EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

COMMUNICATION SYSTEMS DIVISION (SSC)
CH-1015 LAUSANNE, SWITZERLAND
http://sscwww.epfl.ch

Push vs. Pull in Web-Based Network Management

Jean-Philippe Martin-Flatin

Version 1: July 1998
Version 2: October 1998

Technical Report SSC/1998/022

Push vs. Pull in Web-Based Network Management

Jean-Philippe Martin-Flatin
EPFL-ICA, 1015 Lausanne, Switzerland
Email: martin-flatin@epfl.ch   Fax: +41-21-693-6610   Web: http://icawww.epfl.

Abstract

In this paper, we show how Web technologies can be used effectively to (i) address some of the deficiencies of traditional IP network management platforms, and (ii) render these expensive platforms redundant. We build on the concept of *embedded management application*, proposed by Wellens and Auerbach, and present two models of network management application designs that rely on Web technologies. First, the *pull model* is based on the request/response paradigm. It is typically used to perform data polling. Several commercial management platforms already use Web technologies that rely on this model to provide for ad hoc management; we demonstrate how to extend this to regular management. Second, the *push model* is a novel approach which relies on the publish/subscribe/distribute paradigm. It is better suited to regular management than the pull model, and allows administrators to conserve network bandwidth as well as CPU time on the management station. It can be seen as a generalization of the paradigm commonly used for notification delivery. Finally, we introduce the concept of the *collapsed network management platform*, where these two models coexist.

Keywords: Web-Based Management, Network Management, Push Model, Pull Model, Embedded Management Application, Collapsed Network Management Platform.

1. Introduction

If we consider the design of an IP network management application with a software engineering perspective, simple case of distributed application. There are no stringent requirements put on it, such as real-time constr tolerance, and some management data may even be lost. Its complexity stems from only two points: ther sometimes very large number of nodes to manage; and all management data traffic is considered as netwo and should therefore be kept to a minimum.

In the same perspective, if we analyze how IP networks are typically managed today, (that is, how network platforms are designed, how efficient is SNMP as an access protocol, and how efficient is the principle of inherent to the manager/agent paradigm), it is clear that most network management applications do not w comparison with modern distributed applications. Why not use object-oriented analysis, design and imp which are widely adopted by the industry today? Why be limited by the few existing SNMP protocol primitiv data from an agent? Why incur the network overhead of having the manager repeatedly tell every agent what MIB variables it is interested in, when this selection remains constant over time? Why not compress data effi it is transferred between agents and managers? Why use an unreliable transport protocol to send a critica management station when an interface goes down on a backbone router? Why make it so difficult to cross manage remote subsidiaries? Why are management data transfers so often insecure?

In light of the technologies widely used today, many design decisions in IP network management appear i outdated. But they did not in 1988-90, when the first SNMP framework was devised. Moreover, if we place a historical perspective taking into account how the market evolved [13], many deficiencies in today's network management platforms can be analyzed and understood. The success of SNMP-based network ma due to a large extent to its simplicity, so it would be unfair to criticize this simplicity afterwards. Still, the way are typically managed in practice evolved very little throughout the 1990s. If IP network management continu so slowly, it runs the risk of going from simple to simplistic. This could result in a plethora of alternatives bei by multiple vendors, and in the end of open integrated network management.

As we showed in earlier work [12], there are several alternatives to traditional SNMP-based manageme Web-based management, mobile agents, active networks, CORBA, intelligent agents, etc. In our view, Web are the best candidate for improving this situation in the short term. The reason for this is fivefold. First, the describe in this paper are simple, and could be engineered and widely deployed in less than a year; mo conversely, require secure environments (especially for WAN links) which no one can provide currentl simple, yet efficient multi-agent systems for IP network management still remain to be seen. Second, Web t have a limited footprint on network devices, unlike CORBA. Third, not only do Web technologies bring sol above-mentioned problems, as we demonstrate in this paper, but they also offer a smooth migration path, a k they are to be adopted by the industry. Fourth, they allow keeping a coherent single framework for op management, unlike WBEM. Fifth and last, the World-Wide Web has encountered lately a tremendous su enterprise world. Its simplicity, together with the portability of Java, have made it so ubiquitous that it is di to find any software engineering field that is not using (or migrating to use) one of its early technologies (W HTTP, HTML and CGI scripts) or one of its newer technologies (Java applications, applets, servlets, RMI Web expertise is rapidly developing worldwide, and it makes sense to capitalize on this in network managen

The idea of using the Web in IP network management is not new. Experiments with the early Web technolo Web browsers, HTTP, HTML and CGI scripts) started in 1993-94. Initially, they were only confined to seco For instance, people developed HTML forms to standardize and automate problem reporting, which facilita of calldesks. Network administrators also replaced daily, weekly and monthly printed reports with electronic on an internal Web server. More interestingly, administrators began writing symptom-driven HTML forms th could use for routine network troubleshooting; the interactive interfaces provided by the Web proved to be user-friendly than the thick binders full of procedures that operators were used to. When network equipmen tations were shipped in electronic format, they were put on internal Web servers; not only were they easier t administrators could then directly embed pointers to relevant pages of the documentation within symptom-dr pages. This integration of documentation, procedures and tools was a step forward in network troubleshootin

