A System for Remote Management of Routers in Heterogeneous Data Centers

Federico Martín Casares[1] and Rodolfo Kohn[2]

[1] Instituto Universitario Aeronáutico - Córdoba, Argentina
fedecasares@gmail.com
[2] Software and Services Group, Intel Corporation - Córdoba, Argentina
rodolfo.kohn@intel.com

Abstract. Data center complexity continues to increase and has become one of the main barriers to business development and innovation. Heterogeneity is one of the main factors contributing to this complexity. To address this issue, organizations are moving towards manageability standardization by adopting technologies and protocols recommended by the Web Based Enterprise Management specification. However, in the open source arena more solutions leveraging WBEM need to be developed. This paper describes an end-to-end system implementation to manage GNU/Linux based routers using industry standards, which will contribute to the goal of reducing data center complexity. The solution comprises a number of components from a Perl-based user interface to the corresponding CIM schema and providers that instrument the target system to manage routes and interfaces. The entire system was tested in a virtual environment including different virtual networks, servers, and routers. This implementation represents a unique contribution to the open source community.

1 Introduction

Internet Data Centers are relentlessly running mega infrastructures with hundreds of thousands of servers and other special purpose devices executing the billions of lines of code that make new applications and services consumed by the Internet users. The maturity level of different technologies, encompassed by the metaphor of Cloud Computing, has been crucial to the proliferation of these applications. Indeed, the widespread adoption of virtualization, service-oriented architectures, autonomic computing and utility computing is making it possible to offer a production infrastructure that can satisfy exploding computing demand with a simple scalable model consisting of adding compute nodes to clusters and resource pools [1]. This model has also driven down capital costs by taking advantage of low-cost commodity servers in configurations of hundreds of thousands of nodes [1] [2]. However, as a result of this trend, data center complexity has dramatically increased [3] and there is a looming software complexity crisis whose only solution may be autonomic computing [4] [5], i.e. systems with self-management capabilities.

One of the main elements adding to data center complexity is heterogeneity. Data centers are composed of different devices, from several vendors and implemented with distinct and often incompatible technologies. It is possible to find tens of thousands or racks supporting hundreds of thousands of different commodity blades, NAS, SAN, load balancers, switches, routers, and different networking technologies. Additionally, on the software side, it is possible to find thousands of applications, different databases, operating systems, etc. It may be impossible for IT administrators to handle such a variety of technologies while trying to cope with the challenges the Internet poses: unpredictable workloads, intrusions, as well as unprecedented demand for availability and lower costs, among others.

It became obvious that only with the wide adoption of manageability standards can such complexity be managed [3]. Web-based Enterprise Management (WBEM) [6] standards have been embraced by the Distributed Management Task Force (DMTF) [7] as the solution for data center heterogeneity and they are also considered an appropriate solution to enable autonomic computing in heterogeneous data centers [8].

WBEM technologies are being implemented by almost all vendors. It is part of the manageability solutions implemented by all recent Windows operating systems, by VMWare in ESX servers, and it can also be found in Linux and UNIX operating systems. Most vendors are implementing WBEM-based manageability solutions for every device and environment. They can be found in operating systems, virtualization technologies and even in the pre-boot environment as in [9] where a replacement for pre-boot Execution Environment (PXE) is proposed. However, in the open source arena, even though some standards were first implemented there, such as WS-Event in Openwsman [10], and there are various projects, such as SBLIM, Open Pegasus, Openwsman, OpenWBEM, OpenDRIM, and others, there is still a need for more innovations to enable standard manageability.

This paper describes the work done in order to implement WBEM-based manageability for routers and networking devices. The project involved the implementation of the necessary instrumentation in a Linux-based router for remote management of network routes and network interfaces. The Common Information Model (CIM) and the protocol WS-Management were utilized. The solution involved the creation of the CMPI-Router provider and the implementation of the CMPI-Network provider interface offered by the open source project SBLIM. Additionally, a unique client GUI using WS-Management protocol was created and donated to the open source project Openwsman. The entire solution combined open source products like SFCB (Small Footprint CIM Broker), SFCC (Small Footprint CIM client), Openwsman, and Quagga (for routing protocols). Finally, an original set of tests was executed in a completely virtual environment comprised of several servers and routers interconnected through virtual networks. All of them were running on only one physical host.

