A Framework for the Management of Distributed Systems Based on SNMP

George Oikonomou, Theodore Apostolopoulos
Athens University of Economics and Business
76, Patission str., 104 34 Athens,
Greece

Abstract—The traditional task of managing and monitoring a network has never been a trivial one. With recent changes in computing and networking, the area of distributed systems management faces new challenges and increasing complexity. Research in the relevant field reveals that, while there are many research and commercial solutions available, some of them are based on proprietary standards. Others focus on monitoring, while lacking the ability to actively make modifications and fine-tuning. Some others have a narrow target group. This paper proposes a framework for the management of distributed applications. The managed hosts are treated as integral parts of the deployment and not as stand alone, isolated entities. The framework is based on SNMP and is not limited to monitoring. On the contrary, it is capable of carrying out SNMP-SET commands, actively modifying the run-time parameters of the managed application. Finally, it can perform the management of a variety of distributed systems, ranging from small clusters to larger scale deployments such as computational or data grids.

Index terms—Distributed applications management, SNMP

I. INTRODUCTION

The traditional task of managing and monitoring a network has never been a trivial one. Efforts have been made in that direction, a series of protocols have been designed and standards have been developed in order to facilitate the process. However, during the past decades the scenery in computing and networking has undergone revolutionary changes. From the era of single, centralised systems we have moved to an era of highly decentralised, interconnected entities. The framework is based on SNMP and is not limited to monitoring. On the contrary, it is capable of carrying out SNMP-SET commands, actively modifying the run-time parameters of the managed application. Finally, it can perform the management of a variety of distributed systems, ranging from small clusters to larger scale deployments such as computational or data grids.

In the context of this paper, the term “management” is used to refer to all 5 categories of management defined in the FCAPS model (Fault, Configuration, Accounting, Performance, Security) [1]. Furthermore, the term “distributed” is used to describe a system, application or service that is hosted on multiple nodes interconnected over a network. Therefore this includes deployments such as distributed file systems, computer clusters, peer to peer networks and computational grids. However, multiprocessor, multi-core, parallel computing and similar systems are considered out of the scope of our work, even though they are very often referred to as “distributed”.

Section II of this text briefly presents related research initiatives in the field of distributed systems management. The proposed framework’s architecture is described in section III. Section IV outlines a simple deployment and a scenario in which the framework could be used. This paper concludes with section V, discussing the framework’s advantages, compared to currently available solutions.

II. RELATED WORK

Thorough research in the field of distributed systems management reveals the fact that there are many research and commercial solutions available. It also reveals a host of new requirements that were not previously applicable in the case of traditional network management. Furthermore, contemporary applications have elevated demands in terms of availability and response time. Therefore, there is need for mechanisms to manage and monitor deployments. Fault detection,
performance tuning and auditing are some of the issues that need to be addressed.

Based on the above, we can set some criteria that are considered useful for distributed systems management. Such an application should be based on open standards. This means that it can be compatible with already existing systems and deployments and thus can achieve easier and faster penetration. It should be able to provide for active configuration and not simply perform monitoring. It should be flexible enough allowing the users to modify it in order to fit their needs. As stated above, each managed node should not be treated as an entity isolated from the rest of the deployment. Instead, what is needed is a holistic approach, treating each node as part of a larger system. Finally, a solution should target a wide variety of systems, ranging from small clusters to larger scale deployments such as computational or data grids.

To summarise, a framework for the management of distributed systems should:
1. Be based on open standards
2. Support run-time parameters’ modification, instead of simple monitoring
3. Be modular to allow customisation
4. Target a wide variety of systems
5. Have a holistic approach, treating nodes as an integral part of a larger system.

There are quite a few different approaches to the problem. Gloperf [2] was the monitoring library used in the early releases of the Globus toolkit [3]. Actually it is a library based on the Netperf measurement utility [4]. NetLogger is another well-known tool used for monitoring [5]. NASA's GMS is based on the CODE toolkit [6], [7]. It is a modular system that allows users to develop a monitoring module that suits their needs. It can be used for a variety of distributed deployments. JEWEL is an effort made in the direction of distributed system’s monitoring [8]. JAMM is an architecture based on Java and Java Remote Method Invocation (RMI) [9]. JAMM agents can be used to launch a variety of network monitoring tools and then summarise the results.

A notable standardisation effort with participants from many organisations worldwide is the Distributed Management Task Force (DMTF). Web-Based Enterprise Management (WBEM) is the product of the work of the aforementioned task force. It is a set of specifications aiming to assist the task of managing distributed computing environments. It is based on existing internet standards such as HTTP as well as other standards developed by the DMTF [10].