The first important step toward Web-based network management was taken when vendors began embed servers in their network equipment. Bruins [2] reports some early experiments made by Cisco in 1995, where command line interface was mapped to URLs. For instance, a Web browser cou <URL:http://router_name/exec/show/interface/ethernet0/> to a router, which would treat it as if the co `show interface ethernet0` had been typed in interactively. This opened new doors for c management and symptom-driven HTML forms, as there was no more need to `telnet` into netwo Mullaney [16] also describes work conducted by FTP Software, whereby agents send a static, locally dynamically generated HTML page back to the management station in response to an HTTP `get` or `post` r

The second important step was taken when Java applets appeared in Netscape’s famous Web browser, in 1 best of our knowledge, the new horizons that this technology opened up in network management were first p advertised in the July 1996 issue of The Simple Times. The founding article by Wellens and Auerbach [25 the concept of *embedded management application*, and showed the advantages of using HTTP rather tha vehicle data between managers and agents. Although the authors do not explicitly refer to applets in thei solution they propose is to transform an add-on (that has to be ported to many different management pl operating systems) into a single applet that can run everywhere. This applet is stored in the managed device, a by the administrator via a Web browser. Communication between the applet and its origin agent later reli instead of SNMP. Bruins [2] explicitly refers to applets in his description of prototype work by Cisco; but in he describes, once the applet is uploaded, subsequent communication with the agent relies on SNMP, not H is a poor use of applets as we will show in section 3.2.2.

Wellens and Auerbach’s applet-based approach has now been adopted by many network equipment vendors, HTTP servers and management applets in their equipment, but also by some network management platform v support Web browsers as front-ends to their network management platform.

Since the time of this proposal, many new technologies have appeared on the Web. Today, besides apple applications, we can also use servlets, RMI, etc. All these technologies open new possibilities and enable new network management applications. Leveraging on these new technologies, we propose to push Wellens and idea two steps further. First, we show that the design paradigm they propose is just one instance of a m paradigm, the *pull model*, which can not only be applied to ad hoc management, like they do, but als management. Second, we introduce a novel design called the *push model*. Unlike the pull model, it is not b request/response paradigm, but on the publish/subscribe/distribute paradigm. With this scheme, manag transfers are always initiated by the agent, like SNMP notifications delivery in pre-Web network manageme model reduces network overhead, and moves part of the CPU burden from managers to agents.

The remainder of this paper is organized as follows. In section 2, we present a summary of the main shor traditional SNMP-based network management, and outline how Web technologies can address them. In secti we present the engineering details of the pull model and the push model, and analyze the pros and cons of thre cation technologies: HTTP, sockets and RMI. Finally, we introduce the concept of *collapsed network* *platform* in section 5, and conclude with some perspectives for future work.

2. Problems with Traditional SNMP-Based Network Management

This section presents an overview of the problems encountered in traditional SNMP-based network manage IP network management before the Web days), and describes how Web technologies can address them. The can be grouped into four categories: network management platforms, protocol efficiency, security, and transp The terminology used in this paper, as well as the model of a network management platform on which ou based, are both presented in detail in [13].

2.1. Network management platforms

In a recent paper [13], we presented a brief history of IP network management before the Web days, showing came to use vendor-specific management GUIs (called *add-ons* when they are integrated in network platforms). This paper also details the shortcomings of IP network management before the Web. To summariz have four grievances: (i) network management platforms are too expensive, in terms of hardware and so should not be a need for dedicated hardware to manage networks; (ii) there should be unlimited support fo RDBMSs; today, customers are limited by the peer-to-peer agreements that have been signed, or not sign RDBMS vendors and network management platform vendors; if they want the latter to support another RDB happen to own already, they are charged enormous amounts of money for the "port"; (iii) for the sole purpos management[1], some customers must support a Unix system, although they run a business entirely based o Mac's; they want to use a PC or a Mac instead, but they do not want to buy a whole new (and expensi management platform.

The answer of Web-based network management to grievance (i) is the collapsed network management platf will gradually introduce in this paper. Grievance (ii) is addressed by JDBC, although there is a problem wi the poor execution speed of Java interpreted bytecode (even when speed-up techniques are used, such as the JI Grievance (iii) can be solved by the platform independence of Java and the universal interface offered by W

Network equipment vendors, on the other hand, are dissatisfied primarily by the huge costs they have to bea device-specific management GUIs for their equipment. To customers, a given GUI looks more or less the sam what management platform is used underneath. But to network equipment vendors, it does not. When a new GUI is released, the code has to be ported to many different operating systems (Windows 95, Windows 9 NT 4.x, Solaris 2.x, HP-UX 10.x, HP-UX 11.x...) and many different management platforms supporting di (HP OpenView, Cabletron Spectrum, Sun Solstice, IBM NetView...). Over time, despite the relatively small the stability of the major management platform vendors, the number of devices supported by each vendor anc of operating systems to port to have grown so large that the maintenance costs of these management skyrocketed.

