Analyzing Emerging Web-Based Management Standards

Fernando Frota Redígolo
Tereza Cristina Melo de Brito Carvalho
Wilson Vicente Ruggiero
{fernando, carvalho, wilson}@larc.usp.br

LARC – Laboratório de Arquitetura e Redes de Computadores
EPUSP – Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Travessa 3 – 158, sala C1-46
Cidade Universitária – CEP: 05.508-900
São Paulo – SP - Brazil
Phone: +55-11-818-5261

Abstract *With the growth of the World Wide Web popularity, much work has been developed based on Web technologies. In particular, these technologies have been deployed and have attracted a lot of attention lately for improving existing network management tools, or even, for managing today's networks. This paper analyzes two emerging standards relying on web-based management.*

Key Words *network management, web-based management , World Wide Web, Java, JMAPI, WBEM.*

1. Introduction

Nowadays the use of computer networks is disseminated. Many people are employing them at work. As the users become more acquainted with networks, they start to increase their dependency on their services. The network then becomes a strategic asset, and downtimes and faults should be minimized. Moreover, they begin to demand for new and improved services. In order to attend these demands, many different technologies may need to coexist, such as different LAN technologies (e.g. Ethernet, Token Ring, FDDI, ATM), different network protocols (e.g. TCP/IP, IPX/SPX) and different computer platforms (e.g. Novell Netware, MS-Windows family, different flavors of Unix). As a consequence, the networks have become bigger and more complex (Figure 1).

This increasing networks'complexity has as consequence an increasing need of powerful management tools in order to fulfill users' requirements and minimize faults and downtimes. Much work has been done to improve existing tools to better aid network administrators.

With the World Wide Web popularity, many new technologies were developed for supporting new web-based applications; some of them have drawn the attention of management systems designers. The possibility of using web technologies can bring the management world some important benefits, such as:

- An inherently distributed information technology;
- A platform-independent Graphical User Interface (GUI), allowing the network to be managed from different places or even across the Internet;
- A support for security protocols and mechanisms, such as the Secure Sockets Layer (SSL);
- Database connectivity, with periodically or on-demand generated HTML reports;
- Push technology

In order to provide interoperability, standards related to web technologies in the management field have been proposed. The purpose of this work is analyzing these emerging standards as an initial step for developing further work in a web-based network management system.

1.1. Emerging Standards in Web-based Management

The use of web technologies in the management field has already gathered important supporters. In late 1996, many important computer companies joined to define standards for managing the entire network through the Web. They came up with two different standards: the Web-Based Enterprise Management (WBEM) and the Java Management Application Program Interface (JMAPI).

1.2. WBEM

Five influent computer companies (Microsoft, Compaq, Intel, BMC Software and Cisco) launched the WBEM initiative. It proposes to "consolidate and unify the data provided by existing technologies". To accomplish this target, it defines a single, object-oriented model for all

managed resources and a single protocol for accessing these resources; these elements are mapped to the information

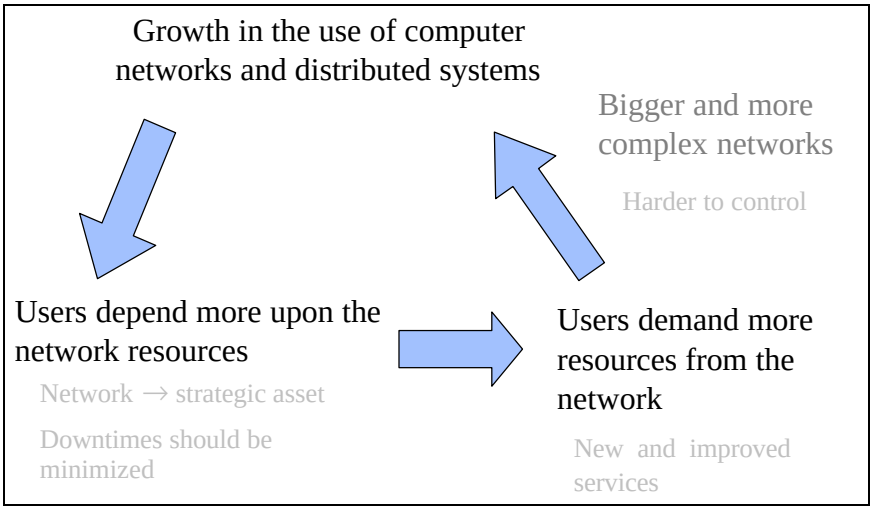

Figure 1 - Vicious circle on a network use and requirements.

model and the communication protocol existing on each managed device.

In order to get broader acceptance, the WBEM elements are being submitted to standard bodies. The communication protocol should be submitted to the IETF (Internet Engineering Task Force) in the near future, while the data model was submitted to the DMTF (Desktop Management Task Force), forming the Common Information Model (CIM)

Besides submitting WBEM to standard bodies, there are also commercial products that promise to implement it, in an attempt to make it a *de facto* standard. One of them is Microsoft's Zero Administration for Windows Initiative (ZAW, [11]), which specifies the system architecture for a fully manageable network, composed of Windows 98 and NT 5.0 machines. The other one is Intel's NetPC specification [10], which is the hardware specification of a fully manageable corporate machine.

1.3. JMAPI

The Java Management API was launched by Sun Microsystems as an extension to its Java language and also gathered important supporters. The main idea behind JMAPI is that managed resources can be modeled as Java objects. JMAPI then offers an application-level framework for building management applications, using components already developed for the Java language and integrating different management technologies under a common GUI.

It is It important to point out that these two standards are not mutually exclusive. WBEM is defined in terms of protocol

operations and a formal model for managed resources, while that JMAPI is a programming interface that leaves protocols as part of its implementation.

2. Analyzing Main Characteristics

2.1. Architecture

Both the WBEM and JMAPI are based on a proxy-architecture, where a server stands between the browser and the managed devices. In order to execute a management operation, the administrator at the browser must contact a management server that, in turn, decides how it should be done, executing any necessary conversions.

In WBEM the central component is called the CIM Object Manager or CIMOM (

, [1,2]). Besides handling requests, it has several functions, such as handling events and enforcing the security in the management system. It has a modular architecture: as it should deal with several different management systems (such as SNMP or DMI), there must be different modules to deal with each one of these systems.

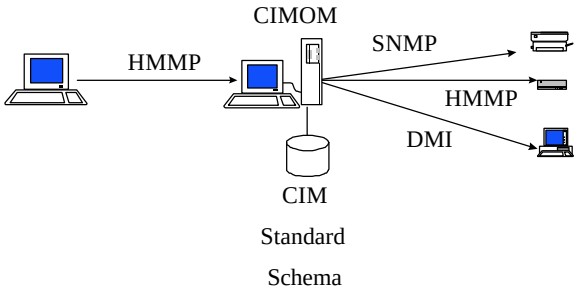

Figure 2 - WBEM architecture

These modules, which therefore act as interfaces between the abstract object defined by WBEM and the real world managed resources, are called providers.

JMAPI's proxy server is called Admin Runtime Module or ARM (Figure 3, [12]). Besides handling management requests and events, it interfaces with a separate relational database server (RDBMS), through the use of the Java Database Connectivity API (JDBC). Every successful management operation controlled by the ARM is logged on a JDBC-compliant database, and a commit/rollback mechanism is responsible for maintaining the integrity of the database. In JMAPI's framework, the RDBMS is responsible for data access security, replication and backup of management data, among other characteristics.

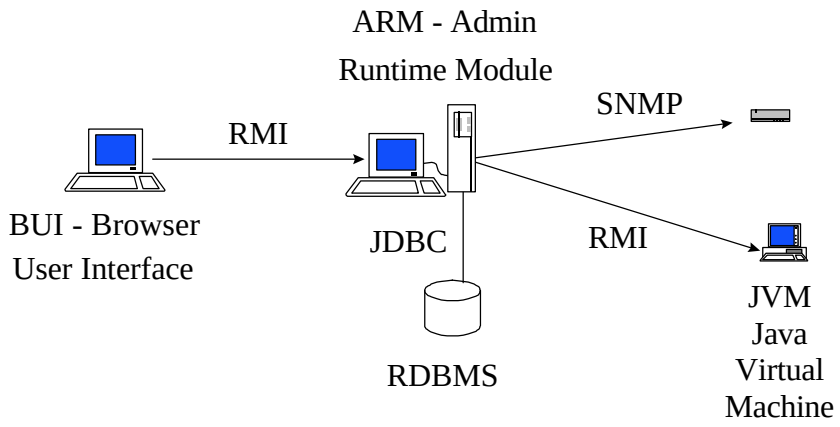

Figure 3 - JMAPI Architecture

Differently from WBEM, where it is not necessary to modify any agents, in JMAPI some features can only be used if there is a Java Virtual Machine (JVM) in the managed device and the agent is implemented in Java. Among these characteristics are extensible agents (in which new features can be downloaded on demand) and an agent versioning subsystem, (in order to update the agents distributed on the network).

2.2. Managed Resources Model

In both standards, the managed resources are modeled after an object-oriented model, with classes, objects, attributes and methods. In each standard there are several classes already organized according to a hierarchy named respectively the WBEM CIM Mandatory Schema (Figure 4, [8, 9]) and the JMAPI base classes (Figure 5, [12]). These definitions are quite similar, not only because of companies that support both standards, but also due to the fact that there are, in DMTF CIM Sub-Committee, JMAPI supporters (like Sun Microsystems, for example), which helped shape CIM model closer to JMAPI definitions. This similarity would also ease a possible convergence of the two standards, attending to the interests of companies that support both.

2.3. Communication Protocol

WBEM communication protocol is the HyperMedia Management Protocol, (HMMP). It is used for all communication between the browser and the CIMOM and between the CIMOM and HMMP-compliant devices ( ). It has several primitives, arranged in three groups [3]:

- Operations: primitives regarding classes and objects manipulations, such as the creation and removal of classes and instances, query operations and method execution;
- Indications: used to signalize the occurrence of an event;
- Security Operations: primitives used in the security subsystem.

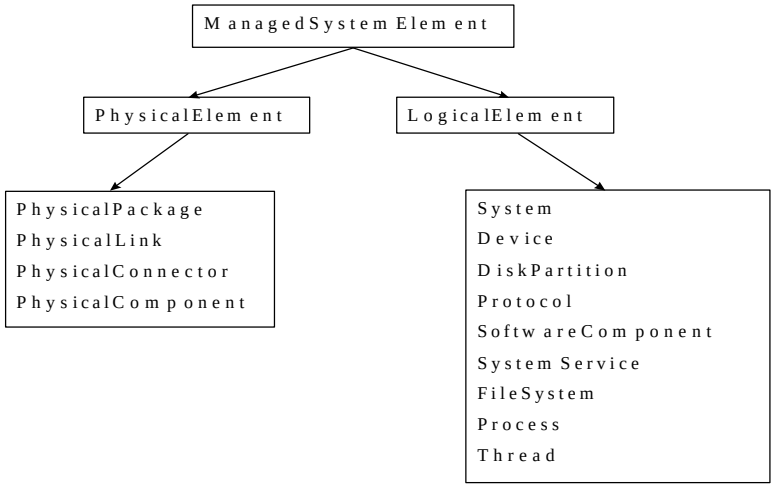

Figure 4 - Classes and subclasses in the WBEM CIM Mandatory Schema

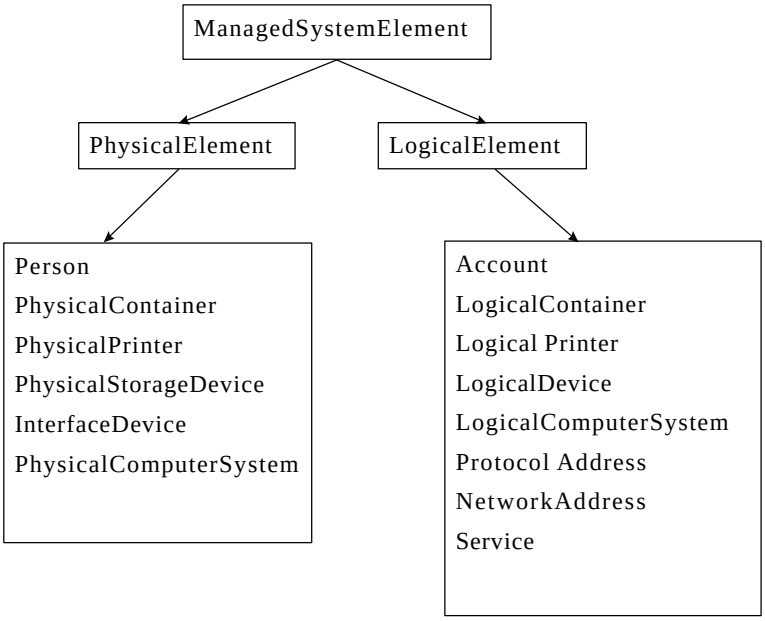

Figure 5 - JMAPI base classes and subclasses.

2.4. Communication Protocol

WBEM communication protocol is the HyperMedia Management Protocol, (HMMP). It is used for all communication between the browser and the CIMOM and between the CIMOM and HMMP-compliant devices (
). It has several primitives, arranged in three groups [3]:

- Operations: primitives regarding classes and objects manipulations, such as the creation and removal of classes and instances, query operations and method execution;
- Indications: used to signalize the occurrence of an event;
- Security Operations: primitives used in the security subsystem.

Contrary to WBEM message-passing protocol, JMAPI uses a higher level communication mechanism called Remote Method Invocation (RMI), which is Java's mechanism for distributed applications [16]. The RMI is an RPC-style mechanism for communication, where the communication details are left to the RMI compiler (it generates special code, called stubs, which will handle the communication between two RMI entities). All communication in JMAPI environment uses RMI (Figure 3).
Both HMMP and RMI depend on a separate transport protocol to work. WBEM defines the Hypermedia Transport Protocol (HMTP), a new connectionless protocol which would provide ordered delivery of data and the handling of lost or duplicated packets (however, current HMMP implementations use TCP as its transport protocol [7]). On the other hand, RMI uses two different transport protocols: besides TCP (the preferred one), it offers an alternative method, which encapsulates an RMI request in an HTTP POST request to the ARM. This feature is used to pass RMI calls through a firewall's trusted HTTP port and is automatically done when the RMI client cannot open a direct TCP connection to the server [16].

2.5. Administrator GUI

Both standards use a browser as the interface between the network manager and the management system. As WBEM uses HMMP for all communication involving the browser, it must be an HMMP-compliant browser (which could recognize *hmmp://* URLs) or instead an Active-X control that speaks HMMP. The browser is not necessary in WBEM framework; in fact, in Microsoft ZAW initiative the browser is an option, as it is replaced by a program called Microsoft Management Console - MMC [11].
JMAPI framework uses a Java-enabled browser as a container for Java applets: after downloading JMAPI initial applet (through HTTP or HTTPS), it will handle all RMI communication with JMAPI server. All applet loading is done through HTTP and HTML pages. Again, an applet viewer can be used instead of a browser.
In JMAPI architecture the GUI has great importance: instead of integrating underlying management technologies in managed devices through a new information model and protocol, it integrates them through the GUI. Special graphical elements, more suitable to handling management data, extend Java's windowing classes and are defined in [13, 14].

2.6. Database Connectivity

In WBEM the database is integrated with the CIMOM and all data access must be done using HMMP query operations (which uses a subset of the SQL language). Contrasting with this approach, JMAPI uses a separate commercial RDBMS, which must be compatible with JDBC. This approach allows access to data not only through JMAPI elements, but also through other standards, such as SQL and ODBC. It also allows that reports in HTML are generated on a periodic

basis, freeing the administrator of consulting periodically the management subsystem.
JMAPI's approach of separating the database server from the management server eases the construction of management applications, freeing the programmer from database details. Another important point is that while on WBEM the CIMOM is a management server plus a DBMS which must reside on the same machine, on JMAPI it is possible to put the RDBMS and the ARM on separated machines. This characteristic also gives the system higher availability: as the database can be replicated, a fault in a RDBMS would not significantly impact the operation of the system.