The most important open source projects implementing WBEM components are SBLIM (IBM), OpenDRIM, OpenPegasus, OpenWBEM, and Openwsman

(Novell). Currently, none of these projects offers providers that implement CIM classes for routers. The SBLIM project offers a provider called CMPI-Network. It is a basic implementation that we extended in order to cover our needs. In addition, WBEM clients are oriented to developers rather than system administrators. Furthermore, none of these projects offer a comprehensive end-to-end solution for routers management.

As a result of this work, it is now possible to remotely manage an important set of routing functionalities in Linux-based routers using industry-accepted standards. Thus, in the future, as other routers implement WBEM manageability, system administrators will be able to operate different routers independently of the various underlying technologies and vendors.

The rest of this paper is organized as follows. Section 2 specifies the technology involved to accomplish our goal. Section 3 describes the solution architecture and provides an explanation of its main components. Section 4 describes the virtual environment used for testing and the results. Section 5 proposes possible future expansions that could enrich the actual implementation. Finally, section 6 summarizes the experience and the contributions made.

2 Technologies Involved

The system implemented involved the standard technologies and protocols proposed by the Distributed Management Task Force (DMT), the routing prototocol OSPF -Quagga open source project-, and other Linux technologies.

2.1 Common Information Model (CIM)

CIM is an information model created to provide a conceptual view of distributed systems. Its main goal is to unify management concepts and tasks independently of different technologies and vendors. This permits systems administrators to handle devices in a heterogeneous data center based on its purpose and functionalities, abstracting from specific details. [11]

In order to cover all of the aspects of a managed environment, CIM was divided into two parts, a specification and a schema.

The CIM specification defines the terms to express the model and its usage and semantics, known as CIM meta-schema. [12]

The CIM Schema provides the descriptions of the actual model. It is comprised of a Core Model and a set of Common Models that extend from the Core. The scope of the Common Models includes systems, services, networks, applications, users, databases and other managament domains. [13]

The strength of CIM includes the richness of its information models and its object-oriented representation, which allow integrators to extend from existing classes to include vendor specific content. [13]

2.2 Web-based Enterprise Management (WBEM)

WBEM is a set of management and Internet standard technologies created to unify the management of distributed computing environments. WBEM provides the ability for the industry to deliver a well-integrated set of standard-based management tools, facilitating the exchange of data across otherwise disparate technologies and platforms. WBEM includes protocols, query languages, discovery mechanisms, mappings, and anything else needed to exchange CIM information. [12]

The DMTF has developed a core set of standards to enable WBEM. Adding to CIM an encoding specification (xmlCIM) which specifies XML elements (described in a Document Type Definition, DTD) representing CIM classes and instances, and a transport mechanism (CIM Operations over HTTP) which defines how the CIM classes and instances are created, deleted, enumerated, modified and queried. Also, the specification defines a notification/alerting mechanism for CIM data. [14]

2.3 Quagga

Quagga is a software suite that provides TCP/IP routing support to computer networks through the implementation of related protocols.

It architecture brings extensibility, modularity and maintainability. It is made from a collection of several daemons (modules) that work together to build the routing table. There may be several protocol-specific routing daemons and also zebra the kernel routing manager, which is responsible for changing the kernel routing table and for the redistribution of routes between different routing protocols. We can add, remove or update a module without causing a direct effect on the other modules. Finally, if a module suddenly crashes, the whole system is not compromised. [15]

Currently, Quagga support routing such as RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, and BGP-4+. Quagga also supports special BGP Route Reflector and Route Server behavior. [15]

2.4 Linux Kernel

A kernel is a program that constitutes the central core of a computer operating system. In order to perform actions or to obtain information about managed resources, we must communicate through the kernel. The Linux kernel offers a series of interfaces, known as sockets, allowing access to advanced network functionalities. The Netlink socket is one of them. It provides a full-duplex communication link between the kernel and the user-space processes. Using this interface we can manage desired resources, like routing tables and network interfaces. Consequently, we made extensive use of Netlink sockets in the CMPI-router provider which is described in following sections.