With Web technologies, this problem is solved by applets, as we will see in section 3: the multiple incarn same add-on are all replaced with a single piece of code, the management applet, written in Java.

Customers and network equipment vendors share two other concerns. First, they both want the time- management GUIs to be reduced. When they purchase a brand new piece of equipment, customers want manage it immediately via their favorite management platform. But many months can pass between the network device that has been trumpeted by marketing is finally released and sold to customers, and vendor-specific management GUI has been ported to all operating systems and all existing network platforms. There are many environments where the constant availability of the network is critical to the smo of the business, and network equipment cannot be purchased unless it can be managed. So, for large compan peer-to-peer agreements with all major management platform vendors, there is a time window during which

[1] Until roughly 1995, Windows-based network management platforms were not powerful enough to manage large net large RDBMSs: in such environments, you had to buy a Unix system. Since then, the power of PCs has increased dram more than the power of Unix workstations. Customers who buy a management platform today are not exposed to anymore.

sell to these customers; this is a problem for customers and vendors alike. For small companies, and especially
companies specialized in cutting-edge technology, this problem is even worse. As their market share is close
are of no interest to management platform vendors, who do not bother signing peer-to-peer agreements with t
customers who want to buy from small companies are reduced to managing their network equipment
user-unfriendly MIB browsers, or with tailor-made software running on a dedicated PC sitting next to
Consequently, many markets are closed to such start-ups, which are desperate to get access to integra
management.

The second problem, which concerns customers and vendors alike, is versioning [16]. When upgrading a ver
MIB and consequently a vendor-specific management GUI, customers and network equipment vendors want
situations where the add-on integrated to the management platform has a different version level from that
supported by the agent. Today, since there is no such a thing as a MIB-discovery protocol, administrators e
manually specify what MIB is supported by what device, which is tedious, or they have to refrain themselve
MIB variables that have changed between the last and the previous MIB versions, which can cause problems

These last two concerns are again addressed by applets in Web-based management. Applet-based managem
is embedded in a network device when you buy it, so you can manage your agent at once. When you upgrade
on your agent, you can easily upgrade the management applet as well. And by transferring the management so
the agent to the manager, we ensure that the version of the vendor-specific MIB is always the same on both

2.2. Protocol efficiency

Since the outset, SNMP-based network management has been hampered by two protocol engineering deci
drastically reduce its efficiency. First, both SMIv1 [20] for the SNMPv1 framework, and SMIv2 [3] for the S
SNMPv3 frameworks, make the use of BER encoding [10] mandatory for SMI MIB data. Unfortunately, this
renown for its inefficiency. Mitra [15] and Neufeld and Vuong [17] describe this issue in detail, and show tha
of administrative data (identifier and length) transferred is very large compared to the actual data (content). S
itself does not mandate the use of any specific encoding rules, other more efficient schemes were defin
PER [11]. But they did not make their way through to the SNMP frameworks. The second issue is in SNMP i
varbind lists are relatively expensive, because the OIDs used to name variables usually take much more sp
values. Also, the absence of an efficient table retrieval mechanism means that the total protocol efficiency
repeated message exchanges (and repeated computations on the agent side).

These issues are addressed in Web-based network management by using HTTP 1.1 instead of SNMP to transf
data between managers and agents. The advantages are fourfold. First, this migration makes it possible to al
encoding, and to use instead a new MIME content type for SNMP, or simply encode SMI MIB data in
Second, the use of persistent connections [6], a key feature of HTTP 1.1, alleviates the network overhead
induced by multiple TCP connection setups and teardowns. Third, pipelining [6], another key feature of HTTP
the manager to make multiple requests without waiting for each response. This reduces latency, but also a
efficient use of TCP connections, when combined with persistent connections: if the time-out value of eac
connection is greater than the polling frequency for that agent, the same TCP connection can be used indefinit
the manager and each agent. Fourth, the network bandwidth usage can be reduced by performing tran
compression. Unlike SNMP, HTTP supports the MIME concepts of *content type* and *content transfer encodin*
data. Therefore, it is possible to compress the payload of an HTTP packet (say with `gzip`) on the agent, and
it on the manager, without the management application being even aware that data is compressed when it
Because the payload is plain text, the expected compression rate is fairly high.

The only problem not addressed by HTTP is the lack of an efficient table retrieval mechanism. This can be c
adding a new primitive to the new MIME content type mentioned above. RMI and Object Serialization o
solution, since they replace communication protocols like SNMP or HTTP with direct object-to-object com
SNMP varbind lists are replaced with serialized objects, and the absence of an efficient table retrieval mec
affects the agent, as we will show further on. But there are also problems with RMI, as we will see in sectio

2.3. Security

Security is a weak point of the SNMPv1 and SNMPv2 frameworks [22]. The lack of secure SNMP `get`'s an
hampered the management of remote subsidiaries for many years. With SNMP, how can an enterprise reason
a VPN spanning over the Internet or some kind of public network? Things have been significantly improved in

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE