On the Future of Internet Management Technologies

Jürgen Schönwälder, International University Bremen
Aiko Pras, University of Twente
Jean-Philippe Martin-Flatin, CERN

ABSTRACT

As the Internet continues to grow, it becomes more and more apparent that existing Internet management technologies need to be improved, extended or replaced in order to extend functionality and reduce development time and operational costs. Within the IETF, IRTF, and IAB, several new approaches are currently under discussion. Evolutionary approaches aim at improving currently used technologies, whereas revolutionary approaches try to replace existing management-specific technologies with standard distributed systems technologies. This article surveys the research and development work under way to develop future Internet management technologies.

INTRODUCTION

The standard management framework currently used in the Internet is named after its main building block: the Simple Network Management Protocol (SNMP) [1]. It was devised in the late 1980s and is widely supported by network devices. SNMP is a special-purpose management protocol that can be used to read and write simple typed variables. The software component that handles the associated Get/Set requests and accesses the internal data structures on managed devices is called an *agent*. In addition to processing such requests, an agent can also generate notifications under certain circumstances and send them as unsolicited messages to the management application (*manager*). This architecture is known as the *manager-agent paradigm*.

Concrete data models for managing specific technologies or protocols are defined and standardized in management information base (MIB) modules, which are written in a language called Structure of Management Information (SMI) [2]. SMI is a data-oriented language based on Abstract Syntax Notation 1 (ASN.1). It requires that complex nested data structures be normalized into a set of interrelated conceptual MIB tables. It does not currently support concepts such as structured data types, objects, or methods.

Although SNMP technology is now well understood and widely deployed, it is still confined to network devices and rarely used for managing systems (PCs, servers) or applications. Even within network element management, there are several functional areas where SNMP has played only a minor role so far (e.g., configuration management).

Since SNMP technology is primarily used in a small number of management areas, it is not surprising that a number of alternative technologies have been proposed recently. This article surveys these proposals and presents the results of related discussions that took place within the Internet Engineering Task Force (IETF), Internet Research Task Force (IRTF), and Internet Architecture Board (IAB).

The rest of this article is organized as follows. We start by presenting the mismatch between the requirements of Internet network operators and the way SNMP has evolved since its inception. Next, we describe evolutionary approaches to improve the SNMP framework. Last, we review more revolutionary approaches based on Extensible Markup Language (XML) and Web services.

MANAGEMENT BACKGROUND

Some background information is necessary to understand and assess the relative merits of different proposals for new or enhanced management technologies. First of all, the requirements expressed by Internet network operators must be understood by protocol developers and application implementors. A number of nonfunctional aspects also have an impact on the selection of network management technologies. The main nonfunctional aspect is the environment in which management takes place. Other key aspects include market and standardization.

Requirements of Internet Network Operators

The Operations and Management Area of the IETF organized several meetings in 2001 to identify and outline a set of requirements for Internet network operators in order for management protocol and application developers to better meet their needs. In June 2002 the IAB organized a workshop on the configuration aspects of network management [3].

During these meetings, it became clear that, from the operators' standpoint, configuration management is the most important problem to be addressed to date. Operators of large backbone networks maintain their network-wide configuration data in a logically centralized database, as depicted in Fig. 1 [4]. Change requests leading to configuration changes in network devices (e.g., new routing policies) trigger transactions on the logically centralized database. Once a new network-wide configuration has been established in the database, complete configuration files or incremental configuration updates for specific network devices are first generated by a configuration data translator, then distributed to all devices, and finally activated. It is not unusual for Internet network operators to write these translators themselves. Due to a lack of well established standards, network operators have to update their translators when new network devices are released, or when new firmware needs to be installed in already deployed devices.

The requirements of the Internet network operators can be summarized as follows:

- It is crucial to make a clear distinction between configuration data (which is rather static) and data that describes operational state (which is dynamic by nature).
- There must be basic operations to download and upload complete configuration files. It is desirable to be able to download or upload only parts of the configuration data.
- The configuration data should be in a textual format to allow the usage of a wide range of text-processing tools (e.g., the UNIX command diff) and version management systems.
- It is necessary to distinguish between the distribution of configurations and the activation of a certain configuration. Devices should be able to hold multiple configurations and enable management applications to activate any of them (only one configuration is active at a time).
- The coordinated activation of configurations could be dramatically simplified by having a transaction mechanism for uploading new configurations and activating them "simultaneously" on multiple devices. Such a transaction mechanism must take into account that connectivity might be lost in the middle of the transaction.
- Finally, ease of use of the management technology is of paramount importance. Configuration management interfaces must be designed such that developing and debugging configuration data translators is cost effective.

