform with the addition of management services as the need arises for them.

1. Introduction

The last two decades have seen a tremendous burst in technology for the field of telecommunications. The network is evolving and is now being designed for unified access, transports and voice switching. Additionally the network elements are now geared towards ferrying high speed data and video. Further with the continuous emergence of new technologies like Voice over Internet Protocol (VoIP), the underlying network elements keep changing. The high degree of complexity accompanying the network element technology necessitates a rich network management system to harness and control the usage of technology while hiding its complexity. As operators begin to add new networks and expand existing networks to support new technologies and products, the lack of scalable, functionally rich NMS systems is starting to be felt all the more.

These problems affect both day-to-day network operation management (higher staffing and Operation-Expenditure requirements) and strategic network growth planning. There exists an urgent need for a flexible, intelligent and scalable NMS architecture that can increase automation of network operation, while being able to provide long-term planning inputs. Another factor influencing NMS architectures is the clear trend of mergers and acquisitions among the key vendors. A key input into the successful integration of these various products into an NMS suite is the ease of integration, a factor which is not very well handled by the traditional hierarchical NMS architecture. These growing demands of interoperability, flexibility and scalability fuel the need for an architectural framework that will address these issues.

2. Network Management System Overview

2.1. Elements of an NMS

Elements of a NMS can broadly be categorized into 5 areas, each of which is described below.

1. Fault Management - This consists of all functions of the NMS related to network problems or faults that occur among the various element of the Network being managed. These functionalities include detection, notification, inference gathering/correlations, and possible automated fixing of these faults in the network.

2. Configuration Management - This encompasses functionality that deals with the configuration information associated with the network elements and their inter-linkages. This could include operations such as monitoring the configuration state of a particular resource, providing interfaces for authorized operators to make configuration changes, logging and auditing of the configuration changes.

3. Accounting Management - This deals with the measurement of utilization of various network parameters associated with specific users or user
groups. This monitoring and analysis of the data enables the operator to be able to charge the customer and to understand capacity utilization of the network itself.

4. Performance Management - This deals with the measurement, aggregation and analysis and reporting of various aspects of network performance. The analysis of this data enables reactive and preventive maintenance.

5. Security Management - This deals with access to all the resources within the network using prescribed rules and guidelines so that it can prevent unauthorized access. This encompasses authentication and authorization of users/groups that verifies the user credentials and permits operations on the various resources.

A typical NMS system has component(s) that provide functionality in the above areas, while drawing inputs from the network elements being managed as well as other functional components within the NMS.

2.2. Traditional NMS Architecture

The following diagram depicts the various functional, modular components that are a part of a traditional hierarchical NMS architecture.

![Figure 1. Traditional NMS Architecture](image)

The various modules which typically form the NMS are:

1. Framework Components: These provide basic platform functionality, including persistence, logging, and security (authentication, authorization and encryption).

2. Mediation Components: These components implement protocol specific interface functionality on the “Southbound” interfaces to communicate with the network elements. The list of components depicted in the diagram are a sample, and the specific protocols implemented will change based on the NMS requirements. All access to the network elements happens via these components.

3. Topology Components: These provide functions related to topology and configuration of network elements. These include discovering network elements, making this information available to other components, as well as adding/modifying/deleting the network element configuration.

4. Fault Components: Components in the module relate to fault collection, aggregation, correlation and reporting. Components here can interface with topology components for object and resource information.

5. Performance Components: These components collect, analyze, and report Performance Metrics from the network elements. The analysis components in these modules frequently interact with the topology components for resource information, as well as with the Fault Components for reporting of discrepancies and other performance issues.

6. Northbound Interface Components: These components implement protocol specific interface functionality on the “Northbound” interfaces to Operations Support Systems (OSS) applications. The list of components illustrated in Figure 1 are a sample, and the specific protocols implemented would change based on the OSS requirements.

7. GUI/HMI Components: These components provide the user-interface into the framework. They consist of user interaction screens that enable the user to query as well as perform operations on the various modules in the Network Management System.

The traditional NMS architecture depicted above has a hierarchical structure with information and process flows typically involving a GUI/Northbound Interface component (User input), a Core functionality component (one of Topology, Fault or Performance) and a Mediation component.

