Network Management using Internet Technologies

Franck Barillaud
IBM CER
06610 La Gaude, France
email: fbarillaud@vnet.ibm.com

Luca Deri and Metin Feridun
IBM Zurich Research Laboratory
8803 Rüschlikon, Switzerland
email: lde@zurich.ibm.com, fer@zurich.ibm.com

Abstract

As networks grow in size, speed and flexibility, the role of network management becomes increasingly important. Recent developments in Internet technologies might provide the capabilities that are well suited to the solutions of some very challenging, outstanding problems in network management. The increasing popularity of the World Wide Web, with its established user interface and the ability to run on almost any platform, offers a new way to provide wide access to complex software applications. The goal of this paper is to describe some key capabilities of Internet technologies and critically assess whether there is a match between these technologies and the needs in managing networks.

Keywords

Network Management, World Wide Web, Java.

1 INTRODUCTION

Networks today are managed typically through the use of powerful and general purpose monolithic management platforms. These to some degree of success provide integration of management tools, but pose a number of disadvantages:

1. Management platforms are expensive, both in terms of software as well as the cost of the hardware required.
2. They are typically complex to install, run and maintain.
3. Management platforms are based on the centralized paradigm of management: a small number of sites (typically a single one) collect data from the network and analyze them. This can create bottlenecks and thus delays in reacting to network problems; in addition, ability to scale becomes an issue.
4. In general, it is difficult to *remotely* access data and tools on the management platforms. The means available are typically primitive such as telnet, and do not match the capabilities offered on the consoles.

Recent emergence of Internet technologies such as the World Wide Web (Berners-Lee, 1992) and the Java language (Arnold, 1996) offers new means to overcome some if not all disadvantages of today’s network management platforms.

The web browser as an end-user interface is available on almost every computer platform and is enjoying widespread use in low-cost access and integration of a broad range of services. The Java language on the other hand provides the means to create software applications that are portable across platforms and can be distributed, accessible through web browsers.

In this paper, we show two ways the web and Java technologies can be applied to the complex world of network management. We outline the effort required to integrate powerful tools into the Web, providing the ability to manage network resources efficiently using HTTP and Java. We also show how the integration of Java with mobile agent technology allow us to create a new generation of roaming applications to better solve problems that affect networks every day.

2 INTEGRATING THE NETWORK MANAGEMENT WORLD INTO THE WEB

The key driver for integration of web technologies into network management is the desire to have simple but powerful tools accessible from every platform. The web technologies (web browser, HTTP, HTML and web servers) provide in addition the following benefits:

1. Web servers act as central repositories, reducing maintenance costs. For example, a management application can be changed at a server, and these changes propagate automatically through web browsers; there is no need to update the browsers.
2. Facilities such as documentation and on-line help can be consolidated at servers.
3. Web "pages" provide a natural environment to integrate multiple services.

A survey of the current trends in the use of the web technologies for network management can be found in (Jander, 1996).

Webbin': HTTP-based Network Management

Webbin' is a research project which aims to simplify the way network management is performed. Webbin is based on the idea that the complexity of protocols such as CMIP or SNMP has to be hidden by the system and that the users have to rely on the services provided by the system and to reuse them every time a new application has to be developed instead of replicating them (craftsman paradigm, i.e. everything has to be custom built for a certain task). The core element of Webbin is a software application called *Liaison* (Deri, 1996a) which allows CMIP/SNMP resources to be managed through HTTP (Figure 1). Liaison is based on a special type of software components called *droplets* that have the ability to be replaced and added at runtime allowing the dynamic modification and extension of the behavior of the application that contains them. Liaison comes with droplets that implement all CMIP and SNMP operations, a basic directory service and a metadata repository for SNMP. Additionally there are a couple of droplets that have the ability to query the metadata information contained inside the OSI stack. The idea is to implement a droplet for each management CMIP/SNMP operation and then cooperate with the existing droplets in order to reuse the services they provide especially with respect to the metadata access. This demonstrates how powerful software components are and how they allow the reuse of existing services and then the incremental building of applications instead of starting from scratch every time.