3 Implementation

3.1 Architecture Overview

The diagram exhibited in figure 1 depicts the architecture of the WBEM/CIM system implemented. In the list that follows a brief explanation about the components is presented.

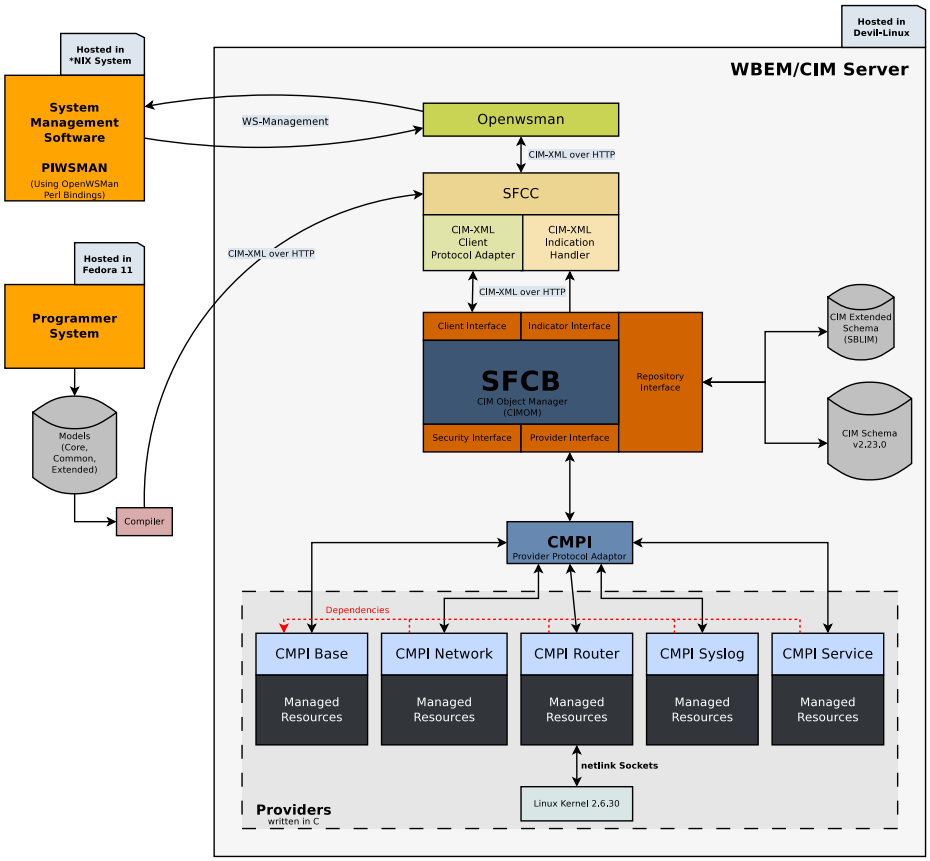

Fig. 1. System Architecture

– **Programmer System:** Based on a GNU/Linux architecture due to its advantages related to software development, this system was used to create the providers, the CIM Schema extensions and the WEB client.
– **Models and Compiler:** Both, the CIM standard (Core Model and Common Models) and its extensions, available in the programmer's computer, are processed by an MOF compiler (in this case mofc from SBLIM project).

Its purpose is to validate our extensions by generating an output intended to be our CIMOM repository. Our extended MOF were processed by this compiler, generating the outputs needed by SFCB repository.

