Web Services-based network management: approaches and the WSNET system

John Soldatos\(^1\) and Dimitris Alexopoulos\(^2,*,†\)

\(^1\)Athens Information Technology, 19,5km Markopoulo Ave., PO Box 68, GR-19002 Peania, Greece

\(^2\)National Technical University of Athens, School of Electrical and Computer Engineering, 9 Heroon Polytechniou Str., GR-15773 Zografou, Greece

SUMMARY

While the Simple Network Management Protocol (SNMP) is still the dominant protocol for managing network elements in IP-based networks and the Internet, network managers are acknowledging its limitations with respect to configuration management, application development and decentralization of management tasks. Web Services (WS) have been recently proposed to alleviate these limitations, given their pertinence to both decentralized management paradigms (e.g., CORBA), and XML management systems which provide efficiency in configuration management operations. This paper reviews architectures for WS-based network management, outlining their advantages and disadvantages. These architectures address management of both individual network elements and composite multi-device networks. Moreover, the paper introduces the architecture of a prototype system for WS-based network management, namely WSNET. Along with presentation of the WSNET system, we provide a set of experimental results reporting performance figures for the WSNET system, as well as for systems based on other WS architectures. These figures allow for a comparative evaluation of the various systems, and manifest the benefits of the WSNET implementation. An important conclusion from our work is that WS should be seen as an accompaniment to conventional SNMP management rather than a replacement. However, there are also cases (e.g., need for secure remote access) where WS serve as a core rather than auxiliary solution, given that conventional methods are not applicable.

Copyright © 2006 John Wiley & Sons, Ltd.

1. INTRODUCTION

Since the early 1990s we have been witnessing a tremendous growth of the Internet, as well as the proliferation of IP-based networks. However, few advances have taken place in the area of network management for these types of networks. The conventional manager–agent interaction paradigm powered by the Simple Network Management Protocol (SNMP) is still the dominant network management solution. SNMP's prevalence is mainly due to its simplicity and penetration in the vast majority of network elements. SNMP has evolved from SNMPv1 to SNMPv2 and SNMPv3 [1]. However, network managers identify several limitations in SNMP-based management, the most prominent being its inefficiency in configuration management operations, application development complexity, as well as its lack of scalability for large networks [2]. Moreover, SNMP lacks essential security features (especially versions 1 and 2), and is not efficient in managing heterogeneous multi-vendor networks comprising a variety of Managing Information Bases (MIBs).

The limitations of SNMP have given rise to alternative management paradigms. Several approaches have been proposed to alleviate the centralized nature of SNMP towards improving the performance of
management operations while offering scalability for large networks. The most prominent decentralized management approaches are based on distributed object technologies such as CORBA (Common Object Request Broker Architecture) [3] and Java RMI (Remote Method Invocation) [4]. In the scope of these paradigms, information can be gathered from any location based on invocations of distributed remote objects on the target network elements. The main advantages of these distributed object technologies are that they allow management operations to be performed based on simple service-oriented APIs. On the downside, they are resource expensive for large object populations, resulting in suboptimal object retrieval. Moreover, they yield poor performance, along with response times that are inappropriate for real-time management operations. As a result, distributed object technologies are deemed more appropriate for service management rather than network management.

Another decentralized approach to network management is based on mobile software agents [5–7]. Mobile software agents save network bandwidth in cases where a mobile agent collects information from a device instead of performing a batch of network management requests to the device’s SNMP agent. Although mobile agents provide advantages for several network management scenarios, they do not constitute a solution that is simple or cost-effective to deploy.

Apart from decentralized approaches based on agents or distributed object technologies, there are also approaches offering management capabilities over the Web, using a web browser as a ubiquitous client interface [8]. This has the advantage of providing ubiquitous access to management information, while also boosting lightweight management consoles, instead of complex centralized consoles. Web-Based Enterprise Management (WBEM) constitutes the most popular paradigm and has been extensively deployed. WBEM provides efficient methods for manipulating management information in heterogeneous environments. This is because it is based on the Common Information Model (CIM), which consolidates a set of common management operations. CIM constitutes an abstract, vendor-independent representation of management information [9].

Recently, XML technologies have been proposed and are gradually adopted for IP network management. XML provides high-level semantics, which can alleviate SNMP’s inefficiency in bulk configuration management operations [10,11]. XML represents a standardized format for data representation and is therefore preferable to most proprietary Command Line Interfaces (CLI) used for configuration management. It is no accident that several equipment vendors (e.g., CISCO, NextHope, Juniper) have already integrated XML management capabilities into their products [12–14]. Also, XML-based management solutions come with a rich set of tools and techniques (surrounding XML), which can essentially accelerate and simplify application development. XML management techniques are in several cases deployed in a distributed fashion, allowing management requests and replies to be transmitted remotely using distributed technologies such as XML-RPC. Note also that several research and standards initiatives on XML-based management are in progress [15–17].

Web Services (WS) are closely related to XML and distributed object technologies, while also featuring analogies to other distributed technologies such as CORBA [18]. For example, WSDL (Web Services Description Language) is very similar to CORBA IDL (Interface Description Language), a URI is similar to CORBA IOR (Interoperable Object Reference), SOAP (Simple Object Access Protocol) is similar to CORBA GIOP (General InterORB Protocol), while UDDI (Universal Description and Discovery Interface) is similar to CORBA interface repository and naming service. At the same time, SOAP and WSDL are XML based. Nevertheless, WS also have differences from distributed object technologies, the main one being that they rely on a loose message passing between client and servers, rather than a very tight coupling.

WS have been proposed for network management and are currently studied both within standard bodies such as OASIS, IETF [15] and the Network Management Research Group (NMRG) [16], and within the industry. The interest in WS-based management is mainly due to a number of expected benefits. These benefits come in addition to network management decentralization. In particular, WS being a W3C standardized technology are envisaged as a way to reduce complexity based on open standards. Open standards along with associated tools and techniques can accelerate application development targeting
automation of management tasks. A WS-based management interface is accessible from a wide range of programming languages and toolkits that provide web services support (e.g., Microsoft .NET C#, J2EE, C/C++, Python, Perl, custom scripts in management platforms), which dramatically increases the scope of the potential client applications. As a direct consequence a WS-based management system can be easily integrated and/or interoperate with legacy network management systems. Moreover, XML/WS interfaces provide a rich set of options for securing the network management applications, such as SSL, HTTP Basic Authentication and WS Security.

WS-based network management systems have already been produced, mainly in the form of research prototype implementations, but also in the scope of industrial products exposing WS network management interfaces [19–22]. Exposing XML and WS interfaces is a trend recently adopted by several equipment vendors. Nevertheless, the WS network management model has not been widely adopted yet. Apart from mature implementations, there is also a need for investigating potential limitations. One of the main concerns relates to the performance of WS-based management solution.

Acknowledging the potential benefits of WS-based network management, but also the concerns associated with this model, this paper reviews architectures for WS network management, introduces a prototype implementation framework (WSNET) and performs a comparative performance evaluation of SNMP, XML and WS solutions. We concentrate on scalable architectures that provide the means to managing legacy networking infrastructures consisting of numerous SNMP-enabled devices. Presented architectures target not only element-level network management solution (i.e., managing an individual device), but also solutions that are appropriate for composite multi-device networks. Indeed WSNET, which was prototyped and evaluated, is a scalable solution that can be applied in composite IP-based networks.

The rest of the paper has the following structure: Section 2 presents a set of architectures for network management based on WS. All these architectures address SNMP-enabled devices, which makes them applicable to the vast majority of legacy networks and devices. Section 3 reports on the prototype implementation of some of the architectures, outlining tools and platforms used, while also including performance metrics and evaluation of WS solutions. Section 3 emphasizes also the WSNET design and implementation, and presents evaluation results in comparison to other approaches. Following this evaluation section, Section 4 presents practical examples manifesting the merits of WS Management and WSNET over conventional SNMP. Finally, Section 5 draws basic conclusions on the potential applicability and adoption of WS-based management.

2. WEB SERVICES ARCHITECTURES FOR NETWORK MANAGEMENT

WS-based network management systems rely on SOAP engines capable of processing XML/SOAP management commands. Depending on the relative placement of the SOAP engines with respect to the network element, several architectures for WS network management can be devised. In the sequel we elaborate on architectures that are based on the popular manager–agent interaction paradigm. These architectures rely on the SNMP protocol for accessing the low-level management capabilities of the devices. In particular, even though management commands are conveyed as SOAP messages, they are ultimately executed on the network elements as SNMP commands. Leveraging the SNMP agents of the network elements presents two distinct advantages:

- It allows WS management to be applied to the vast majority of existing network devices, thus rendering the architectures applicable to legacy networking environments.
- It exploits the benefits of WS-based interfaces relating to their high-level semantics, integration and security features.

The architectures presented in the sequel address management of individual network elements, but also of multi-device networks.
2.1 SOAP Engine Embedded on the Device

The simplest approach to managing devices based on WS is the direct or embedded approach depicted in Figure 1(a). According to this approach a SOAP engine is embedded in the network element, allowing the element to receive commands based on an XML/SOAP interface. Based on this embedded SOAP engine the device receives management commands in the form of SOAP messages and accordingly executes them on the network element. Execution is based on the mapping of SOAP commands to SNMP commands. Realizing such a mapping hinges on exporting basic SNMP commands (e.g., get(), set(), getnext()) as WS. The parameters of these WS are encapsulated in the SOAP messages, and accordingly executed by the SNMP agent of the device. This approach can be deployed incrementally on top of every SNMP-enabled device.

The embedded/direct approach is oriented to management of individual network elements (i.e., the Element Management Layer (EML)). Several vendors expose WS management interfaces to their products. The advantage of this approach is that it exports a remote networked interface that can be directly used for loose integration with other systems, as well as for distributed management. The XML/SOAP interface allows distributed managers to issue commands targeting the particular network element. These commands do not suffer from security restrictions imposed by firewalls given that they are based on the HTTP protocol.

On the downside, the embedded approach is limited to managing individual devices. It is not appropriate for issuing WS commands for multiple devices. Addressing a multi-device environment would require the execution of multiple WS commands on each of the devices. Another concern with the direct approach relates to the security of the management commands through the SOAP interface. In particular, implementation of WS security features is required.

The embedded solution increases the cost of the network element. This is due to additional costs for embedding the SOAP engine in the device. A lower-cost option is to host the SOAP engine in a workstation attached to the device. Nevertheless, using an attached workstation may result in poor performance in terms of the execution/response time for the management commands.

2.2 Embedded SOAP Engine with Higher-Level Interface

A variation of the embedded approach adds a higher degree of sophistication to the XML/SOAP interface exposed to remote managers. Instead of the straightforward mapping of WS requests to SNMP commands of these WS are encapsulated in the SOAP messages, and accordingly executed by the SNMP agent of the device. This approach can be deployed incrementally on top of every SNMP-enabled device.

The embedded/direct approach is oriented to management of individual network elements (i.e., the Element Management Layer (EML)). Several vendors expose WS management interfaces to their products. The advantage of this approach is that it exports a remote networked interface that can be directly used for loose integration with other systems, as well as for distributed management. The XML/SOAP interface allows distributed managers to issue commands targeting the particular network element. These commands do not suffer from security restrictions imposed by firewalls given that they are based on the HTTP protocol.

On the downside, the embedded approach is limited to managing individual devices. It is not appropriate for issuing WS commands for multiple devices. Addressing a multi-device environment would require the execution of multiple WS commands on each of the devices. Another concern with the direct approach relates to the security of the management commands through the SOAP interface. In particular, implementation of WS security features is required.

The embedded solution increases the cost of the network element. This is due to additional costs for embedding the SOAP engine in the device. A lower-cost option is to host the SOAP engine in a workstation attached to the device. Nevertheless, using an attached workstation may result in poor performance in terms of the execution/response time for the management commands.
commands, this scheme exports an XML/SOAP interface with higher semantics (Figure 1b). This higher-level interface allows clients/managers to issue composite EML management commands that comprise a variety of individual SNMP commands that execute in an atomic manner. This is realized based on a mapping of higher-level commands to individual SNMP commands. While such a mapping provides access to more sophisticated functionality from the XML/SOAP interface, it introduces significant implementation complexity relating to handling multiple SNMP commands as a single atomic management operation. As a characteristic example, sophisticated error handling is required, given that a failure in a single SNMP command results in a failing composite command. Thus, exporting a higher-level interface complex may increase the complexity and ultimately the cost of the network element.

2.3 SOAP Engine Controlling Multiple Devices

Figure 2 depicts an alternative scheme to exposing a WS network management interface for a network device. The scheme is in principle similar to the direct approach, with the exception that the SOAP engine is not embedded in every device. Rather, it resides in a Network Management System (NMS) that controls multiple network elements. This approach does not require that the devices expose an XML/SOAP interface. This is because it is the NMS system that provides functionality for parsing XML/SOAP messages and accordingly mapping them to SNMP commands of the target device. To this end, the NMS has access to the Management Information Bases (MIBs) supported by the devices.

The advantage of this approach is that it is appropriate for offering a WS interface to any SNMP-enabled device. Also, it is a low-cost method, given that the SOAP engine and additional parsing/mapping software is not likely to essentially increase the cost of the NMS. Furthermore, it offers the possibility for controlling/managing multiple devices from a single point, based on WS.

The downside of this solution is that the SOAP engine constitutes a possible single point of failure, which compromises the availability of the WS interface. Note also that the performance of this solution may be questionable, as the number of controlled devices increases. As far as security requirements are concerned, these are similar to those of the embedded approach.

2.4 XML-Based Architectures

Several Extensible Markup Language (XML)-based network management techniques have recently emerged toward overcoming the limitations of SNMP management in configuration operations. Since XML documents can have much higher-level semantics than SNMP, they are more appropriate for bulk configuration operations. Exporting XML interfaces for network management has also other advantages since:

![Figure 2. Single SOAP engine for multiple devices](image)
• XML is easy to generate, parse and process, supports sophisticated data structuring, and can therefore handle complex organizations of management information.
• XML DTD (Document Type Definitions) and XML Schemas specify and validate the structure of XML documents, thus alleviating developers of network management applications from tedious tasks.
• XML comes with numerous W3C technologies supporting rapid development of network management applications (e.g., Extensible Stylesheet Transformations (XSLT), which transform XML documents to other XML formats and XPath/XQuery discovering XML elements subject to criteria).

XML management systems accept XML documents containing management commands and accordingly execute them on the network. Hence, XML management systems expose XML interfaces to manager applications. Formatting network management commands as XML commands eases the authoring of management scripts and applications [23], while increasing the programmability of management operations [24,25]. Moreover, it provides potential for standardizing the management interfaces through specifying management commands in the form of XML Schemas or DTDs.

The low-level execution of management operations on devices based on XML management systems is performed by either XML or SNMP agents [17,26]. Managing SNMP-enabled devices based on XML architectures requires a mechanism for conveying XML management commands to SNMP commands, as well as a scheme for mapping XML-based management information to SNMP MIBs [27]. Special modules called XML/SNMP gateways undertake to perform these mappings. Several XML/SNMP gateways have been developed mainly in research labs to support the emerging XML management paradigm [17,28]. Gateway modules make extensive use of methodologies and techniques surrounding XML (e.g., DOM, SAX, XPath, XQuery) toward translating XML commands and XML information structures to conventional SNMP commands and MIB objects [28,29]. Figure 3 depicts the building blocks of an XML management system comprising an XML gateway and exporting an XML interface to the manager.

SOAP, being a standard XML protocol for sending messages over HTTP (or SMTP), is perfectly tailored to XML management systems. SOAP allows XML managers to convey management command to XML management systems over HTTP. In a typical scenario, a SOAP client (i.e., manager application) generates XML-encoded SOAP messages, which are accordingly by a gateway module within or attached to the XML/SNMP gateway module. The SOAP client request/message contains Remote Procedure Call (RPC) information, which is taken by the manager application. This message is passed to an HTTP client, which sends it to an HTTP server at the gateway side. The ultimate processing is performed by the SOAP server, which determines the (remote) object that needs to be executed at the XML/SNMP gateway module. Accordingly, the result follows the reverse direction reaching the remote client and finally the manager application.

Note that Figure 3 concentrates on XML management for single network elements. A more sophisticated structure of the XML information models, as well as an extension of the XML management commands, can allow for higher-level XML/SOAP interfaces addressing multi-device networks. These higher-level interfaces consist of XML documents comprising composite XML commands for multiple devices, which can accordingly be executed on various individual element-level XML management systems, as shown in Figure 4. Each of the XML systems in Figure 4 features a structure similar to the element-level system presented in Figure 3(a). The implementation complexity of such network-level XML management systems is significantly higher, given that the execution of composite management commands raises a host of issues relating to handling of failures, security and performance, while also asking for effective and intelligent techniques for structuring management information. Consequently, few efforts have up to date addressed the implementation of such systems.

XML management systems with SOAP interfaces constitute an emerging architecture for WS-based network management. This architecture leverages the advantages of XML systems with respect to the programmability and cost-effectiveness of the management operations. Moreover, they alleviate SNMP limitations in configuration management. The drawbacks of XML-based architectures lie in the
processing overhead incurred by the parsing and construction of XML messages, which can lead to serious performance penalties. XML/SNMP gateways tend to increase response times for management operations. While this is a disadvantage of XML management per se, it becomes less important when SOAP interfaces are used. This is because the fact that XML/SNMP gateways deal with XML documents is, in several cases, eliminating the additional overhead imposed by SOAP. SOAP constitutes a more natural interface for XML management systems that already handle XML messages. This is a significant incentive for using SOAP in XML systems. Note that SOAP interfaces are anyway very likely to proliferate, since SOAP is an open standard with a growing community of developers and vendors supporting it.

XML systems are expected to be deployed more and more in the future, as soon as they mature. It is also important, however, that security issues (i.e., authentication and encryption of management messages) are addressed. Authentication and encryption of XML messages is a subject of intensive work, within the W3C (e.g., in the XML Signature Working Group), and OASIS (e.g., the Security Services Committee). Moreover, SNMPv3 includes authentication and encryption mechanisms (reflected in IETF RFCs...
2271–2275). As a result, the missing piece is a framework for mapping XML security mechanisms to SNMPv3.

2.5 Multi-Tier Web Services Management Systems

Figure 5 depicts a WS-based network-level management architecture, combining the individual architectures discussed in the previous paragraph. According to this architecture, a network-level management system executes network-wide operations exploiting WS-based management systems at various levels, in particular:

- element-level management systems, being either XML based or simple SNMP systems featuring a SOAP interface;
- network-level management system, such as XML-based Network Layer Management Systems.

We call the architecture depicted in Figure 5 an N-tier Web Services network management system. This system aggregates WS exposed by underlying management systems, and exports a higher-level interface enabling management of multi-vendor multi-device networks. The underlying WS management architectures are totally transparent to the network management system, which only deals with the SOAP interfaces and the functionality exposed by lower-layer systems. As an additional feature, WS of lower-level systems can be registered with a UDDI registry, allowing the network management system to dynamically discover network management services and composing higher-level services and applications. The main benefit of this architecture is that it exposes a high-level WS interface enabling the execution of management commands dealing with a variety of devices within the target network. Moreover, this architecture exploits the capabilities and sophistication of underlying individual systems. Nevertheless, such an architecture is complex to implement given the significant effort required for structuring heterogeneous (i.e., multi-vendor, multi-device) management information (e.g., some of the challenges are discussed in Reference [30]), but also the number of different distributed systems (XML/SNMP gateway, SOAP engines, HTTP servers, network elements) involved. Moreover, complex operations require that failure handling accounts for failures in the individual management systems, while a complete security solution demands that security is addressed in every individual subsystem. The implementation complexity implies a high cost for such an implementation.

Figure 5. Multi-tier network management system
3. IMPLEMENTATION AND PERFORMANCE EVALUATION

3.1 Web Services Network Management Implementation (WSNET)

A proof-of-concept implementation referred to hereafter as WSNET was undertaken regarding the direct use of WS in network management. In more detail the overall system can be categorized in the element-level management (ELM) and the network level management (NLM) systems.

WSNET Element-Level Management

At the element level management consists of the following subsystems, as depicted in Figure 6:

- Apache AXIS SOAP engine;
- the Java implementation of WSNET WS;
- Java to SNMP interface through Sun’s JMAPI library and a Java SNMP wrapper;

At the element-level management of WSNET a SOAP engine (namely Apache AXIS) is attached to the managed network element and used to invoke over HTTP the exposed management WS. In detail, all clients/remote managers use functions snmpGet, snmpSet and snmpGetNext exposed by the WSNET WS. On the back-end the WS handles the request and makes the appropriate SNMP call to the requested Network Node, utilizing either the XML/SNMP or Java/SNMP interface according to the request made by the client/remote manager. Finally the response is provided to the client wrapped as the standard SOAP protocol defines.

WSNET Network-Level Management

Regarding the network-level management system of WSNET, it is built utilizing the underlying element-level management. In detail, as depicted in Figure 7, more complicated functions are exposed as a WS to clients/remote managers using the same SOAP engine (Apache AXIS). More specifically, a function named invoke() is implemented and accepts as arguments a list of simple SNMP get and set commands.

Figure 6. Overall architecture of WSNET at element-level management
The WSNET NLM service processes the request, passing the SNMP commands to the WSNET ELM system for further processing and finally passes the results to the client/remote manager.

3.2 Web Services Interface to an XML-Based Network Management System

Another prototype implementation concerned a SOAP/WS interface to the XMLNET system [24,31], which implements high-level XML network management interfaces based on the paradigms depicted in Figures 8 and 9. The XMLNET system includes an element layer management subsystem allowing multiple element-level operations included (in an appropriately structured XML document) to be executed on a single network device. At the heart of this element-level system lies (Figure 8) an XML management engine that is capable of parsing XML documents containing multiple atomic management commands (i.e., commands corresponding to multiple individual SNMP operations), and accordingly mapping them to the respective SNMP operations. To this end, we exploited the SXG XML/SNMP gateway by Klie and Strauss [28]. Thus, the element-level subsystem of the XMLNET system exposes a high-level interface to SNMP network elements. Note that the XMLNET system also provides a higher-level XML interface that can process more complex XML documents containing management operations concerning composite multi-vendor multi-device networks. These XML documents are then broken down into finer XML documents that can be executed by element-level XMLNET subsystems, as shown in Figure 9. A more detailed presentation of the higher-level management capabilities of XMLNET is provided in Reference [32].

3.3 Performance Evaluation of the Web Services-Based Management Schemes

In a nutshell, our implementation work resulted in two alternative implementations of systems for WS management on SNMP-enabled devices. Based on the SOAP interfaces we measured response times corresponding to the execution of multiple individual SNMP operations, namely 3, 5, 8, 10 and 15 requests, using the WSNET system conforming to the WS architectures described in Sections 2.1–2.3, and the XMLNET system utilizing a WS interface to an XML-based architecture (described in Section 2.4). Toward deriving a meaningful comparative benchmark, we also provide the time required to complete the same
operations through natively invoked Java-SNMP calls as well as utilizing only the XML-based system described in Section 2.4 (i.e., performance of layers lying beneath the WS-SOAP interface). All the operations corresponded to SNMP `get()` operations of standard MIB II objects. Note also that measurements constitute averages over 50 experiments to account for statistical fluctuations stemming from the utilization of hosts and the network load. Emphasis should be given not to the absolute values, but rather...
to the relative overhead introduced by WS. This is because implementations were carried out as research prototypes and using technologies and techniques (e.g., Java, Java-SNMP, AXIS, prototype XML/SNMP gateway) that introduce significant delay overheads. We expect absolute figures to be significantly lower in the scope of industrial embedded high-performing implementations.

Figure 10 illustrates the response times measured for management operations executed over the SOAP interfaces using either the direct WSNET approach or the WS interface to the XMLNET element-level system. Moreover, the response times for the same operations using a conventional Java-based SNMP interface as well as for the standalone XMLNET system are presented. Response times show that carrying XML messages over the network along with SOAP processing results in a significant additional delay amounting to more than 200 ms. While this is a significant performance overhead, it is to some extent expected given that SOAP calls are performed remotely, therefore introducing a network latency factor. Note also that the management information transferred through the SOAP is structure in the form of SOAP-compliant messages, which results in large chunks of networked data even for a simple SNMP operation. As the number of operations increases, the percentage of useful information contained in the exchanged SOAP messages increases. It is therefore good practice to encapsulate multiple SNMP operations within a single SOAP message. This is shown in Figure 11, which illustrates the percentage of additional delay introduced by each one of the methods as the number of operations increases.

Results shown in Figure 11 demonstrate also that SOAP is a proper interface for XML-based network management systems. This is because XML management systems have anyway to deal with the overhead of parsing XML documents and therefore suffer less from a SOAP-based interface. In general, SOAP is much better for implementing high-level management interfaces conveying management commands that consist of numerous individual requests.

Given the significant performance penalty of SOAP-based implementations compared to conventional SNMP implementation, network managers should deploy and use SOAP interfaces only when they really need to exploit the benefits of WS. For example, WS interfaces should be used when remote access to the management system is needed, when loosely coupled integration with other network systems is required, or when an XML-based network management system is used anyway. Using WS whenever they provide true value is perfectly in line with our approach for suggesting WS as a technology that complements rather than replaces conventional SNMP-based management.
4. PRACTICAL EXAMPLES

4.1 Overview of Web Services Added Value over SNMP

The benefits of WS management over conventional SNMP management can be summarized in the following points:

- WS offer a variety of standard access interfaces: in a WS management environment network operations can be invoked programmatically through a wealth of open source and commercial frameworks providing infrastructure for such an invocation (e.g., Java, .NET/C#, Python). Note also that a SOAP interface to WS allows for W3C standards-based access to management operations based on a flexible XML-based protocol.
- WS allow for secure remote access to management operations: several applications may exploit plain HTTP interfaces to WS-based management operations, to ensure that that management traffic survives firewalls. This is essential in several cases where there is a need to manage a network from remote (e.g., over the public Internet).
- WS approaches can provide high-level management interfaces: structuring management operations as WS allows for the development of a multilayered management architecture. Simple operations can be assembled in a higher level in order to construct more complicated, value-added management operations. Consequently, network application development becomes more flexible, since reusability across management operations is exploited.

Figure 12 illustrates the various interfaces provided by WSNET toward invoking management operations, whereas the following paragraphs illustrate practical use cases that leverage the above-mentioned benefits.

4.2 Use Case Example: Ubiquitous Secure Remote Access

WS-based management and WSNET allow for ubiquitous secure remote access to network management services. In several cases operators, service providers, as well as enterprise networking consultants, need to access geographically dispersed Network Management Systems (NMS) (e.g., toward handling problematic situations, as well as toward performing scheduled operations). Conventional solutions for secure remote access to geographically dispersed NMS systems include dial-up connections and IPSec Virtual Private Network (VPN). These solutions cannot be effective, however, when there is a problem or failure.
within the NMS. In such cases, network managers have to inspect and manage individual network elements. While SNMP provides the option for accessing network elements remotely, it lacks essential security features, given that it provides only loose authentication features. Also, most firewalls are likely to drop SNMP traffic. In such cases the WS paradigm provides a salient solution, since it can offer secure remote access to management operations. Also, WS interfaces operate over HTTP and are not blocked by firewalls.

4.3 Implementation Example: Value-Added Configuration Management Operations

As also demonstrated in Reference [32], WS interfaces can have high semantics toward structuring composite operations that include several element-level operations. An example of a composite operation is illustrated in Tables 1 and 2, which depict SOAP messages for element- and network-level operations, respectively. These messages correspond to the creation of an ATM virtual path connection (VPC). WSNET exposes high-level network management interfaces comprising several element-level interfaces. This is a value-added feature of WS, especially in the scope of configuration management operations that entail a host of individual element-level SNMP operations. Nevertheless, such a composition entails several challenges such as error handling (i.e., how a failure of an elementary operation comprising a composite operation is handled). These challenges are outlined in Reference [32], along with solutions that have been implemented and tested to support robust composite operations.

5. CONCLUSIONS

In this paper we presented the most prominent architectures for managing networks based on the WS paradigm. The emphasis was on architectures that can support the vast base of SNMP-enabled devices. Some architectures hinge on exporting SNMP operations as WS, while there is also a class of emerging XML-based network management systems that can be easily enhanced with SOAP/WS interfaces. Note also that several architectures support single atomic SNMP operations, while others support higher-level management operations featuring high-level semantics (e.g., spanning multiple network devices).
The latter offer flexibility and enhance the programmability of network management operations, while being appropriate for configuration management. Nevertheless, higher-level management interfaces demand significant development effort, which is a factor inhibiting their wide development and penetration. Also, these architectures need to mature technically, while having security issues resolved. It is no accident that most vendors to date offer SOAP interfaces only at the element level (i.e., on individual elements). However, the advent of XML-based network management systems along with the intensive work toward improving WS security can significantly boost the development of higher-level WS interfaces.

Apart from presenting architectures for WS management, we also conducted a set of prototype implementations and measurements aimed at providing insight on the performance of WS for management operations. Results showed that WS incur a significant performance penalty. This, however, is minimized as the number of management operations conveyed within SOAP management messages increases. Moreover, WS interfaces are perfectly tailored to XML systems without any essential degradation of their performance. This is highly due to the fact that XML systems have anyway to parse, process and construct XML documents.

Performance analysis results suggest also that WS should be used when they offer true benefit and added value (e.g., distributed remote management capabilities). This is fully aligned to the view that WS do not come to replace but to support conventional SNMP management. However, there are indeed many practical examples demonstrating the added value of WS approaches over SNMP mechanisms. In this paper we have elaborated on one such example dealing with secure remote access to geographically dispersed network elements. This is required, for example, in order to deal with a failure of an NMS. In such scenarios, a WS-based approach is salient, since conventional dial-up and IP VPN solutions are not applicable.

Table 1. Element-level Web Service invocation SOAP envelope

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:elementOperation
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
      xmlns:ns1="http://soapinterop.org/">
      <ns1:arg0 xsi:type="xsd:string">10.20.0.58</ns1:arg0>
      <ns1:arg1 href="#id0"/>
    </ns1:elementOperation>
    <multiRef id="id0" soapenc:root="0"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
      xsi:type="ns2:Vector"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
      xmlns:ns2="http://xml.apache.org/xml-soap">
      <item xsi:type="xsd:string">system/sysName,.1.3.6.1.2.1.1.5,S,get,i_seq=1,
dt=0</item>
      <item xsi:type="xsd:string">channelUpc,I,set,value=1,seq=2</item>
      <item xsi:type="xsd:string">channelVPIIn,I,set,value=1,seq=3</item>
      <item xsi:type="xsd:string">channelVPIOut,I,set,value=2,seq=4</item>
      <item xsi:type="xsd:string">channelValidateStatus,I,set,value=5,
  seq=5</item>
      <item xsi:type="xsd:string">system/sysUpTime,.1.3.6.1.2.1.1.3,T,get,seq=6,
dt=100,if_greater=100000,if_exec=sh doFinalCheck.sh</item>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

Copyright © 2006 John Wiley & Sons, Ltd.

DOI: 10.1002/nem
Table 2. Network-level Web Service invocation SOAP envelope
REFERENCES

AUTHORS’ BIOGRAPHIES

Dimitris Alexopoulos was born in Athens, Greece, in 1979. In 2002 he received the diploma of electrical engineering and computer science from the National Technical University of Athens (NTUA). He has worked as software engineer for INTRACOM S.A and INTRASOFT INTERNATIONAL S.A. Since 2003 he is a PhD candidate and a research associate at the NTUA (Telecommunication Laboratory). His research interests are distributed and programmable network management; service oriented programming, grid computing and web services. He has published journal and conference papers in these areas.

John K. Soldatos was born in Athens, Greece in 1973. He obtained his Dipl-Eng. degree in 1996 and his PhD in 2000, both from the National Technical University of Athens. He has had an active role in several EU co-funded research projects (EXPERT AC-094, WATT AC-235, IMPACT AC-324, Chameleon EP 20597, CATCH-2004 IST-1999-11103, and LION IST-1999-11387), and is now involved in the CHILFP6-506909 project. He has also consulted in many ICT projects for leading Greek enterprises, with the roles of project manager, team leader and chief technical architect. He has co-authored more than 65 papers published in international journals and conference proceedings. Dr. Soldatos serves as reviewer in numerous magazines, journals and conferences, and expert evaluator for ICT tenders. Since March 2003 he is with Athens Information Technology, where he is currently an Assistant Professor. His current research interests are in Network Control and Management, Grid, Pervasive/Ubiquitous and Autonomic Computing.