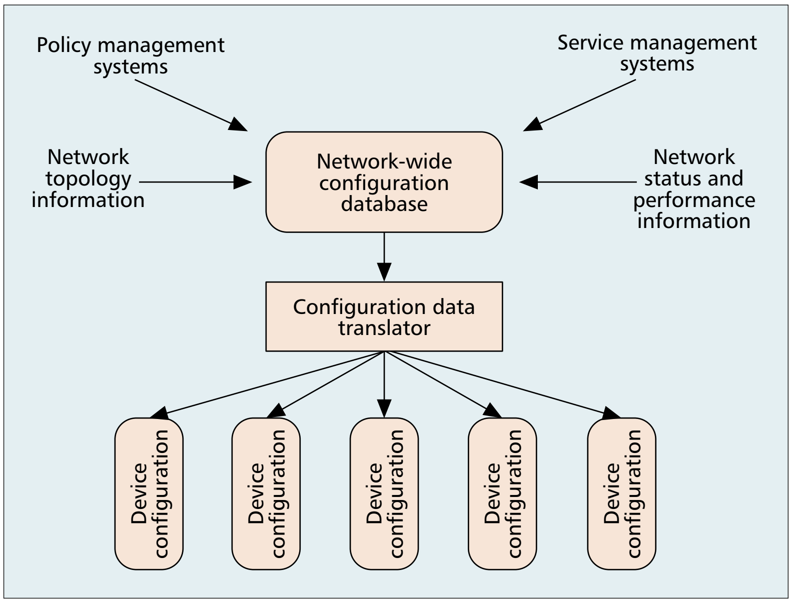

Policy management systems
Service management systems
Network topology information
Network-wide configuration database
Network status and performance information
Configuration data translator
Device configuration
Device configuration
Device configuration
Device configuration
Device configuration

Figure 1. *A configuration management model.*

Management Environment

The SNMP framework was designed to:

- Minimize the number and complexity of management functions realized by the agents
- Be extensible to accommodate additional and unanticipated aspects of network operation and management
- Be as much as possible independent of the implementation of particular hosts or gateways [5]

As a result, the main strengths of SNMP are its simplicity, interoperability, and low footprint on agents [6].

SNMP must also work effectively when the network is not fully operational. This reflects in the selection of a connectionless transport protocol (UDP), which allows management applications to exercise full control over the retransmission strategy.

Another design choice was to keep SNMP as independent as possible of other network services. This is one of the main reasons why, in SNMP version 3 (SNMPv3), security is self-contained and does not rely on other external security services such as key exchange or certification services.

But the environment in which management operations take place has dramatically changed since SNMP was devised. Looking at today's network technologies and the actual usage patterns of SNMP, it is obvious that devices could perform more complex management operations at low cost. It is reasonable to expect that devices, especially high-end routers and switches, will become increasingly programmable, and that it will become possible to execute more control software directly on the devices.

Furthermore, as described by Wellens and Auerbach [7], SNMP need not use UDP. When network connectivity is lost, non-SNMP mechanisms are usually used to bring back connectivity before management operations can resume.

Finally, SNMP was standardized at a time

From the network operators' standpoint, configuration management is the most important problem to be addressed to date.

when it was customary for the IETF to invent an ad hoc protocol each time a new problem had to be solved. As we will see, this resulted in a lack of skilled developers for writing increasingly complex management applications. Since the late 1980s the environment has changed, and the goal is now to reuse standard technologies wherever possible.

MARKET ASPECTS

Independent of any technical considerations, several market aspects must also be considered when evaluating proposals for future Internet management technologies [6].

The first aspect is branding. SNMP has a bad reputation among network administrators and Internet network operators. Even though it is still widely used for monitoring network devices, many people associate SNMP with the terms *insecure*, *cryptic*, *complex*, *slow*, and *limited functionality*; whether these terms are technically justified or not is irrelevant to them.

A second aspect is market control and protection. It is normal for companies to try to secure lucrative niche markets. This is true of the management technologies market as well. Some companies contribute to standards mostly to keep control over key technologies and their associated markets. Others prefer to use proprietary management interfaces in order to lock in their customers.