2.7. Distributed x Centralized Management

Both standards define a centralized architecture, where different administrators manage the network through the same server. It is also possible to have some degree of distributed management through different servers managing different resources (this would enable, for example, two subnetworks separated through a WAN link to be managed by different management servers). However, none provides facilities for manager-to-manager communication (i.e. communication between proxy servers), which would enable hierarchical management and data correlation between two entities.

2.8. Security Subsystem

A web-based management system must have a carefully designed security subsystem: the flexibility of managing an entire network with a browser, outside firewalls, opens a new door for outside intruders. WBEM and JMAPI take rather different approaches to security.
JMAPI does not define a formal security subsystem of its own; rather, it relies on Java and RMI security, besides on efforts being made to add security facilities to the Java language. Three key elements compose this strategy: the Java security model [17], the Java Security API [18] and extensions to RMI [16]. The former defines rules for applets, class loading, among others. The Java security API defines facilities to deal with keys, certificates, access control lists (ACLs), message hashing and digital signatures. Finally, the latest RMI specification enables the use of encrypted sockets, based on SSL.
The approach made by WBEM is the definition of a whole new security subsystem [6]. It is divided in two subsystems: network security and schema security. The former deals with securing HMMP traffic, and defines three levels: authentication only (Level 1), authentication plus message encryption (Level 2) and Level 2 plus external security mechanisms (Level 3). Schema security deals with access control and also defines three independent levels, ranging from full access for authenticated users (Level 1) until Access Control Lists (ACLs) for each class or object (Level 3).
It is especially important to examine how these security elements can provide key security services, like authentication, non-repudiation, confidentiality and discretionary access control.

2.8.1. Authentication

JMAPI, or rather, Java, offers only basic blocks for programming an authentication mechanism: encryption, certificates, keys, hashing. An authentication mechanism must be devised in order to guarantee JMAPI security.
On the other hand, WBEM network security defines a simple authentication mechanism, based on a challenge/response algorithm, for its level-1 network security protocol:

- A client requesting a logon sends the server the username requesting the logon;
- The server, in order to authenticate a client, sends back a newly generated random 128-bits value, the "nonce";
- Both sides compute an Access Token using the MD5 message digest algorithm:

Access Token = MD5(MD5(password) XOR nonce)

- This Access Token is then included in every HMMP operation, in order to authenticate the requests (it must be equal to the one calculated by the server).

Although the username's password does not need to be recorded on the server (only its digest) and the password is never transmitted, it is vulnerable to playback attacks. If an intruder can eavesdrop the network and intercept any HMMP operation, then it is possible for him/her to obtain the Access Token and put it in his own HMMP operations. To minimize this drawback, "nonces" are never reused and the Access Token has limited lifetime (the server can invalidate it at any time).

2.8.2. Confidentiality

Both standards have not defined precisely how confidentiality will be guaranteed. While that on JMAPI confidentiality will be enforced through the use of SSL together with RMI (which is not possible at present), WBEM defines a draft protocol for network security level 2 which uses RSA public key cryptography during the authentication phase and RC4 symmetric cryptography in subsequent messages. These keys are exchanged using the following protocol:

- A client requesting a logon generates a public/private key pair (with 40-bits key strength) and sends the server the username and its newly calculated public key;
- The server then also generates a 128-bits "nonce" and a public/private key pair, and sends back the "nonce" and its generated public key;
- Both sides generate the Access Token, as in level 1; the client then requests to the server a session key, including the Access Token encrypted with the server's public key;
- The server decrypts the Access Token to validate the client (as in level 1), generates a 40-bits session key and sends it back, encrypted with the client's public key;
- The client then decrypts the session key, which should now be used for subsequent requests. The Access Token is still used, in unencrypted form, to identify the client requesting an operation.