3. Limitations of the Traditional NMS Architecture

The traditional NMS architecture as depicted in Figure 1 places the described modules in a hierarchical order. This hierarchy represents a typically network-centric view of management with information flowing up from the network elements via a set of mediation and data transformation components. However, there are multiple limitations of this architecture which are described below.

1. Ever-expanding management requirements from service providers as well as growing complexity in the
network itself, has given rise to scenarios where the information and process flows in an NMS are much more complex and involve multiple components in the same hierarchical layer. A typical example of this is a “Service Correlation” module (described in Section 6.2.1) that is almost a mandatory component for a modern NMS. In a typical Service Correlation process flow there are multiple functional components across Fault, Performance and Topology domains that need to provide/consume information in order to achieve the end result (which is the extraction of customer impact of network faults). The traditional hierarchical NMS architecture does not lend itself to easy modeling of these flows. In order to facilitate this, workarounds have been put in place for these scenarios typically involving the building of point-to-point interconnects between the various modules as required. These rigid workarounds have made the various logically separate NMS modules tightly coupled to each other thus increasing the cost of maintainability, reusability and extensibility.

2. Another area where current NMS architectures are weak is interoperability. The NMS domain has traditionally never had universally accepted standard communication protocols. This is even truer of internal communication between NMS components, where standards-based component interaction has not been the norm. With the current trend of mergers between NMS software products, this lack of standardization causes very costly integration overheads of product components. Due to the inflexible and rigid nature of the traditional hierarchical NMS architecture, the time and effort spent in integrating a new system component is extremely high.

3. With the advent of VoIP networks where the number of network elements that need to be managed increases exponentially, the scalability requirements from an NMS system have changed dramatically. There is a need for dynamic event collation, load balancing, coupled with near-real-time management of internal request to resources. These are not well addressed by the current frameworks and systems. The requirements of an adaptive framework to support huge volumes of requests based on priority is not inherently built into the architecture, and usually ends up being added at high development cost.

The proposed EDSOA architecture attempts to overcome these limitations.

4. Service-Oriented Architecture concepts

4.1 Service Requestor, Service Provider and Service Registry

Service-Oriented Architecture (SOA) is characterized by a clear demarcation of the concepts of service consumers and service providers. Service providers are entities that represent a business function and expose their functionalities via well defined and self-describing service interfaces. Service requestors interact with service providers based on their service interfaces and they locate the service providers from the service registries either statically or dynamically through a search mechanism. Availability of service factories obviates the need for plumbing work on the part of the service requestor to configure the service provider, resulting in loose coupling between the provider and the requestor. This enables seamless interchange and extension of service provider implementation code without affecting the service requestor implementation.

4.2 Enterprise Network Service Bus

An Enterprise Network Service Bus (ENSB) is a medium that provides the necessary plumbing required for event-driven interaction between various services implemented in disparate platforms such as .NET, J2EE, Web Services, C++ etc. This loose coupling enables seamless replacement and additions of the service implementation code as well as the service implementation platform and the remoting protocol. AN ENSB based on Message Broker pattern [3] and Message Bus pattern [2] facilitates this kind of loose coupling by providing an infrastructure for event-driven interaction between services thus removing the service consumer’s dependencies on the service provider’s implementation specifics.

![Figure 2. Elements of a Service Bus](image)

Message Brokers act as inter-connects between the services managed by the Service Bus and also between external applications and managed services. They consist of connectors, message transformers and message filters, which respectively externalize the protocol transformation logic, event data format transformation logic and event filtering logic from the implementation logic of the core service functionality.
This design makes the service implementation focused on business logic and hence better reusable.

Message Bus is the channel over which the events are transmitted between managed services and between external applications and the managed services. Examples include JMS, SOAP etc. The connectors are specific to each Message Bus.

The ENSB Configuration consists of settings for inbound/outbound connectors, inbound/outbound event transformers and inbound/outbound event routers for each of the managed services. This configuration information is read by the Service Bus at the start-up and all the managed services are wired up accordingly to communicate with each other.