The third aspect is that there does not seem to be a sustainable market for open management. When SNMP was created, one of the original ideas was that open network management standards would create a competitive market, which would lead to improved management applications. After more than a decade of field experience with SNMP, we must acknowledge that this model does not seem to work very well. We have a reasonable market for low-end, reasonably open management applications whose main components are MIB browsers and data collectors. But systems management and application management are still to a large extent proprietary, and the number of management applications that are able to manage complex networks rather than individual devices is quite limited.

The fourth aspect has to do with purchase decisions. Management capabilities usually have little influence on decisions to buy a device: the main criteria are technological features and price. Very few businesses are able to compute the total cost of ownership (TCO) of a piece of equipment. As a result, most people prefer to save money during the purchase phase, even if they have to spend much more during the operational phase due to missing, incompatible, or nonstandard management interfaces.

The fifth aspect has to do with developers' skills. Because management often comes second to technological features, the employees assigned by equipment vendors to develop management instrumentation are often inexperienced. After a few years, they quickly move to more attractive functions. It is therefore difficult for a vendor to grow in-house expertise in Internet management.

The last aspect has to do with reuse. Over the years, the software engineering market has learned that implementation and education costs can be reduced significantly by using domain-independent technologies rather than domain-specific ones. Management applications are no exception to this rule. The number of highly experienced SNMP developers is rather small, notably because students and employees are more interested in learning generic protocols than SNMP. One outcome of this lack of skills is that many of the tools available to date for implementing management applications are rather primitive. To overcome this issue, general-purpose technologies should be adopted for management wherever possible. They reduce both development and training costs, and increase the chances of having smart people develop smart management applications.

STANDARDIZATION ASPECTS

In addition to technical and market issues, one should also take standardization issues into consideration when evaluating new Internet management technologies. The first observation is that standardization efforts at the IETF often take too much time. The short cycles of the late 1980s have been replaced by long (and sometimes hard) negotiations between vendors. For management interfaces, it is crucial to have a good timing. If reasonable specifications are available early enough, there is a good chance that vendors will adopt and implement them. Conversely, standards that are published after vendors have implemented and fielded their own proprietary solutions are unlikely to be adopted — there is generally no business case for supporting multiple interfaces.

A second aspect is data modeling. Management data models need continued maintenance as the underlying technology evolves. Although SMI has clear rules on how to evolve definitions while retaining interoperability with deployed implementations, we see little interest in the IETF Working Groups (WGs) to update existing MIB module definitions. Network device vendors are also relatively slow in implementing updated definitions in real products, because the marketing impact of announcing support for a new version of an MIB is very small.

Some of the IETF rules to progress standards also impact the clarity of data models. In some cases, related definitions are spread across several MIB modules just to be able to pass existing modules unchanged along the standards process. A good example is the `IF-INVERTED-STACK-MIB`, which contains a table that semantically belongs in the `IF-MIB`. The main outcome of splitting modules to cope with the IETF standard process rules is confusion. It is easy for people outside the IETF to miss a fragment of related definitions.

Last, design by committee rarely works well. The understandable attempt to accommodate every preference or resource constraint usually leads to data models that are difficult to understand, relatively vague in some areas, and not easy to use for developing robust and interoperable management applications.

Evolution of the SNMP Framework

One approach to solve these problems and meet the above mentioned requirements is to gradually improve the existing Internet management framework.

Evolution of the Data Definition Language

The IRTF Network Management Research Group (NMRG) has developed a next-generation data definition language called SMIng [8]. SMIng improves the expressive power of the current SMI language by introducing arbitrarily nested data structures. It facilitates reusability of complex structured types in order to achieve more commonality in MIB module designs. It also provides a language extensibility mechanism to allow for incremental language enhancements.

The SMIng development was driven by the goal to stop the proliferation of different data modeling languages within the IETF and to harmonize management models at a higher conceptual level. SMIng therefore separates management protocol-independent data type definitions from the protocol-specific mappings. The example in the upper half of Fig. 2 shows how a reusable data structure for physical or logical network interfaces can be defined in the SMIng language. The class `BasicInOutErrStats` defines a collection of counters for basic traffic statistics. (We ignore here the fact that 32 bits might not be sufficient for high-speed traffic streams). The class `Interface` has an index attribute and a statistics attribute holding basic traffic statistics for an interface. The bottom half of Fig. 2 shows an SNMP protocol mapping. The columns of the `ifTable` are explicitly listed in the object statements. Several SNMP objects are needed to represent the `stats` attribute, which has a composite type. In general, attributes that are defined using other SMIng classes are "flattened out" in the SNMP protocol mapping, following the structure of the underlying composite type.