In the field of Grid computing, one important approach is the one proposed by the Open Grid Forum (OGF), an organisation which is the result of the merger of the Global Grid Forum (GGF) and the Enterprise Grid Alliance (EGA). OGF’s Grid Monitoring Architecture (GMA) uses an event producer – event consumer model to monitor grid resources [11]. However, as the name suggests, the GMA lacks active management and configuration capabilities.

gLite [12] is a grid computing middleware, developed as part of the Enabling Grids for E-scienceE (EGEE) project [13]. gLite implements an “Information and Monitoring Subsystem”, called R-GMA (Relational GMA), which is a modification of OGF’s GMA. Therefore it also only serves monitoring purposes.

The same is true for the MonALISA framework. MonALISA stands for “Monitoring Agents using a Large Integrated Services Architecture” and “aims to provide a distributed service architecture which is used to collect and process monitoring information” [14]. Again, the lack of capability to modify the running parameters of the managed application is notable.

Policy Based management is another recent research proposal. Policy management has been developed by the POLICY workgroup of the Internet Engineering Task Force (IETF) and a set of Requests for Comments (RFCs) have already been published. IETF recently started working together with DMTF to develop new policies. There are a few research proposals that adopt policy management for distributed systems. For example, “Ponder” defines a language to specify policies for the management of distributed systems [15]. However, those approaches have not yet achieved high market penetration.

Deeper study of the aforementioned approaches reveals that some of them are based on non-open or proprietary standards. Others focus on monitoring, while lacking the ability to actively make modifications and fine-tuning. Finally, some of them have a narrow target group, for instance focusing only on Java-based applications or only grid deployments.

This paper proposes a framework for the management of distributed applications. The framework is based on SNMP, which is an open and widely accepted standard. It can target a wide range of applications without modification. Finally, it is not limited to monitoring. On the contrary, it is capable of carrying out SNMP-SET commands, actively modifying the configuration of the distributed application. Thus, it meets the criteria set in a previous paragraph.

III. FRAMEWORK DESIGN

A distributed deployment is comprised of a number of host nodes. Those hosts are interconnected over a network and share resources to provide services to the end user. The proposed framework’s aim is to provide management facilities for those nodes. Through the management of those nodes we achieve the management of the entire deployment. Therefore, those hosts are not treated as stand alone entities, isolated from the rest of the system. On the contrary, they are treated as integral parts of a larger system.

The implementation of the framework is based on three logical entities.

- Management Node
- Service Node
- Management Representative

Fig. 1 portrays a simple distributed system as well as the three entities mentioned above.
The functionality of each of the three entities is discrete. However, this does not rule out the case of physical co-location of two types of node. For instance, a service node may also be operating as a representative node at the same time.

The architecture is compatible with the SNMP framework. As we will describe later, nodes function as SNMP entities, either as clients, servers, or both, depending on their role in the deployment. The messages exchanged between nodes are also SNMP messages.

A. Management Node

A management node is a host used in order to monitor and configure the various operational parameters of the distributed system. In this context, the management node is a simple Network Management Station (NMS) as defined in the SNMP architecture.

This is the “client” part of any SNMP deployment. In our case, any existing SNMP client software can be used without need of changes.

B. Service Node

A “Service Node” is the term used when referring to any node – member of a distributed system deployment. For instance, in a distributed file system, the term would be used to describe any device that shares storage space. Similarly, in a computational grid, each execution host would be referred to using the same term.

In our architecture, a service node executes an SNMP agent process. This handles the protocol part of the SNMP message exchange, by receiving requests, forwarding them to the appropriate sub-agent and transmitting responses.

In order for a service node to participate in our framework, it implements a set of MIB objects. Some of those are directly related to the framework itself. Those objects are defined in Node-MIB. This is accompanied by the respective SNMP sub-agent which handles the actual requests and generates responses and traps. The objects contained in this MIB control functions such as joining or leaving a domain and notifying about errors.

Fig. 2 depicts the building blocks of a service node.

Apart from the objects in Node-MIB, there is a set of objects related to the operational parameters of the managed application itself. Those objects are defined in a separate MIB, for which the term “Application MIB” is used. This is also accompanied by an SNMP sub-agent, which handles the actual SNMP message processing. It can be a standardized one or a proprietary, application specific one.

C. Management Representative

As stated previously, the objective of the framework is to manage the service nodes as nodes participating in a larger deployment and not simply as stand-alone, isolated nodes. In order to achieve that, we introduce an entity called the “Management Representative”.