As in level 1, the Access Token is still used unencrypted to authenticate the requests, which is still dangerous, as it can be still extracted by eavesdropping and used in impersonation. Although an intruder cannot submit the server a custom HMMP operation (as the operation itself is encrypted), it can force the server to process garbage, in a sort of denial-of-service attack: the server sees the valid Access Token, decrypts the operation and tries to process it, with unknown consequences.
There are a few problems with the use of keys in WBEM. First, it uses only 40-bits keys, which is not considered as a strong encryption (especially in the public/private key pair), and it may not resist to real-time cryptanalysis. Besides, the use of random public/private keys in the server instead of certificates can lead to impersonation of the server: there is no guarantee, as there is with a certificate, that the public key sent is from the server desired. An intruder may spoof an address for the server, establish a secure channel with the client and receive confidential information, such as passwords, from the trusting client.
The use of stronger encryption algorithms and certificates is left to level 3 network security, which is open for implementation-specific extensions to the security model.

2.8.3. Discretionary Access Control

Like authentication mechanisms, JMAPI offers facilities to deal with ACLs, enabling the programming of access control in JMAPI objects. Combined with a custom security manager, it may be possible to use it to control not only access to objects' attributes, but also to its methods. However, it must be hard-coded, and changes to ACLs demand recoding the ARM.
Instead of using Java ACL mechanisms, it is also possible to use access control mechanisms present in the RDBMS. It depends, though, on a JMAPI authentication subsystem, and is limited only to access to data already gathered, not to management methods.
In WBEM access control is left to the schema security subsystem, especially level 2 and 3 (in level 2, we can bind ACLs to groups of objects, while that in level 3 we can specify ACLs for each object)

2.8.4. Non-Repudiation

There is no mechanism to deal with non-repudiation in both standards. While that in JMAPI it must also be built, in WBEM the possibility of impersonation makes non-repudiation not feasible.

JMAPI approach of leaving the whole security subsystem open is rather dangerous, despite its flexibility. As the authentication protocol must be coded almost from scratch, an implementation error could have some serious consequences, giving an intruder access to the entire management system. It is not very practical also: a new security policy means recoding JMAPI elements. The Java design team is looking into these drawbacks, and an authentication mechanism and dynamic security policy are on their to-do's list.
WBEM approach is not without faults either. Level 1 network security is subject to eavesdropping of management information and impersonation. Level 2 is more secure, but the small size and randomness of public/private keys and the transmission of the Access Token unencrypted can compromise security in face of eavesdropping, real-time cryptanalysis, and impersonation.

2.9. Use of Web-related technologies

One of the main forces behind web-based management is the possibility of using web protocols and technologies towards the management area. Not only it is easier and faster to implement applications based on existing technologies but it also needs less testing (the technology itself does not need to be tested, only conformance tests need to be applied).
WBEM makes little use of web technologies, justifying the "web-based" title by using Java applets and/or Active-X controls for its user interface. Besides using Java applets, JMAPI uses other web technologies:

- Context-sensitive help pages for the management system (eventually for documenting the network) are composed of HTML pages. These pages must be built using some JMAPI-specific tags, which will enable a precompiler to create an integrated table of contents, cross-reference index, glossary and indexed keyword table (for its help subsystem's search mechanism).
- RMI can use HTTP as its transport protocol: if a direct TCP connection to the server port is not possible (for example, if a firewall stands between them), then the applet encapsulates RMI messages in HTTP POST messages.
- RMI will be able to use SSL as its transport protocol.

There are two important technologies that should also be considered when it comes to web-based management: Secure Sockets Layer (SSL) and push technologies.

2.9.1. SSL

In order to guarantee confidentiality in the communication between the browser and the server, these standards could use the SSL protocol, which is already implemented in most browsers and has been extensively used and tested.
It is possible to use SSL with WBEM: instead of mapping HMMP over a standard TCP connection, it could be mapped over an SSL-channel. However, it may be unnecessary as WBEM designers decided to implement encryption in HMMP messages with level 2 network security. As level 2 is not formally defined, a more secure choice may be using SSL with level 1 to avoid impersonations and disclosure of confidential information.
At present it is not possible to use JMAPI with SSL. Although there are SSL Java implementations available and RMI uses sockets or HTTP as its transport protocol, it is not possible to modify the current RMI implementation to support SSL. It is promised that the next major version of RMI (present in its JDK 1.2) will support the use of "custom sockets", which can be modified by the programmer (among the "custom sockets" already supported there will be an "encrypted socket" type that uses SSL).