In November 2000 the SMIng effort was moved from the IRTF to the IETF. An IETF Working Group (WG) called SMIng was chartered to develop a standards track specification for the next-generation data definition language, starting from the NMRG proposal. The SMIng WG, in the first phase, documented the objectives for a new data definition language [9]. In the second phase, proposals to meet the objectives were requested, and after some discussion two strong proposals remained in the list of candidates. One of them was the SMIng language developed by the NMRG. An attempt to merge the two competing proposals into what would have become SMIv3 failed, primarily because no consensus could be reached on the syntax of the SMIv3 language and, more important, how much the SNMP naming system should be changed to identify nested data types on the wire. The SMIng WG was finally shut down in April 2003 without producing a standards track specification.

```
class BasicInOutErrStats {
   attribute inOctets  { type Counter32;  ... };
   attribute inErrors  { type Counter32;  ... };
   attribute outOctets { type Counter32;  ... };
   attribute outErrors { type Counter32;  ... };
   ...
};
class Interface {
   attribute index { type InterfaceIndex; ... };
     attribute stats { type BasicInOutErrStats;
       ... };
   ...
};
snmp {
     table ifTable {
       oid        interfaces.2;
       index      (ifIndex);
       object     ifIndex     { implements
                  Interface.index; ... };
       object ifInOctets  { implements
                  Interface.stats.inOctets;  ... };
       object ifInErrors  { implements
                  Interface.stats.inErrors;  ... };
     object ifOutOctets { implements
                  Interface.stats.outOctets; ... };
     object ifOutErrors { implements
                  Interface.stats.outErrors; ... };
     ...
   };
   ...
};
```

Figure 2. *SMIng definition of an interface and the mapping to SNMP.*

The IRTF Network Management Research Group has developed a new data definition language called SMIng.

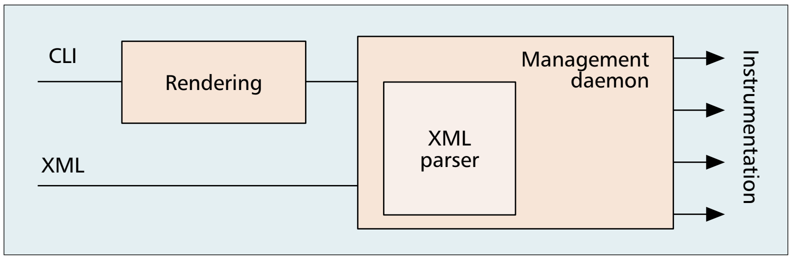
Evolution of the SNMP Protocol Operations

The IETF Evolution Of SNMP (EOS) WG was chartered in February 2001 to work on new SNMP protocol operations that basically improve the efficiency of bulk data retrievals [10]. Several options have been investigated.

The first option uses compression mechanisms that reduce the noticeable overhead caused by redundant object identifier (OID) fragments in SNMP protocol data units (PDUs). The variants of this option differ in how much computation is needed and how much compression is achieved. One of them, OID delta compression [8], is a relatively lightweight algorithm that achieves good compression ratios on tables with simple and complex indexing schemes. The advantage of the compression proposals is that compression can be applied to existing SNMP PDUs, which requires minimal changes to existing protocol operations.

The second option leverages suppression mechanisms where common OID fragments are suppressed. This works by introducing new protocol operations that operate on complex data structures rather than primitive lists of named variables. The various proposals for achieving this differ in the data selection and filtering mechanisms supported.

The third option is based on new MIB modules that facilitate bulk data transfers at the MIB level rather than the protocol level. These approaches have lost support because there are serious implications with regard to security and



■ **Figure 3.** *The structure of the agent's implementation.*

access control — especially if other protocols such as FTP or HTTP are used for the actual bulk data transfers.

The first two options generally benefit from larger PDU sizes. The default SNMP over UDP transport only guarantees the transport of SNMP messages of a size up to 484 bytes, which is too small for bulk data transfers over most layer 2 technologies. The NMRG therefore defined a transport mapping for SNMP over TCP to allow for larger SNMP messages with a minimum guaranteed message size of 8192 bytes [11].

As in the SMIng case, the EOS WG could not reach consensus on the proposals and was closed down in April 2003.