This entity receives requests from the management node and forwards them to the relevant service nodes that participate in the distributed system. After a series of message exchanges, it will respond to the initial request. A management representative is more than a simple ‘proxy’ that receives and forwards requests. It performs a number of operations including the following:

- Exchanges messages with other representatives regarding the state of the system as a whole.
- Keeps a record of the service nodes that belong to deployment.
- Redirects management requests.
- Collects statistical data.

Fig. 3 below displays the simple scenario, during which a management node receives a management request. Based on the request, it then performs the required management operations on the appropriate service nodes. Those operations can be simple monitoring requests or more complex ‘set’ commands. Finally, the representative formulates a response to the original request, based on the intermediate responses from the service nodes.

In this scenario, the initial request requires the exchange of intermediate messages between the representative and three service nodes. All the message exchanges, including intermediate ones are SNMP Protocol Data Units.

In the previous example, a small scale distributed deployment is assumed. In the general case of a larger scale, geographically sparse deployment, the functionality of the representatives itself is decentralised. As stated previously, part of the framework is the communication between representatives.
Fig. 3. Simple scenario displaying the functionality of a management representative.

Fig. 4 displays the generic case of a geographically sparse system. As in the case of representative – service node communication, the message exchange between two representatives is implemented with the use of SNMP PDUs. Therefore, in SNMP terms, the representative acts as an agent as well as a client at the same time.

It is worth noting that the initial request does not state explicitly which service nodes are involved in the management task, unless the user wishes to specify them. When this is the case, the decision about the destination of the intermediate message exchange is part of the functionality implemented in the representative. Therefore, this message exchange is transparent towards the management node and thus transparent to the end user as well.

Choosing the destination of the intermediate requests as part of the functionality of the representatives has 2 advantages:

- The management node (and therefore the user) is not required to have detailed knowledge of the full system topology.
- The user has the option of issuing requests to discover part of the (or the entire) topology whenever that becomes necessary.

In order to achieve the above functionality, a representative node is further split into building blocks. As mentioned previously, a representative acts as both an SNMP agent as well as a client. The requests received by the agent are forwarded to the core module for further processing. The core performs the following functions:

- Determines whether the request can be served locally.
- If the node can not serve the request then it determines the recipients of intermediate messages that should be dispatched. Those are other peer management representatives.
- Creates the intermediate messages.
- Generates the final response to the original request based on the intermediate responses.
- Maintains information about the distributed system topology as well as the supported management objects.

The finalised intermediate messages are forwarded to the SNMP ‘client’ part of the representative, as displayed in Fig. 5.

D. Domains

In our architecture, a representative node is responsible for the management of a group of service nodes. In order to achieve that, a deployment is logically separated into domains. Each domain has at least one service node and at least one representative. The relationship between domains and representative nodes is many to many. A representative is responsible for a domain, either on its own or in cooperation with other peer representatives. This choice is based on performance and reliability requirements. Furthermore, a representative may be responsible for the management of more than one domain.

Domains themselves are organised in a hierarchical structure which resembles the Domain Name System (DNS) hierarchy. The top level of the hierarchy (root node of the tree) corresponds to the entire deployment. The exact rationale behind the domain hierarchy of each individual deployment can rely on a variety of factors. For example a system might be separated into domains based on the geographical location of nodes or based on the structure of an organisation. In any case, the hierarchy can affect the performance, ease of management and scalability of the management system.

Fig. 6 depicts the domain hierarchical structure, as well as the many-to-many relationship between domains and management nodes.

E. Management Information Base

In order to achieve the desired functionality the Management Representative implements a Management Information Base which is specific to the framework. However, this MIB does not contain any objects directly related to the managed application itself, as this would result in an application-specific system. On the contrary, the objects are generic and designed to support the representative’s task.
The objects contained in that MIB can be further divided into categories. The list presented here is only an indicative subset.

- Representatives: Those objects are related to the representatives – members of the deployment.
- Nodes: Contains objects and management information about the service nodes that join or leave the system. Information about new nodes is originally known only to the representative responsible for the node’s domain. However it is propagated to other representatives as part of intra-representative message exchange.
- Domains: The relationship between nodes and representatives is hierarchical. Objects in this category offer management capabilities for domains and the relationship between the various deployment nodes.
- Aggregates: One of the functionalities is the collection of statistical data. That data is not statically pre-determined. The objects in the MIB can be used to define new aggregate functions, depending on user requirements.

As one may notice, the overall design opens security related issues that need to be addressed. Such issues are group authentication, access control and whether or not to use encryption techniques for the message exchange. Those considerations are taken into account in the design of our framework. However a detailed description of the techniques used is out of scope of this paper.

IV. A SIMPLE SCENARIO

This section attempts to further elaborate the design of the framework with a simple scenario.