2.9.2. Push Technology

The use of push technologies can be quite useful in a network management: instead of requesting information periodically, an entity could register with the information provider, which would then send the information periodically or whenever it changed. Some scenarios that could be imagined through the use of push technologies:

1. System software on managed devices could be updated dynamically. Instead of upgrading the software on every machine, a repository server could send the managed device the new version to be installed;
2. The previous scenario could be extended further: through the Internet the repository server could be automatically fed with newer versions of drivers and applications directly from their own manufacturers;

Similar to the first scenario, JMAPI offers an agent versioning subsystem: a JMAPI-compliant device can download new versions of its agent libraries whenever they are available on the ARM. However, instead of using push mechanisms, it is based on polling, where the agent verifies with the server the latest version available. It could, though, be implemented inside JMAPI, by reprogramming the agents.
Implementing a push mechanism with WBEM is also possible, however, as it is an application-level mechanism, it would be implemented on top of WBEM, not within the architecture.

2.9.3. Java Applets and Active-X controls

Both protocols rely heavily on these technologies at the browser side. While JMAPI uses only Java applets, WBEM uses either applets or Active-X controls, encouraging the use of the latter. JMAPI also uses Java on the server side, differently from WBEM (its SDK provides C++ libraries to support the implementation of the CIMOM and its providers).

2.10. Portability

Despite using a browser, which should enable the system to be managed from any place on the Internet, WBEM is very Windows-centric. It must use an Active-X enabled browser, which nowadays means only Microsoft Internet Explorer (or Netscape Navigator for Windows with special plug-ins), and in WBEM SDK the providers communicate with CIMOM using Microsoft's Common Object Model (COM), leaving it tied up to Windows platforms. There is already some work being done to ease this lack of portability. There is a Java API which provides applets a HMMP implementation and functions to handle CIMOM information. The providers and the CIMOM can also communicate through HMMP and there is already a WBEM SDK version for HP-UX systems.
As it depends only on a Java Virtual Machine implementation, JMAPI can be used on any system on which there is such an implementation. Despite being relatively easy to find a JVM in browsers (for the BUI) and in operating systems (for both the ARM and for managed devices), it limits its use on networking devices, such as switches and routers. For these devices the ARM must be modified to convert JMAPI requests to theirs management system, in a similar fashion as WBEM providers do.

2.11. Compatibility with other Management Protocols

In WBEM there must be a different provider for each management system. These providers are registered with the CIMOM so that, when a request arrives, it knows which provider is capable of handling it. In latest WBEM Beta SDK, there are several providers, which deal with SNMP-devices and information specific to Windows 32-bits systems. This SNMP provider deals only with SNMPv1 protocol, and it does not support SNMP traps. As an SNMP device is viewed as a collection of CIM schema objects, the SDK offers a MIB to CIM compiler, easing the burden of converting between the two different information models.
SNMP support in JMAPI is implemented as SNMP classes. These classes support SNMPv1 protocol, loading of MIB definitions and offer a way to convert SNMP traps to JMAPI events. To support SNMP devices, the ARM must be built using these classes.
It is important to point out that making WBEM interoperate with a new management system can be less painful than on JMAPI. In order to do so, a provider must only register with the CIMOM (through the use of an appropriate API), which requires no modification to CIMOM's code and could be done without shutting down the system. Supporting a new protocol on the ARM requires its modification and, depending on how it was done, could require a new compilation of its code. Moreover, many JMAPI characteristics can only be used if we upgrade the managed device with a JVM, like the agent versioning subsystem and extensible agents.