COPS-PR AND SPPI

The IETF Resource Allocation Protocol (RAP) WG defined the Common Open Policy Services Protocol — Policy Provisioning (COPS-PR) protocol [12] and its associated data definition language, the Structure of Policy Provisioning Information (SPPI) [13]. COPS-PR was designed to provision complex and continuously changing device configurations generated from policy-based management systems.

The design of COPS-PR addresses well-known issues in SNMP. In particular, COPS-PR uses TCP as its transport, which makes it possible to support large message sizes and use transport-layer security mechanisms. COPS-PR also assumes that only one entity can have exclusive control for a given subject category on a device. This assumption makes it easier to share state between managers and agents, and thus reduces complexity.

The SPPI language is a variant of SMI adapted to COPS-PR. It does not support SMIng features such as complex nested data types. In fact, the release of SPPI was one of the motivations to develop a protocol-neutral data definition language within the NMRG.

So far, COPS-PR and SPPI have failed to gain significant market acceptance. One reason is that Internet network operators are concerned about the increased complexity and maintenance costs associated with yet another management technology, which only partially fulfills their requirements. Another reason is that these protocols only provide minor improvements over SNMP, which could easily be integrated into the SNMP framework.

XML-BASED APPROACHES

XML-based management has been around for several years now. One of the pioneers in this area is the Distributed Management Task Force (DMTF), who developed the Web-Based Enterprise Management (WBEM) architecture and its main building block, the Common Information Model (CIM). CIM schemas define management information for users, applications, networks, systems, events, policies, and so on. They are defined in a language called Managed Object Format (MOF) and can be viewed in the form of UML class diagrams. To transfer management information over the wire, WBEM uses XML encoding.

Despite the fact that a number of vendors have been working on XML-based management for several years, the traditional Internet management community (IAB and IETF) has ignored it for a long time. Activities in this area were limited to those of the NMRG, where a mapping from SNMP MIB modules into XML Document Type Definitions (DTDs) was defined [8] and an XML-based management application prototyped [6].

The situation has changed, however. At the June 2002 IAB workshop, one of the interesting conclusions was the unanimous support to investigate XML-based network management. At the 54th IETF meeting in Yokohama, Japan, there was also a well-attended Birds of a Feather (BOF) meeting to discuss the use of XML in configuration management. As a result of this interest, a new WG called Network Configuration (NetConf) was formed in May 2003.

Independent of standardization, some vendors already support XML-based management in their routers.

JUNOSCRIPT

In January 2001, Juniper introduced its JUNOScript application programming interface (API) for the JUNOS network operating system [14]. This API gives management applications full access to the agent's management data using a lightweight remote procedure call (RPC) mechanism encoded in XML. As opposed to SNMP, JUNOScript uses a connection-oriented transport mechanism (e.g., `ssh` or `telnet`). A major advantage of this approach is that related management interactions can be grouped into sessions, which makes locking and recovery relatively simple. In addition, management information is no longer limited in size and can be exchanged reliably.

Figure 3 shows the internal management structure of a Juniper router. It is interesting to note that the Command Line Interface (CLI) and XML interface rely on the same pieces of software. The main difference between the two is that an additional rendering component is needed for the CLI to translate between the human-readable CLI interface and the more verbose XML-based messages. It is possible to switch rendering off with a single command, which may be useful for debugging purposes. Although there are many XML parsers on the market today, Juniper decided to implement their own parser for performance reasons. This parser understands only a subset of XML.

Interactions between a manager and an agent are based on XML messages, which can be regarded as RPCs. Although several XML-based RPC standards already exist (e.g., XML-RPC), Juniper considered these standards too complex and decided to use their own encoding. The Juniper encoding is indeed quite simple. Figure 4 shows an example of an RPC call performed by a manager to get statistics about an interface of a device. The call is embedded within rpc start and end tags. After the start tag come one or more methods; in this example there is just a single `get-interface-information` method. Each method may have a number of arguments; in this example there is only one: `statistics`.