- **SFCC (small footprint CIM client):** A C based API which makes communications between the clients and the broker (CIMOM) possible. We used this API to establish communications between SFCB and Openwsman.
- **SFCB Interfaces:** These interfaces allow communication between CIMOM and external components.
- **SFCB (Small Footprint CIM Broker):** By using information models available in the CIM repository, it manages direct communications between WBEM components, such as providers and clients. Furthermore, it validates the syntax and the semantics of the messages, providing a certain level of security to the system. When we make a request through PlWSMan client, SFCB is the responsible to contact providers and to gather the information needed from specific managed resources. SFCB is a CIMOM implementation of SBLIM project.
- **CIM schema:** The CIM Schema supplies a set of classes and associations with properties and methods that provide a well-defined conceptual framework within which it is possible to organize the available information about the managed environment.
- **CIM extended schema:** Extensions to DMTF defined standard models. In order to address our management needs, we extended the DMTF network model, to handle specific routers characteristics.
- **CMPI (Common Manageability Programming Interface):** C based programming interface, intended to provide the abstraction capability of technology and terminology used by CIMOM. Our providers were developed to be CMPI compliant.
- **CIM Providers:** Providers are special classes that communicate with managed resources, to access related information or perform actions, and to communicate it to SFCB. In order to provide access to router information and to perform actions on it, existing providers (CMPI-base, CMPI-network, CMPI-service and CMPI-syslog) were adapted and a new one was developed, CMPI-router.
- **Netlink Sockets:** Communication interface between the providers and the GNU/Linux kernel. Used when accessing managed resources.
- **GNU/Linux Kernel:** It is the bridge between the user-space and managed resources. It represents the virtual abstraction of those resources.
- **Openwsman:** It provides WS-management capabilities to promote interoperability between management applications and managed resources. We used Openwsman to allow communications between the remote clients and the WBEM system.
- **System Management Software (PlWSMan):** This user interface enables us to perform management actions over the system. Through its intuitive interface, we were able to obtain information about the resources managed, as well as to perform actions on them.

3.2 Complete path

When the operator performs an action through the PlWSMan user interface, the WBEM client sends a WS-Management request to the Openwsman daemon in the WBEM/CIM server. Subsequently, Openwsman translates this request into one or various CIM-XML commands and transmits those commands to the CIM Broker (SFCB) by using SFCC API. After the validation of messages, SFCB invokes the implemented providers to gather needed information or perform desired action on a specific managed resource. Finally, the CMPI compliant providers such as CMPI-Router and CMPI-Network execute the requested tasks through interactions with GNU/Linux kernel (i.e. Netlink sockets).

3.3 CMPI-Network

The CMPI-Network implementation had no support for the RequestStateChange method of Linux_EthernetPort CIM class (class that provides capabilities and management of an Ethernet port). This method allows us to establish the operational status of an Ethernet interface by specifying the parameter RequestedState. Given its importance to the management of network interfaces, we implemented it using BSD sockets. Even though it is only one method, several compatibility issues arose. The GNU/Linux kernel and the CIM class exhibit differences in the manner in which they treat network interface properties. For example, “Enabled”, “Disabled”, “Shut Down”, “Offline”, “Test”, “Defer”, “Quiesce”, “Reboot” and “Reset” are the Ethernet Port states specified in the CIM schema but the GNU/Linux kernel does not support all of them. Moreover, some specific information required to fill the CIM class is not available in GNU/Linux systems.

3.4 CMPI-Router

The CMPI-Router is a totally new implementation of a CIM provider, intended to offer router management capabilities for GNU/Linux systems. It was written in C following the CMPI standard and the SBLIM provider architecture. Although, its primary goal is to cover all CIM classes and associations related to router manageability, the following have been implemented:

- **Linux_LANEndpoint:** a communication endpoint which, when its associated interface device is connected to a LAN, may send and receive data frames. LANEndpoints include Ethernet, Token Ring and FDDI interfaces.
- **Linux_NextHopIPRoute:** specifies routing in an IP network.
- **Linux_CSHostedRoute:** associates the Computer System that scopes/provides context for the route and the next hop route defined on the System.
- **Linux_RouteUsesEndpoint:** associates a communication point from which data can be sent or received and one of a series of ’hops’ to reach a network destination.

In order to provide access to managed resources the provider was divided into three layers (as specified in SBLIM provider architecture):

- **CMPI dependent layer:** it implements the required CMPI APIs of the corresponding provider type. For example: "EnumInstanceNames", "EnumInstances", "GetInstance", "CreateInstance", "SetInstance", "DeleteInstance" and the methods of the specific CIM class. It is the nexus with SFCB.
- **CIM dependent layer:** it contains CMPI / CIM dependent utility functions and requires a CMPI / CIM environment. The main task is the implementation of the factory functions to create instances and object paths of the class / association. [16]
- **OS dependent layer:** it is independent of the CIM technology and abstracts the platform specific resource access. The module implements utility functions to access the resource data. It offers the provider data structures, which are similar to the provider type dependent interfaces, e.g. enumerations of all entries. [16] The hard work was done in this layer, creating a "Netlink layer" to interact with the GNU/Linux kernel and another intermediate layer intended to convert data gathered from the "Netlink layer" to CIM compliant structures.