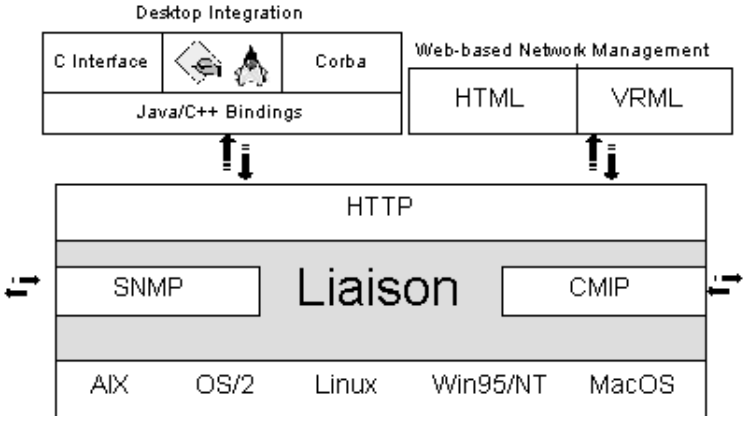

Figure 1 Liaison Architecture

The basic Liaison configuration contains droplets for:

1. browsing CMIP and SNMP resources using a Web-browser;
2. displaying network topology in 3D using VRML (Virtual Reality Markup Language) (Deri, 1997b);
3. performing batch operations which are used by *external bindings*, available in C, C++ and Java, which enables the creation of simple management applications by exploiting Liaison's services (Deri, 1996b);
4. managing CMIP and SNMP instances from Corba (OMG, 1995) through *Corba Bindings* (Deri, 1997a), exploiting the external bindings.

The droplet paradigm allows one to combine services easily. For instance Liaison comes with a droplet that implements a directory service. This service is used by the discovery droplet that is responsible for locating the resources available on the network. Composition of services has several advantages. It keeps the application complexity low and allows service implementations to be replaced with new and more efficient ones without having to affect the users of those services. Liaison has the ability to transparently locate and exchange information with other Liaison's running on different hosts. This enables one to use the Liaison that is closest to the managed resource. In fact, thanks to the TCP/IP-based Liaison-to-Liaison communication services, a Liaison that has to deal with management resources running on a remote host, when possible delegates the request to another Liaison that is running locally with respect to the managed resources. This solution allows bandwidth to be saved because:

- local computation: all communications Liaison-managed resource are local,
- Liaison-to-Liaison communication uses a simple protocol that moves less data than an equivalent CMIP/SNMP request/response,
- Liaison-to-Liaison communication is always 1:1 (1 request/1 response), whereas CMIP is 1:n in the case of scoped operations.

CMIP/SNMP protocols have been mapped to URLs (Deri, 1996c). This mapping is very important because the entire architecture relies on it. It is based on strings, i.e. every value is mapped to a string, and it allows a URL to be mapped uniquely to a management operation and vice-versa. The URL is composed of 5 elements, http://<host>/<protocol>/<operation>/<context>?<parameters>, where: a) <host> identifies the host where the HTTP server runs, b) <protocol> specifies the protocol used c) <operation> specifies the protocol operation d) <context> specifies the context to use, if any e) <parameters> contain the operation parameters, if any. At the moment we are using HTML for basic network management. This is very useful for instance in situations where Liaison

is installed on a managed device and users connect to the box via SLIP or PPP protocols and manage it using a conventional Web-browser. Whenever it is necessary to provide a more sophisticated application or when multiple operations have to be performed in batch mode, then users can write their own applets or applications using the external bindings. In case an interaction with Corba is needed, then the Corba bindings offer a way to manipulate CMIP/SNMP resources from a pure Corba world without having to deal with large amount of generated code.

3 NETWORK MANAGEMENT AGENTS

The management of high speed networks often require massive transfers of data. Such transfers can be inhibited by bandwidth constraints, storage limitations and data ownership considerations. Today's management systems are based on a platform-centered paradigm that separates applications from the required data and services. This centralized paradigm has been stretched to its limits by the emerging large scale, complex multi-domain networks. There is a growing need for new technologies to automate and distribute management functions to enable scaleable and robust operation.

Mobile agents are programs that can be dispatched and executed remotely under local or remote control. They can be used to move management functions to the data rather than move the data to these functions, thereby facilitating network management architectures for distributed automated management of networks of arbitrary scale and complexity.

In the traditional management environment, a program that monitors virtual circuits at an ATM switch must poll MIB tables and analyze them at the network management station. Both the data provided by the MIB as well as the control functions that it supports are rigidly built-in at MIB design time. The network management platform must poll and process large amount of MIB data (the virtual circuit MIB table can include thousands of entries) at a fast pace. On the other hand, a mobile agent can be delegated to the ATM switch to monitor and analyze the virtual circuit table . It can access the MIB and other relevant data directly by the instrumentation located in the hardware, more efficiently and reliably than a remote network management platform.

Health functions may also be used as examples to illustrate the need for mobile agents in the management of high speed networks. Health functions cannot be included as part of a static MIB design as they may vary from site to site and over time. Nor can they be usefully computed at centralized management platforms, as this can result in excessive polling rates and lead to errors due to perturbations introduced by polling. Mobile agents can support flexible and effective evaluation of health functions and threshold decisions at devices. For example, an agent in a LAN hub can check the status of a port and disconnect it when its load exceeds a threshold.

Use of mobile, intelligent agents can also enable managed devices to acquire autonomous management capabilities. For example, when communication with management entities is lost, a device may activate management programs, i.e. agents, that provide it with sufficient pre-defined management functions to allow autonomous operation of the unit.

SMAP: Smart Management Applications

The goal of the SMAP research effort is the development of a distributed management paradigm to overcome the disadvantages of centralized network management. Smart Management Applications (SMAPs) are small, mobile network management tasks that can be executed at one or more locations in the network with the following key benefits:

1. small footprint, as SMAPs are specialized to handle a small set of network management tasks and are activated only when necessary;
2. localized problem resolution, as SMAPs can execute close to the managed equipment. thus providing a closer, detailed view of the network state as well as reducing network management traffic by consolidating collected data locally.

In the SMAP approach, the management applications (or at least parts of them) move closer to the probes (agents) to solve management tasks. The centralized manager or console is reduced to the role of presenting information and invoking management tasks as necessary.

SMAP Architecture

The architecture of the SMAP platform is shown in Figure 2. The platform is based on the principle that management applications should only impose a minimal overhead on the managed system.

SMAPs are code objects which can be easily down loaded on demand to carry out management tasks. There are two categories of SMAPs:

1. SMAPs which drive management tasks, for example, a management console application;
2. SMAPs which when activated, carry out network management tasks such as monitoring packet delays at a network node.

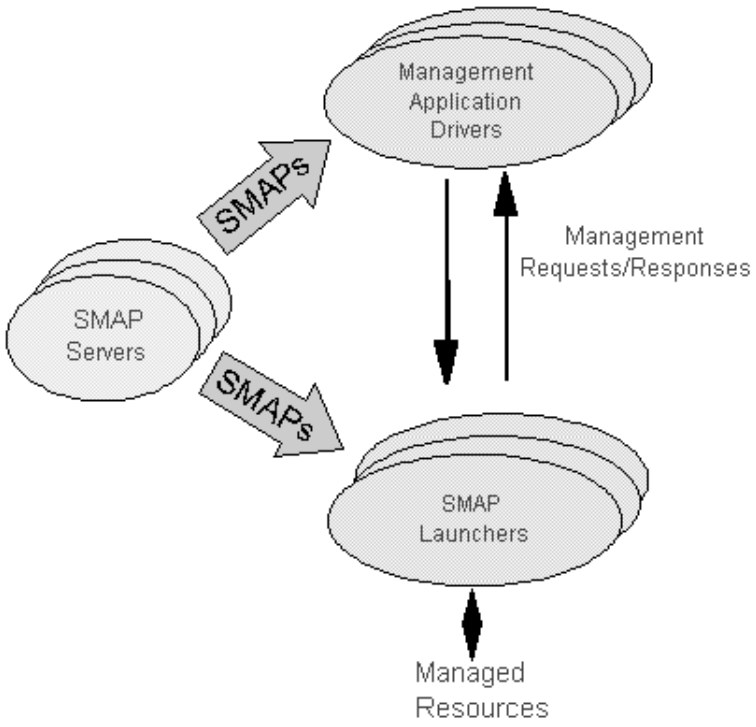

Figure 2 SMAP Architecture

The SMAP platform consists of three components, distributed across the network:

1. Management Application Drivers activate management tasks in the network and also provide, if necessary, the network operator interface.
2. SMAP Launchers receive commands from management application drivers to carry out a management task, download appropriate SMAPs for the task, and start them. Activated

SMAPs can communicate with other SMAPs or with the managed resources.
3. SMAP Servers are SMAP repositories, used by both management application drivers and SMAP launchers.

The Java language has been chosen for the implementation of the SMAP platform. Java provides convenient facilities for the movement of code, a required feature of the SMAP architecture and is portable across many platforms.