The reply to the RPC call looks quite similar and includes the requested statistics (`InOctets`, `InErrors`, `OutOctets`, and `OutErrors`). The manager can analyze this response by using XPath. This makes it possible to find specific information (e.g., to identify the interfaces for which the number of errors exceeds a certain threshold). The response can be translated by using Extensible Stylesheet Language Transformations (XSLT), or formatted using Cascading Style Sheets (CSS).

```
<rpc>
   <get-interface-information>
      <statistics/>
   </get-interface-information>
</rpc>

<rpc-reply>
   <interface-information>
      <InOctets>123456</InOctets>
      <InErrors>789</InErrors>
      <OutOctets>654321</OutOctets>
      <OutErrors>0</OutErrors>
   </interface-information>
</rpc-reply>
```

Figure 4. *An example of an XML-encoded RPC call.*

WEB SERVICES FOR MANAGEMENT

Outside the traditional Internet management community, a number of technologies are being developed that may become important for Internet management. Web services are perhaps the most interesting of all.

This technology is being standardized by the World Wide Web Consortium (W3C). It promises to provide a single uniform software infrastructure to support a wide range of distributed services, thereby reducing training and software development costs. According to some, the real power of Web services is that they are expected to become standard components of future operating systems. As such, Web services may become easy to use and integrated within common office applications such as spreadsheets and databases.

Although Web services have not been specifically designed for management purposes, people may find them attractive to develop management applications. Spreadsheets, for example, may be used to retrieve usage figures from the network and inform the administrator if certain usage figures exceed certain thresholds. Databases may be used to periodically retrieve usage statistics and plot them over time. Web services could also facilitate the integration of network, application, and systems management by offering a unified communication model [6].

Despite all the recent marketing announcements, it should be noted that Web services are still very much under development. This technology is not yet mature, and its applicability in the area of Internet management is still an object of research.

Web services consist of several building blocks built on top of XML (Fig. 5). The first one is the Simple Object Access Protocol (SOAP). SOAP is fundamentally a stateless one-way message exchange mechanism that uses XML for encoding. SOAP messages can be exchanged over different underlying transfer protocols (e.g., HTTP and SMTP). Most people currently envision using SOAP over HTTP/1.1.

The second standard is the Web Services Description Language (WSDL, pronounced wisdle), which is used to define the actual Web services. WSDL files include the operations supported by a particular service, the parameters of these operations, the type of the returned value, the protocol binding (usually SOAP), and the location of the service (expressed in the form of a Uniform Resource Identifier, URI).

In the example given in Fig. 6, two messages are defined: `Statistics` and `StatisticsResult`. The first message does not contain any parameters. As in the example presented in Fig. 4, the second message contains four integer parameters: `InOctets`, `InErrors`, `OutOctets`, and `OutErrors`. Additionally, the WSDL file includes a section that assigns a name (`InterfaceInfoService`) to this service, the mapping onto the underlying protocol, and the URI of the service. Note that several lines have been omitted in Fig. 6 for the sake of readability. Just like SOAP, WSDL uses XML for encoding.

Note that a companion standard, Universal Description, Discovery, and Integration (UDDI), is often considered a building block of Web services, although it is defined by a vendor consortium called OASIS and not formally endorsed by the W3C. UDDI is supposed to be useful for discovering WSDL files.

As mentioned already, Internet management is traditionally based on the manager-agent paradigm. In this approach, the manager sends `Get` and `Set` commands to the agent, and receives responses from the agent. With Web services we have three different possibilities.

If we do a straightforward mapping, the agent runs an HTTP server, the manager runs an HTTP client, and the manager performs `Get`

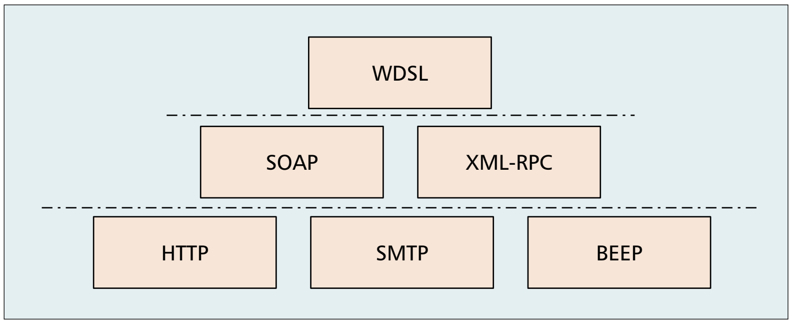


Figure 5. *Web services: a layered view.*

Although Web services have not been specifically designed for management purposes, people may find them attractive to develop management applications.

```
<definitions name="InterfaceInformation"
   ...

  <message name="Statistics">
  </message>

  <message name="StatisticsResult">
    <part name="InOctets"
        element="xsd:unsignedInt"/>
    <part name="InErrors" ...
    <part name="OutOctets" ...
    <part name="OutErrors" ...
  </message>

  <service name="InterfaceInfoService">
    <port ...
        {mapping on underlying protocol}
        {URI of web service}
    </port>
  </service>

</definitions>
```

Figure 6. *An example of a WSDL file.*

and `Set` RPCs via SOAP. Polling is implemented by invoking `Get` on the same attribute on a regular basis.