Finally, we extended the CIM classes and associations in order to cover several characteristics relevant to GNU/Linux systems not contemplated by default CIM classes.

3.5 PlWSMan

PlWSMan is a WBEM client/listener implementation. Developed in Perl, and using the bindings provided by Openwsman, it is a web-based GUI that allows the user/administrator to monitor and execute actions on a remote computer system (remote manageability).

PlWSMan is divided into two main parts: a core and several modules. The core is composed of two libraries: plwsmanCGI.pm, responsible for the graphic interface, and plwsman.pm, designed to manage communications with Openwsman. Each module, in turn, represents an implementation for a particular CIM class to monitor and execute actions related to that class.

Available modules:

- **Linux_BaseBoard:** A computer board base representation.
- **Linux_ComputerSystem:** A Computer System representation.
- **Linux_EthernetPort:** An Ethernet Port available in a Computer System.
- **Linux_IPProtocolEndpoint:** A Protocol Endpoint with IP capabilities.
- **Linux_LANEndpoint:** A Communication Endpoint such as Ethernet, Token Ring and FDDI interfaces.
- **Linux_LocalLoopbackPort:** Representation of the Loopback virtual network interface.
- **Linux_NextHopIPRoute:** Specifies a route in an IP network.

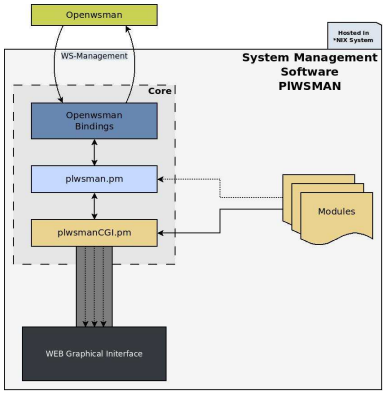

Fig. 2. PlWSMan Architecture

- **Linux_OperatingSystem:** An Operating System running in a Computer System.
- **Linux_OperatingSystemStatisticalData:** A collection of Operating System statistical information.
- **Linux_Processor:** A computer processor representation.
- **Linux_Service:** A Service running in a GNU/Linux Operating System.
- **Linux_UnixProcess:** A Unix process running in a GNU/Linux Operating System.

4 Experimental Network

In order to test the implementation and to demonstrate its benefits as well as the manageability opportunities stemmed from WBEM technologies, we created a completely virtualized environment. This environment hosted a typical network comprised of five different local area networks (LAN), three Linux based routers (Markarian, Dumbbell, Blinking) and two computer nodes (Helix and Stingray). It is worth noting that the whole environment run in only one physical machine with following characteristics:

- **Processor:** Intel(R) Core(TM) 2 CPU E6700 @ 2.66GHz (Cache size: 4096 KB) IntelVT enabled.
- **Mainboard:** ASUSTeK P5N-E SLI.
- **RAM Memory:** Corsair 4GB PC2-6400 DDR2 DIMM Dual Channel Memory Kit.
- **Hard Disk:** Hitachi HDT725025VLA380 - SATA 3.0Gb/s - 500 GB.
- **Video Card:** ATI Technologies Inc Radeon HD 5870 - PCI-E 2.0 x16 - Core clock: 850 MHz - Mem clock: 2400 MHz (4800 DDR) - Mem bw: 153.6 GB/sec - Mem type: GDDR5.
- **Operating System:** Fedora 12 x86_64.