A. Setup

Assume that the service node is a typical web server. The software is an Apache Web Server. The node belongs to a cluster and some load balancing technique is in use to achieve the distributed provision of the service. This technique could be software based, like through the use of the apache mod_backhand, which is documented in [18], or through the use of a hardware-based load balancing device. Management of said node is achieved through the use of an SNMP agent, which is implemented to handle requests for the objects of the APACHE2-MIB. This MIB is application specific and corresponds to the “Application MIB” mentioned in a previous section. The same agent is executed in all hosts participating in the cluster.

With the above we can achieve management of all single nodes participating in the distributed deployment. However, each service node is treated as a stand-alone device, isolated from the other nodes that belong to the same cluster.

In order to implement our Node-MIB and the corresponding agent, one can implement a sub-agent of an SNMP AgentX. The SNMP AgentX is a plugin-enabled SNMP daemon. One could refer to it as a super-daemon. A user can write a new SNMP SubAgent and assign it to AgentX. The SubAgent does not bind a port and listen for requests. On the contrary, requests (SNMP Get or Set) are all handled by AgentX and are propagated to the appropriate SubAgent. Thus, there is no need to code a new SNMP daemon from scratch, since AgentX performs all protocol related functionality [16]. There are some implementations of AgentX readily available. Net-SNMP (formerly known as ucd-snmp) is among the most widely used [17].

Using the terms of section III, the building blocks of the service node are as follows:

- Application: The Apache web server
- Node-MIB: the new MIB and the corresponding SNMP sub-agent
- Application MIB: APACHE2-MIB
- SNMP agent: The Agent X

Similarly, the Management representative is implemented using the Net-SNMP toolkit:

- SNMP Agent: Agent X
- Core: the SNMP sub-agent and the respective MIB
- SNMP client: the command line tools, parts of the Net-SNMP toolkit.

In this sample deployment, assume 2 of those clusters, each with a number of service nodes. All nodes participate in the same load balancing scheme serving the same page. The deployment is split into two domains, Domain A and Domain B, one per cluster. There are also two management representatives, one responsible for each domain. Those are Representative A and B respectively.

B. Execution

One of the objects in the APACHE2-MIB is “busyWorkers”. The full object ID (OID) is \{enterprises.19786.1.apache2-mib.apache2MIBStatus.3\}. It is used to access the number of processes that are busy serving requests.

1. The user wishes to know the total number of busy processes over all hosts participating in Domain B. However he is not aware of the details of the responsible representative. Instead, he chooses to submit his request to Representative A.
2. Representative A has knowledge of the topology. Therefore, upon receiving the request, it forwards it to Representative B.
3. Representative B issues separate SNMP-Get requests. One to each of the service nodes in the cluster to obtain the value of the aforementioned object.
4. Each node issues a response with the current value of the object.
5. Upon reception of the responses, it calculates the total and submits a new response to Representative A.
6. Representative A sends the final response to the user.

This scenario is very simple. However, it can be expanded to more complex topologies. It can also be used to demonstrate execution of SNMP-Set commands.

V. ADVANTAGES

As stated in an earlier section, one of the important advantages of this framework is that it is based on a well known and widely used standard. There is a wide variety of SNMP software available for free, as open source projects or as the result of research initiatives. Such an example is the Multi Router Traffic Grapher (MRTG) tool [19]. Furthermore, numerous implementations of SNMP agents as well as management station software are available as commercial products. Thus, integration of the proposed framework with existing systems is relatively easy and straightforward.

In section 3, during the description of the framework, it is stated that the management station can direct the original request to any representative. Therefore the system has a “Multiple Points of Entry” feature. The request will be served regardless of what representative it is originally directed to. Provided there is no network or node failure isolating that representative from the rest of the deployment, the request will be redirected adequately. Furthermore, this aids in achieving higher availability since the system does not have any single point of failure. If a domain is served by a single representative and that node fails, the domain will be cut off. The system will behave as if all the domain’s nodes had left the group but will otherwise remain unaffected.

It has also been stated elsewhere that many currently available solutions focus on monitoring while lacking the capability to actively make modifications and fine-tuning. The proposed framework is not limited to monitoring. On the contrary, it is capable of carrying out SNMP-SET commands, actively modifying run-time parameters of the managed application. Those commands are based on SNMP message exchange. The framework can be configured to support the specific management objects used by the managed application.

Finally, the proposed framework is not application-specific in any of its parts. Therefore it can perform the management of a variety of distributed systems, ranging from small clusters to larger scale deployments such as computational or data grids. If a distributed application has SNMP management support, then this framework can be used with minor or no modifications.

REFERENCES