4.3 Staged Event-Driven Architecture

A common approach to handling scalability is multi-threading, which handle concurrency by virtualizing the Operating System resources available to a thread. But this approach hides the fact that the system resources available are limited and shared, eventually resulting in a degradation of performance with increasing load. An alternate approach is to provide control over the resource management to the application itself, thereby allowing it to performance tune itself. Staged-Event-Driven-Architecture (SEDA) [1] provides a blue-print for such an approach. In SEDA, applications are constructed as a network of stages, each with an associated incoming event queue and a pool of event handler threads. Each stage represents a robust building block that may be conditioned to load by threshold or filtering its event queue. This allows applications to make informed scheduling and resource-management decisions as per varying request loads.

![Figure 3: A SEDA Stage][1]

5. Case for Service-Oriented Architecture based NMS Framework

This section aims at establishing the key design and architecture criteria required for a robust, scalable Network Management Framework. It correlates this with the standard characteristics of a SOA and makes the case for the suitability of SOA for designing a Network Management Framework.

1. Network Management systems are primarily event-driven software subsystems that are modeled by means of defining workflows. These workflows are triggered by events that are generated either from the “Southbound” side (i.e. the network elements or element managements systems) or from the “Northbound” side (i.e., from the NMS users or from the OSS systems that interface with the NMS). Additionally, the triggered workflows, as part of their definition, require that the various NMS modules such as Fault, Configuration and Performance support inter-module service requests. An Event-driven architecture such as SEDA with its support for inbuilt message/request queues lends itself very well to design of a Network Management Framework.

2. Another key requirement for an NMS is scalability and performance. Traditionally, these requirements have been addressed using two broad methods
   - Optimizing application design using high performance algorithms and data structures.
   - Deploying the application on higher end hardware, employ clustering and other hardware augmentation techniques. Though this is an effective solution, this is a brute force and a cost prohibitive approach.

While these are effective approaches to architect systems that manage small and medium sized managed networks, they face difficulties when required to scale up to manage large sized networks (upwards of 10000 network elements) in a cost-effective manner. This is partly due to the resource virtualization approach (described in Section 0), followed in traditional NMS architectures. A SEDA based design approach, optionally coupled with application clustering, will provide a more adaptable solution for dynamically scaling up, while at the same time, offering consistent performance.

3. The network management domain is dealing with ever-changing network element technologies. Due to this constant change, one of the key design requirements is to have a flexible and extensible set of external interfaces (“Northbound” and “Southbound”), as well as core business component interfaces that are loosely coupled to each other. Traditionally, this has been a failing of most NMS systems, and has resulted in extensive rework/enhancement requirements for supporting a new NE/OSS interface, or integrating a new business component from a different niche product. An SOA based architecture which inherently facilitates loose coupling and “pluggability” of new interfaces will significantly enhance the long-term

---

[1]: #/S3568.png

Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP’05)
0-7695-2452-4/05 $20.00 © 2005 IEEE

Authorized licensed use limited to: UNIV OF ALABAMA-TUSCALOOSA. Downloaded on April 13, 2009 at 18:07 from IEEE Xplore. Restrictions apply.
value of the implemented NMS. This extensibility will also allow the system implementers to follow an evolution-upgrade approach, rather than a big-bang approach.

4. In view of the fact that the basic modules of an NMS are common, we believe there is tremendous scope for implementation of a generic framework that can cater to the requirements of multiple equipment vendors as well as service providers. The ready availability of SOA/SEDA based components (e.g. Enterprise Service Buses which provide the basic plumbing required for an SOA-based system) is a significant advantage for quick time-to-market for an NMS framework based on SOA.

6. Service-Oriented Architecture based NMS Framework

The traditional NMS architecture depicted in the preceding section is transformed into an SOA-based framework, with services that offer well-defined and scalable interfaces to the NMS functionality. This section explains the features of the architecture which we believe will lead to an extremely robust, scalable, extensible and maintainable NMS implementation.

6.1. Key Features

The key features and advantages of the proposed architecture are as follows.

1. Reusability: The SOA based framework provides for a number of base components that would otherwise need to be built from scratch. The Service Bus provides the infrastructure for event-driven interaction between managed services.