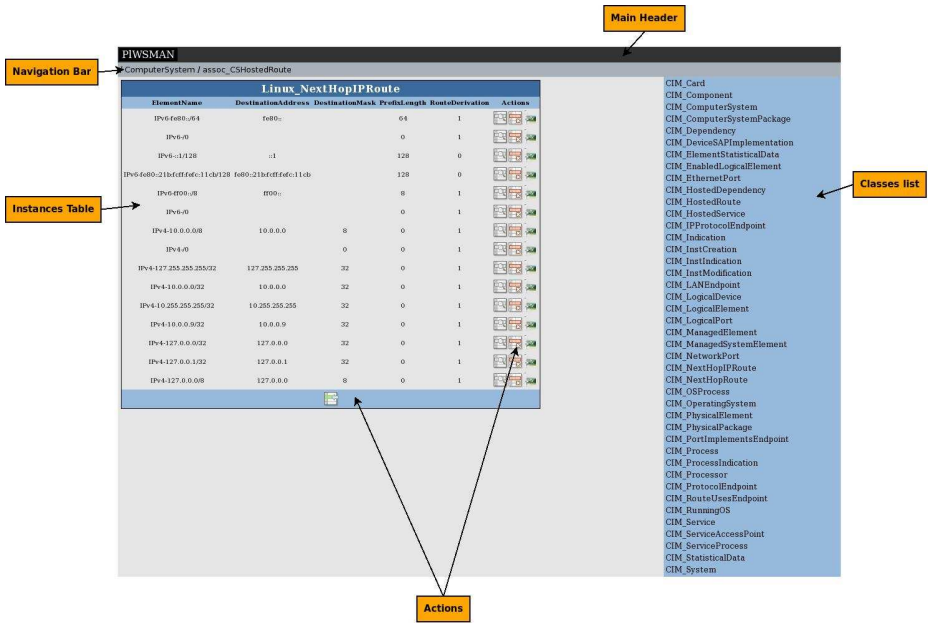

Fig. 3. PIWSMan User Interface

Figure 5 shows the deployment within our virtual environment. Below the list of networks and the respective node's interfaces used are presented:

- **192.168.1.0/24:** Dumbbell/eth1 - Blinking/eth1
- **192.168.2.0/24:** Dumbbel/eth2 - Markarian/eth1
- **192.168.3.0/24:** Markarian/eth2 - Blinking/eth2
- **192.168.4.0/24:** Blinking/eth0 - Stingray attached
- **192.168.5.0/24:** Markarian/eth0 - Helix attached

Note 1. As you can see in figure 5, we specified the KVM multicast addresses for all network interfaces. This KVM characteristic allows us to create KVM VLANs shared among QEMU virtual machines using a UDP multicast socket, effectively making a bus for every QEMU virtual machine with the same multicast address "maddr" and port.

The interface eth0 in Dumbbell allowed us to establish connections with external networks.

The virtualization infrastructure chosen was KVM (Kernel-based Virtual Machine) due to the possibility of using native virtualization, a.k.a. full virtualization hypervisor. The host (Domain 0) run Fedora 12. By using personalized Perl scripts the complete deployment was automated.

We used Devil-Linux for the routers, which is one of the most used GNU/Linux based routers. It is a simple and clean distribution with wide support for routing functionalities and does not have graphical user interface thus reducing the

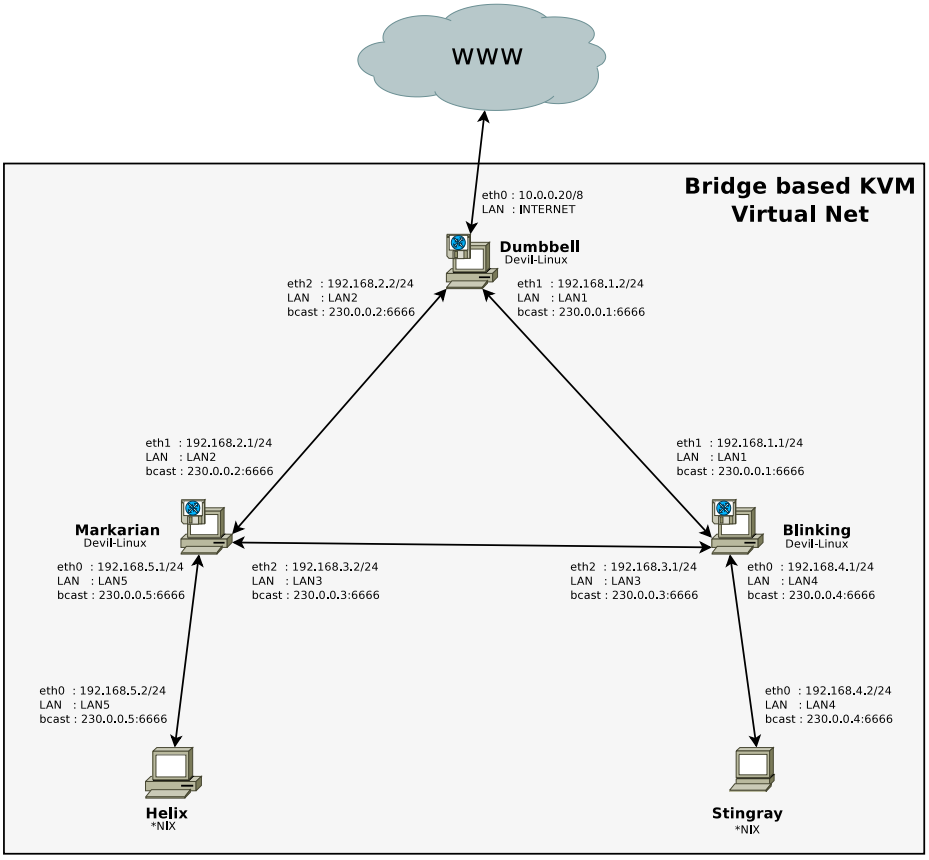

Fig. 4. Experimental Network

consumption of resources. Its customization capabilities allowed us to create a special distribution with support of WBEM technologies by writing the required scripts. Consequently, we were able to provide a complete OSPF network through Quagga.

We tested some Linux distributions such as OpenSuSe, CentOS, Gentoo and Debian for the clients, all of them used GNU/Linux Kernel 2.6 and standard tools (ip, traceroute, tcpdump, mtr, iptraf, etc.).

Several possible scenarios were established, proving the correctness of the WBEM system implemented. For each one, a series of tests were conducted such as configuration tests, functional tests, concurrency tests and performance tests.

Ones of the most representative tests to prove the implementation correctness is described in following lines:

1. After we configured the network with our specific settings, we started the Quagga daemons in the routers. Then, we used tcpdump (i.e. tcpdump -i eth1 ip[9] == 89) on them to confirm that OSPF was working properly.

2. We observed that the routers automatically set the routes based on OSPF algorithm.
3. We started a TCP echo client in the Helix computer and the corresponding TCP echo server in the Stingray computer. Then we observed the path taken by the packages using the tcpdump packet analyzer in the routers. The resulting path was: Helix > Markarian > Blinking > Stingray. On the other hand, we also confirmed this path by using traceroute command.
4. We used PlWSMan to add a new static route in Markarian router with a higher priority than the existing ones, in order to use the alternative path: Helix > Markarian > Dumbbell > Blinking > Stingray. This change in the configuration was made possible by using the remote GUI and the WBEM implementations, including the CMPI-Router provider installed in Markarian router. After the modification, we observed that the path taken by the packages had changed as expected.
5. Finally, we used PlWSMan to remotely turn off the eth2 network interface from Dumbbell router. Again, this operation was made possible by deploying the CMPI-Router provider into Dumbbell. Then, OSPF routes automatically changed after a few seconds. As a result, we observed that the system reacted as expected. The path returned to its original configuration.

Note 2. The user interface PlWSMan was tested in all the nodes and the routers deployed in the experimental network as well as in external computer nodes.

A set of tests like the one above helped us to demonstrate the correctness of our WBEM system implementation.

The results of the tests can be found as part of the source code and the documentation available in SBLIM and Openwsman projects respectively.

5 Future Work

CIM is a broad standard, whose main objective is to provide a common method to present the elements of a managed system. One of these elements is the routing protocol OSPF.

The CMPI-Router provider could be improved by adding the implementations of CIM OSPF classes not included in this work. Therefore, it would be necessary to interact with a service that provides OSPF functionalities. A good choice would be Quagga. Recently, in order to supply all the requirements of the CIM OSPF classes, a patch to Quagga was developed. This patch provides a new way to dump routing information, which we can use to fill OSPF classes. The patch was submitted to Quagga project and approved to be applied.

Another important improvement would be to support the MPLS routing protocol. However, since Quagga does not provide full MPLS support, it would not be an improvement for the near future.

Finally, since the technologies used are widely encompassing, there is a vast amount of possible implementations such as creating new PlWSMan modules, adding support for more routing protocols and developing power management providers.