SMAPs for driving and carrying out management tasks are coded as Java classes (as applications or applets) and are retrievable through HTTP servers across the network (Figure 3). The management application drivers consists of an appletViewer or a Java-enabled browser which loads the appropriate driver SMAPs as required. Once activated, these SMAPs connect to selected SMAP launchers.

In a launcher, the SMAP Launcher Java class instantiates an instance of the ApplicationHandler class, which in turn downloads appropriate SMAPs from an HTTP server and activates the management task. Loaded classes are removed when no longer needed.

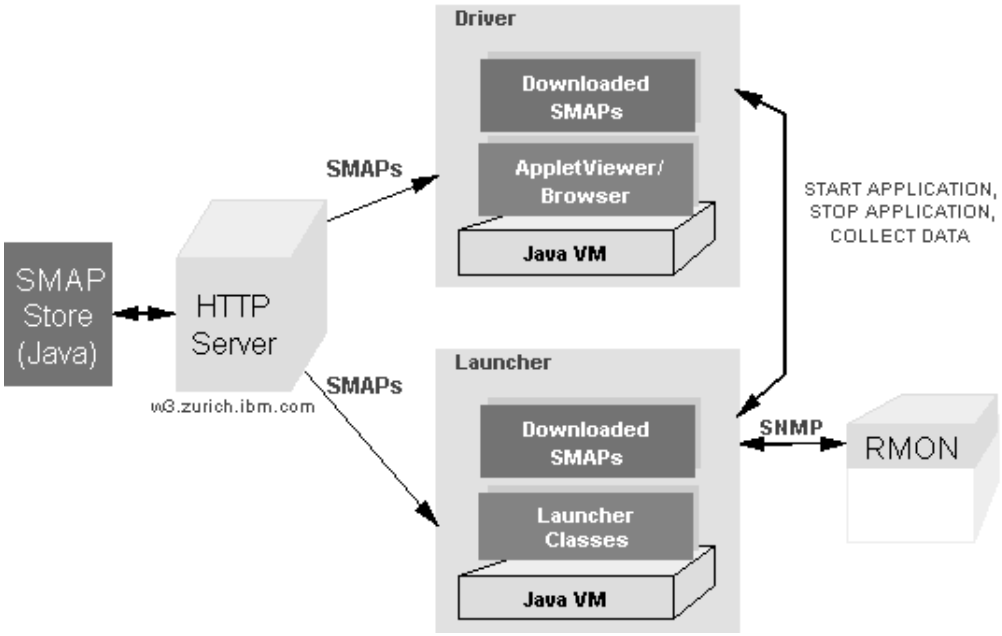

Figure 3 SMAP RMON Example

Example : LAN Traffic and RMON using the SMAP platform

Observation of packets on a LAN can prove helpful to network managers and designers. Traffic patterns between hosts or LAN segments provide useful information that can help produce more efficient LAN designs. Traffic data can allow network operators to pass some of the operating costs to network users based on usage patterns. Packet types observed on a network can also reveal potential problems that would otherwise be undetected by traditional network management software.

RMON (Remote Monitoring) MIB (Waldbusser, 1995) is an Internet standard for remote and distributed performance monitoring of ethernet and token-ring LANs. The statistics collected by an RMON probe is available through SNMP. Using customizable filters, an RMON probe can be used

to count packets based on packet characteristics (protocol, size, ...). RMON also provides platform independent means to capture packets.

We applied SMAP concepts to LAN management using RMON probes. We selected an initial set of four management problems, based on monitoring IP traffic on LAN segments:

1. segment usage: determine the number of packets local to, going out of, and coming into the LAN segment.
2. segment traffic per protocol: collect and count IP packets based on the IP protocol type.
3. HTTP server usage: collect and count HTTP packets on the same segment as the HTTP server to determine Web usage for selected LAN segments.
4. segment traffic patterns: capture IP packets and determine the top 10 source and destination pairs based on the number of bytes/packets exchanged.

As an example, the segment usage application is implemented as follows: we use a Web browser to load the user interface code, a Java applet , from an HTTP server. From the user interface, we select the management task (segment usage), the destination SMAP Launcher, and initiate the application by pressing the start button. This causes two actions:

1. first, a connection to the destination SMAP Launcher is established; and
2. second, a START_APPLICATION command is sent.

The instance of the ApplicationHandler class on the launcher site loads Java class for this application. Using a network class loader, all required Java classes are automatically loaded (SegTrafficUsage, Filter, Channel, and all SNMP classes). The driver can request data by sending COLLECT_DATA commands to the SMAPs, and terminates the management task by issuing a STOP_APPLICATION command.