3. Conclusions

JMAPI	WBEM
Better use of web technologies	Web technologies used only on GUI
RPC-style communication protocol	Message-level communication protocol
Security subsystem not defined in the standard, depending on Java security extensions	Defined security subsystems can be subject of impersonation, cryptanalysis and eavesdropping
Greater portability achieved through the use of Java language	Portability depends on porting WBEM libraries to new platforms
Supports SNMPv1 operations (including traps)	Supports SNMPv1 operations, except traps
Building management applications demands knowledge of Java and the management protocols that will be used (such as SNMP)	Demands knowledge of Java/Active-X, HMMP and may demand knowledge of the management protocols that will be used (if there is not an specific management provider)
Advanced features, such as agent versioning and help subsystem	Applications must be built on the top of the architecture
Accessible through any Java-enabled browser	Accessible through any Active-X or Java-enabled browser
Integration with commercial RDBMS	DBMS integrated within management server
Queries made through SQL and JDBC	Queries made through subset of SQL language
Needs JVM on managed devices to use some features	Does not need to change any managed device

Table 1 - Summary of JMAPI and WBEM characteristics.

The use of web technologies can bring benefits to the management area, drawing the attention of companies and researchers alike. Two standards are being developed for web-based management: the Web-Based Enterprise Management (WBEM) and the Java Management API (JMAPI). Table 1 summarizes their main characteristics.

The main drawbacks in the JMAPI standard are the lack of an integrated security subsystem and the need to upgrade managed devices with a JVM to use some of its features (although some companies are seriously considering using Java in network devices, which may give JMAPI broader applicability in a near future). It presents, however, better use of web technologies and advanced features for building web-based management systems.

WBEM, on the other hand, makes little use of web technologies and defines new elements, which will need to go through testing yet. It demands knowledge of an extra management protocol and information model, and its security subsystem also presents some serious flaws.

Choosing one of these standards is not an easy task. There are important supporters on both sides, and important companies supporting both sides, which can lead to an uncertainty about which one, if any, should prevail in the future. Both standards are still immature and present drawbacks that must be looked into so that they can be used on enterprise networks. Technically speaking however, JMAPI may present more facilities for experimenting with linking web technologies inside the management field, especially for people who already know management applications.

4. References

[1] Introduction to HMMP, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-intro-03.txt

[2] HyperMedia Management Protocol Overview, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-overview-03.txt

[3] HyperMedia Management Protocol, Protocol Operations, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-opns-05.txt

[4] HyperMedia Management Protocol, HMMP Events, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-events-02.txt

[5] HyperMedia Management Protocol, Query Definitions, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-query-03.txt

[6] HyperMedia Management Protocol, Security and Administration, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-security-03.txt

[7] HyperMedia Management Protocol, TCP Transport Mapping, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-tcp-transport-02.txt

[8] HyperMedia Management Protocol, Mandatory Schema, May 1997 http://wbem.freerange.com/wbem/draft-hmmp-mandatory-schema-03.txt

[9] Common Information Model, Core and Common Schema, Desktop Management Task Force, Version 1, April 1997 ftp://ftp.dmtf.org./cim/cimdoc20e.PDF

[10] NetPC (Network P System Design Guidelines , Version 1.0b) ftp://download.intel.com/ial/wfm/netpc.pdf

[11] Microsoft Zero Administration for Windows Initiative http://www.microsoft.com/management

[12] Java Management API Programmer's Guide http://java.sun.com/products/JavaManagement/documents/prog_guide/prog_guide.pdf

[13] Java Management API User Interface Style Guide http://java.sun.com/products/JavaManagement/documents/style_guide/style_guide.pdf

[14] Java Management API User Interface Visual Design Style Guide

http://java.sun.com/products/JavaManagement/documents/visual_guide/vis_guide.pdf

[15] Java Management API Help Developer's Guide http://java.sun.com/products/JavaManagement/documents/help_guide/help_guide.pdf

[16] RMI Architecture http://java.sun.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html

[17] White Paper: Secure Computing with Java: Now and the Future http://java.sun.com/marketing/collateral/security.html

[18] Java Security API http://java.sun.com/products/jdk/1.1/docs/guide/security/JavaSecurityOverview.html