Web services also support publish-subscribe. In this case, a manager may express its interest in certain attributes published by an agent in a WSDL registry. The agent then sends the data to a message queue, and the manager receives these events from that message queue.

Last, Web services enable managers to invoke advanced operations on agents. These operations may perform statistical computations, update a routing table, perform load balancing, and so on.

In any case, the capabilities of the agent can be described in a WSDL file, which enables a manager to discover them at runtime (e.g., by retrieving them from a central UDDI registry implemented as a relational database). For performance reasons, it may be necessary for the manager to locally cache the WSDL files of all the agents in its management domain (which poses the problem of when to update them).

In addition to this typical request-response interaction pattern, agents should also be able to send notifications to the manager. With a straightforward mapping, the manager should thus also implement an HTTP server, and the agent an HTTP client. With publish-subscribe, notifications are sent via the same message queue mechanism as subscribed data.

As opposed to SNMP, which operates over UDP, Web services can operate over TCP, which means the maximum size of requests, responses, and notifications is no longer an issue.

As mentioned earlier, an important problem for network operators is configuration management of multiple devices. This form of management requires support for atomic transactions. A relatively new technology that may be useful for this purpose is the Business Process Execution Language for Web Services (BPEL4WS [15]), which seems likely to supersede previous proposals from IBM (Web Services Flow Language) and Microsoft (XLANG).

Although many general-purpose Web services technologies are already available, it is important to standardize specific Web services for network management. These standards could take the form of XML schemas or WSDL files. An important decision to be made is the level at which these standards should operate. Two extreme approaches are possible:

- The WSDL files define just a set of basic operations, such as `Get` and `Set`. In this approach, parameters are passed as opaque types, which means the WSDL file does not specify or interpret the types of the various MIB object values. It is possible, however, to define these types in a higher-level XML schema.
- The WSDL files define separate messages for each MIB object. Examples of such messages are `GetIfInOctets` and `ChangeIfOperationalStatus`. A parameter that belongs to both messages could be `ifIndex`. In this approach, WSDL files specify all the details that are necessary to manage a device. There is no need to define a higher-level XML schema. In fact, these WSDL files include the same kind of details as current MIB modules.

Since the second approach requires no additional XML schema, it would be relatively easy to use the resulting Web services from standard applications, like spreadsheets and databases. A risk, however, is that performance may be adversely affected in case such applications rely for their type checking on generic parsers that support all XML features.

The NMRG and the OASIS Management Protocol Technical Committee have just begun investigating the use of Web services for Internet management. The Parlay Group has recently translated its open service provisioning APIs into WSDL definitions. Unfortunately, these definitions are still difficult to use, since they require detailed knowledge of intelligent networks. Parlay-X took another approach and defined easy-to-use Web services for open service provisioning. Examples of such services include "connect A to B," "give status of X (on/off)," "send SMS," and "recharge prepaid card." Parallel to standards groups and vendor consortia, many research projects are also investigating the use of Web services for Internet management (e.g., the Dutch Freeband WASP project). Further research is needed, for example, in the areas of object naming and performance.

Conclusions

SNMP has been around for almost 15 years now. Although it is widely used for monitoring network devices, it has not been very successful for performing other important management functions such as configuration management. To discuss the situation, the IETF, IRTF and IAB organized various meetings in which they proposed future directions for Internet management. This article presents the main outcome of these meetings and gave an overview of the approaches that were investigated. These approaches fall into two categories: evolutionary and revolutionary. Evolutionary approaches were taken by three IETF Working Groups: SMIng, EOS, and RAP. The goal of the SMIng WG was to produce an improved version of the SMI data definition language; the goal of the EOS WG was to produce new SNMP protocol operations for efficient bulk data retrievals. Both groups leveraged previous work by the IRTF-NMRG. In addition, the RAP WG defined SPPI and COPS-PR, which enable the provisioning of network devices with policy-based configuration data.