2. Extensibility and Ease of Integration: The architecture allows for configurable way of specifying filters and interceptors for the services and also manages the life-cycle of the managed services. The Core NMS Service Bus manages the core NMS services such as Fault Service, Topology Service, Performance Service and Northbound Interface Service, while the Southbound Interface Bus (SIF Bus) manages the various mediation services. This enables the easy integration of new NMS services without any change in the existing services.

3. Scalability: Each of the managed services on the service bus is considered as a stage in SEDA. An event queue, an adaptive resource controller and event dispatcher threads are associated with each of them. The event queue may be an in-process (in-memory) queue or an out-of-process queue such as a JMS server. This addresses scalability issues in a very comprehensive manner.

4. Enhanced Security Paradigm: Another feature of the architecture is the integrated and extensible Security paradigm. The Security Service Router enables the framework to easily adapt to any external security framework mandated by the Service Provider.

6.2. Architecture Illustration

This section uses specific NMS use cases to illustrate the functioning of the proposed SOA-based NMS Framework.

6.2.1. Service Correlations. The concept of “Service Correlation” is becoming increasingly critical in Next Generation Network Management Systems. This essentially involves the linking of specific network events to the potential/actual service Impact of those Events. The information flows in this application is a good use case for the Event driven, SOA-based design. Requirements are as follows:

1. The Fault Management Component (FMC) receives an event reporting an outage of a specific optical circuit in a SDH (Synchronous Digital Hierarchy) network.

2. The FMC sends a query to the Topology Management component (TMC) to ask whether the circuit is “protected” (i.e. has a redundant link configured). Based on query result the severity of the fault can be ascertained.

3. The FMC queries the TMC for information on the “End-to-End” (E2E) paths that make use of this circuit.

4. The FMC queries the Performance Management component (PMC) to extract the performance metrics related to the availability/load on the network element on which the circuit was configured. This is required to validate that the circuit problem is not a core network
element issue which might lead to multiple failures and needs to be flagged.

5. Based on all this information, it then queries a Customer Relationship Management database which gives it the linkages of customers to the E2E paths, and the Quality-of-Service parameters of those customers.

6. After processing all this information, the FMC would issue a service alarm with the details of the service impact that has been calculated.

6.2.2. Integration of Authentication Authorization and Accounting Servers. While implementing an NMS framework at a Communications Service Provider (CSP), a significant aspect of the integration relates to the Security Framework deployed by the CSP. In a lot of cases, the CSP has already setup a security framework with Authentication Authorization and Accounting (AAA) servers, roles, responsibilities etc. In this context, it is necessary for the NMS to integrate seamlessly into the security framework instead of advocating something on its own. The key requirements are:

1. Ability to adapt to the access mechanism made available by the AAA infrastructure. This could be LDAP, RADIUS, proprietary etc.
2. Ability to define the security requirements of the NMS framework such as roles, access permissions etc.
3. Ability to map these NMS requirements to the existing security framework at the CSP without having to make changes to the NMS security paradigm. The framework needs to be independent of the actual AAA implementation with a mapping component in between.

Given this context, we see why the SOA-based NMS framework would fit in very well. The protocol specific services satisfy requirement 1, and the Security Gateway service provide for requirements 2 and 3.

7. Future Work

SEDA based design provides a cost-effective way of scaling up applications. In order to achieve further scalability a load-balanced cluster of SEDA based NMS may be used (Figure 5).

![Figure 5: Load Balanced cluster](image)

High system availability can be further achieved by providing transparent fail-over mechanism over the cluster. Future work will focus on enhancing the current architecture to support such a deployment plan.

8. Conclusion

In this paper we have provided a compelling case for an Event-driven and Service-Oriented Architecture for an NMS. The traditional NMS architecture addressed the problems of interoperability, extensibility and scalability through ad-hoc and point solutions. The proposed architecture provides an elegant solution for not only addressing the pain-points of the current generation NMS but also to cater to the requirements of the next generation NMS such as interoperability with legacy and specialized systems that have been inherited due to acquisitions and mergers, massive scalability and support for communication with new generation network elements.

10. References