6 Conclusions

Network manageability is essential for the successful operation of a data center. The infrastructures of internet data centers continue to expand, totaling hundreds of thousands of devices, applications and different technologies, which sometimes are incompatible with each other, reducing or virtually eliminating interoperability.

During this work, we created a solution providing WBEM capabilities for remotely managing GNU/Linux routers. With our implementations, a system administrator would be able to manage routes and network interfaces from a decired router. Moreover, we created an extensible solution, thereby facilitating future implementations or expansions from open source community.

Based on our experimental results, we were able to validate our implementation and verify several features provided by WBEM/CIM. We observed the great importance of: using standards to manage networks comprised of different kinds of hardware and software, respecting a common communication language, using a common way of modeling resources, and stimulating cooperation among companies.

Finally, we have donated the developed code and the written documentation to open source projects, the providers to SBLIM and the GUI to Openwsman. Thus, people around the globe will be able to integrate and use this solution improving efficiency in network management within their data centers.

References

1. Michael, M.; Moreira, J.E.; Shiloach, D.; Wisniewski, R.W.; "Scale-up x Scale-out: A Case Study using Nutch/Lucene," Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, vol., no., pp.1-8, 26-30 March 2007, doi: 10.1109/IPDPS.2007.370631
2. Isard, M. Autopilot: automatic data center management, ACM SIGOPS Operating Systems Review, Volume 41, Issue 2, April, 2007
3. A. Westerinen and W. Bumpus, "The Continuing Evolution of Distributed Systems Management," IEICI Transaction on Information and Systems, vol. E86-D, no. 11, 2003/11/01, pp. 2256-2261
4. J.O. Kephart and D. Chess, "The Vision of Autonomic Computing," Computer, vol. 36, no.1, 2003, pp. 41-50
5. Yousif M., "Autonomic Computing, Foreword," Intel Technology Journal, Special issue on Autonomic Computing, vol. 10, no. 4, 2006, Retrieved April 2011 from http://www.intel.com/technology/itj/2006/v10i4/foreword.htm/
6. DMTF. Web-Based Enterprise Management (WBEM) standards, Retrieved January 12, 2009 from http://www.dmtf.org/standards/wbem/
7. Distributed Management Task Force Web page. http://www.dmtf.org/
8. Tewary V. and Milencovic M., "Standards for Autonomic Computing," Intel Technology Journal, Special issue on Autonomic Computing, vol. 10, no. 4, 2006, DOI: 10.1535/itj.1004.03.
9. A. Kinzhalin, R. Kohn, D. Lombard, and R. Morin. 2009. Enabling the autonomic data center with a smart bare-metal server platform. In Proceedings of the 6th international conference on Autonomic computing (ICAC

'09). ACM, New York, NY, USA, 87-96. DOI=10.1145/1555228.1555257 http://doi.acm.org/10.1145/1555228.1555257/
10. A. Nashif and R. Kohn, "Eventing in WS-Management: Implementation details and a real world demo," Management Developers Conference, Web site as of April, 2011: http://www.mandevcon.com/2007/schedule.html/
11. The DMTF Technical Committee. CIM concepts. White Paper DSP0110, Distributed Management Task Force, Inc., 1001 SW 5th Avenue, #1100 Portland, OR 97204, June 2003. Version 0.9.
12. Distributed Management Task Force & WBEM Solutions. CIM Tutorial. Distributed Management Task Force & WBEM Solutions, 1001 SW 5th Avenue, #1100 Portland, OR 97204, 2003.
13. The DMTF Technical Committee. The Common Information Model. Technical Note, Distributed Management Task Force, Inc., 1001 SW 5th Avenue, #1100 Portland, OR 97204, January 2003.
14. The DMTF Technical Committee. DMTF Standards and Terminology. Technical Note, Distributed Management Task Force, Inc., 1001 SW 5th Avenue, #1100 Portland, OR 97204, June 2003.
15. Quagga. Quagga documentation. http://www.quagga.net/docs/docs-info.php/
16. Heidi Neumann – LTC Systems Management – IBM. CIM Management Instrumentation: Architecture. High Level Design Recommendation, IBM. October 2003.