So far, the evolutionary approaches have failed or had limited market acceptance. The IETF accepted this failure recently. In early 2003, it relaxed its requirement that new MIB modules should also contain writable objects. The IETF still encourages the development of read-only MIB modules, however.

Also, various participants of the June 2002 IAB workshop expected failure of the evolutionary approaches; it is therefore not surprising that an important outcome of the workshop was to focus more on revolutionary approaches. Currently, most activities in Internet management center around XML-based approaches. Several vendors already ship products that offer easy-to-use XML-based interfaces for configuration management; to standardize such interfaces, a new IETF WG (NetConf) was recently created.

Web services also seem to be a promising technology. Research in this area has just begun; further work is needed to investigate its merits in Internet management.

Currently, most activities in Internet management center around XML-based approaches. Several vendors already ship products that offer easy-to-use XML-based interfaces for configuration management; to standardize such interfaces, a new IETF Working Group (NetConf) was recently created.

Acknowledgments

Many ideas presented in this article were discussed at the June 2002 IAB workshop on network management and at various meetings of the IRTF Network Management Research Group. We would like to thank the participants of these meetings for their contributions, especially Phil Shafer and Bert Wijnen.

Part of this research was sponsored by the Telematica Instituut (TI) in the Netherlands.

References

[1] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411, Dec. 2002.

[2] K. McCloghrie *et al.*, "Structure of Management Information Version 2 (SMIv2)," RFC 2578, Apr. 1999.

[3] J. Schönwälder, "Overview of the 2002 IAB Network Management Workshop," RFC 3535, May 2003.

[4] L. Sanchez, K. McCloghrie, and J. Saperia, "Requirements for Configuration Management of IP-Based Networks," RFC 3139, June 2001

[5] J. Case *et al.*, "A Simple Network Management Protocol (SNMP)," RFC 1157, May 1990.

[6] J.P. Martin-Flatin, *Web-Based Management of IP Networks and Systems*, Wiley, 2002.

[7] C. Wellens and K. Auerbach, "Towards Useful Management," *The Simple Times*, vol. 4, no. 3, July 1996.

[8] Internet drafts produced by the IRTF NMRG: http://www.ibr.cs.tu-bs.de/projects/nmrg/.

[9] C. Elliot *et al.*, "SMIng Objectives," RFC 3216, Dec. 2001.

[10] R. Sprenkels and J. P. Martin-Flatin, "Bulk Transfer of MIB Data," *The Simple Times*, vol. 7 no. 1, Mar. 1999.

[11] J. Schönwälder, "Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping," RFC 3430, Dec. 2002

[12] K. Chan *et al.*, "COPS Usage for Policy Provisioning (COPS-PR)," RFC 3084, Mar. 2001.

[13] K. McCloghrie *et al.*, "Structure of Policy Provisioning Information (SPPI)," RFC 3159, Aug. 2001.

[14] P. Shafer, "XML-Based Network Management, White paper, Juniper Networks, Aug. 2001. Available at http://www.juniper.net/techcenter/techpapers/200017.html.

[15] F. Curbera *et al.*, "Business Process Execution Language for Web Services," v. 1.0, July 2002; http://www.ibm.com/developerworks/library/ws-bpel/.

Biographies

JÜRGEN SCHÖNWÄLDER [M] (j.schoenwaelder@iu-bremen.de) is associate professor of computer science at International University Bremen, Germany. He received his diploma in computer science in 1990 and his doctoral degree in 1996 from Technical University Braunschweig, Germany. His research interests are network management, distributed systems, and network security. He is an active member of the IETF and chair of the NMRG of the IRTF.

AIKO PRAS (pras@cs.utwente.nl) is a senior researcher at the Telematics Architecture Group of the University of Twente (UT), the Netherlands. From this university, he received in 1995 a Ph.D. degree in network management architectures. His research interests include Web services, network measurements, and accounting. He participates within the WASP and M2C research projects, and is a member of the IRTF NMRG.

JEAN-PHILIPPE MARTIN-FLATIN [SM] (jp.martin-flatin@ieee.org) is technical manager of the European DataTAG project (http://www.datatag.org) at CERN. He coordinates research activities in networking and middleware for data-intensive transoceanic Grids. Prior to that he wrote a book on Web-based management and was a principal MTS with AT&T Laboratories Research, Florham Park, New Jersey. He holds a Ph.D. degree in computer science from EPFL, Switzerland. He is a co-chair of the GGF Data Transport Research Group and a member of the IRTF NMRG.