Communicating SMAPs

The prototype implementation of the SMAP architecture uses a simple messaging scheme for inter-SMAP communications. This scheme is efficient, but it cannot support a richer form of communication between SMAPs. There are a number of alternatives. One is the use of Java facilities such as the Remote Method Invocation (RMI) API (Sun, 1996) or CORBA. In this approach, the communication between SMAPs is viewed as one between Java or CORBA objects, i.e., method calls. A second approach is to use intelligent agent technologies such as Aglets (Chang, 1996) to enable mobility as well as security.

4 CONCLUSION

Internet technologies such as the World Wide Web and the Java language will play an important role in the future of network and systems management. Already today, many in the industry are basing their next generation management applications on Web-based management (Jander, 1996).

In this paper, we have shown how the Internet technologies can be applied to network management. Webbin' allow networks to be managed through a web browser and provides a set of tools and programming interfaces to enable the development of network management applications. The SMAP approach supports distributed management, combining the web, Java and mobile agent technologies.

The pace at which Internet technologies change is remarkable; technologies change or new ones

introduced almost every week. We expect that these developments will not only enhance how we do network management, but keep us researchers and developers busy for a long time.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Lucas Heusler for his contributions to the SMAP work, Robert Akolk, Bela Ban, Dieter Gantenbein for contributions to Webbin', and the anonymous referees for their comments on the paper.

5 REFERENCES

Arnold, Ken and Gosling, James (1996). *The Java Programming Language.* Addison-Wesley, Massachusetts.

Berners-Lee, T., Cailliau, R., Groff, J. and Pollermann, B. (1992). World-Wide Web: The Information Universe. *Electronic Networking*, Vol. 1, No. 2, Spring 1992.

Chang, Dan and Lange, Danny (1996). Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW. Proceedings of the *OOPSLA'96 Workshop on Toward the Integration of WWW and Distributed Object Technology*, San Jose, California. October 1996.

Deri, Luca (1996a). Surfin' Network Resources across the Web. Proceedings of the *IEEE Workshop on Systems Management*, Toronto, Canada. June 1996.

Deri, Luca (1996b). Network Management for the 90s. Proceedings of the *ECOOP '96 Workshop on System and Network Management*, Linz, Austria. July 1996.

Deri, Luca (1996c). HTTP-based SNMP and CMIP Management. *Internet Draft* (draft-deri-http-mgmt-00.txt), December 1996.

Deri, Luca and Ban, Bela (1997a). Static vs. Dynamic CMIP/SNMP Network Management Using CORBA. Accepted for publication at *IS&N'97*, Como, Italy. May 1997.

Deri, Luca and Manikis, Dimitris (1997b). VRML: Adding 3D to Network Management. Accepted for publication at *IS&N '97*, Como, Italy. May 1997.

Jander, Mary (1996). Web-based Management: Welcome to the Revolution, *Data Communications*, November 21, 1996 issue, 39-53.

Object Management Group (1995). *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1995.

Sun Microsystems (1996). *Java Remote Method Invocation Specification*, Prebeta Draft, Revision 1.1, November 1996.

Waldbusser, Steven (1995). *Remote Network Monitoring Management Information Base*. Internet RFC-1757.

6 BIOGRAPHIES

Franck Barrillaud is a member of the Network Management Product Design Team at IBM CER,

La Gaude, France. He is working in the area of ATM network management development. Since 1996, he is leading the effort to introduce internet technologies into the network management products of the IBM Networking Division. His focus areas are in distributed and scalability aspects of network management and the introduction of artificial intelligence tols to solve network management complexity.

Luca Deri is currently a PhD student at the University of Berne, working part-time at the IBM Research Division, Zurich Research Laboratory. He received his degree in Computer Science with a thesis on Network Management from the University of Pisa. He worked at Finsiel S.p.A. as a consultant after the graduation, and was a research fellow at the University College of London. His professional interests include network management, software components and object-oriented technology. His Web page address is http://www.zurich.ibm.com/~lde/.

Metin Feridun is a research staff member at IBM Research Division, Zurich Research Laboratory since 1990. He previously worked for BBN Systems and Technologies (1987-1990), and GTE Laboratories (1983-1987). He received the S.B. degree (computer science) from Massachusetts Institute of Technology in 1978, the M.Eng. degree (computer engineering) from Rensselaer Polytechnic Institute in 1979, and the Ph.D. degree (electrical engineering) from Cornell University in 1983. His research interests are in distributed network and systems management.