Evaluating the Performance of SNMP and Web Services Notifications

Weldson Queiroz de Lima, Rodrigo Sanger Alves, Ricardo Lemos Vianna, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco, Lisandro Zambenedetti Granville
Institute of Informatics
Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brazil
Email: {wlima, sanger, vianna, janilce, liane, granville}@inf.ufrgs.br

***Abstract*— Web Services against SNMP comparisons have been carried out by the network management community in order to understand the impact on adopting Web Services as a management tool. In these comparisons, however, notification messages have been neglected to a secondary plane in such a way that the current conclusions about the performance of Web Services may not apply for networks extensively managed via notifications. In this paper we first evaluate the performance of Web Services notifications encoded following the WS-Notification specification, and compare it with the performance of SNMP traps. Then, we introduce a configurable SNMP to Web Services gateway that reacts to SNMP traps on behalf of an SNMP manager, retrieves further information from the notifying entity, and builds up WS-Notification compliant messages which are sent to Web Services-based managers. We finally evaluate the performance of the proposed solution in order to redraw the current conclusions about Web Services against SNMP, now explicitly considering the also important notification support.**

I. Introduction

In the last three years, the network management community has been actively investigating the use of Web Services for management. One of the main concerns has been related to the Web Services performance, mainly compared with the performance of the *de facto* standard TCP/IP management solution, i.e., the Simple Network Management Protocol (SNMP) [1]. Today, it is well known that SOAP (Simple Object Access Protocol), the Web Services basis protocol, can consume less bandwidth than SNMP when retrieving a large number of objects from a managed device [2]. Besides, the response time of a Web Services call can be reduced near to SNMP response time depending on the strategy adopted to map SNMP operations to Web Services operations [3]. Broadly, the main conclusion is that if one decides to use Web Services as a management tool, Web Services performance is not a main drawback considering the SNMP performance.

Such conclusion, however, is based on the observation of SNMP and SOAP messages for request-response interactions only: up to now, Web Services versus SNMP comparisons have neglected to a secondary plane the also important notification messages. Notifications are used to report to interested managers special events detected inside managed devices. In SNMP, notification support is materialized by trap messages, which are generated within managed devices and asynchronously sent to management stations. Although the current Web Services-related developing standards do include specifications for notification support (e.g., WS-Event [4] and WS-Notification [5]), there is no proper investigation contrasting Web Services notifications with SNMP traps. Thus, in networks whose management is extensively based on notifications, the current conclusions about the performance of Web Services against SNMP may not apply.

Since we do not believe that SNMP is going to be replaced by Web Services in a short term (in fact, we believe that SNMP and Web Services are probably going to co-exist in integrated environments), in our investigation we consider the necessity of an SNMP to Web Services gateway, whose goal is to integrate SNMP-enabled devices into Web Services-based management applications. This integration, specifically concerning notifications, is achieved translating SNMP traps to SOAP messages encoded according to the WS-Notification [5] specification. Although the translating gateway enables the SNMP and Web Services integration, it also introduces an additional processing overhead that may prevent Web Services-based managers to properly react to time sensitive events reported by just arrived notifications. In addition, Web Services notifications consume more bandwidth than SNMP traps because SOAP employs text-encoded messages, while SNMP employs shorter binary-encoded traps.

Although at a first sight Web Services notifications seem to perform poorer than SNMP, it has to be considered that SNMP traps do not exist alone: they usually trigger, in the notified manager, a sequence of SNMP requests to query the notifying device about complementary information related to the event being reported. This additional information helps the manager to better understand the reasons associated to the event and thus improve the quality of the actions taken to react to the notification. We take advantage over these additional interactions to improve the performance of Web Services notifications: we propose an enhanced gateway that reacts to notifying SNMP devices on behalf of an SNMP manager to retrieve complementary information, and builds up richer SOAP notifications sent to Web Services managers. This minimizes Web Services manager requests by moving them to the gateway, which decreases the bandwidth consumption and delivery delay.

1-4244-0143-7/06/$20.00 ©2006 IEEE.

The remainder of this paper is organized as follows. In Section 2 we review the related work on Web Services for network management. Section 3 elaborates a direct mapping of SNMP traps to WS-Notification messages, while Section 4 checks how different the performance of both SNMP traps and WS-Notification messages is. Section 5 presents our proposed SNMP to Web Services notification-aware gateway and evaluates such gateway considering the final traffic generated in an event report and the associated notification delivery delay. Finally, we provide the final remarks of this paper and present future work in Section 6.

II. Related Work

The research on Web Services for management was born following previous investigations on the use of the eXtensible Markup Language (XML) in network management. Both Web Services and XML-based management research has addressed, in some extend, the issue of integrating these technologies with the *de facto* standard SNMP framework. Usually, such investigations propose the adoption of intermediate gateways to allow the communication between SNMP-enabled devices and Web Services/XML-based managers.

In two previous works, we ourselves have investigated the use of SNMP/Web Services gateways operating at three different levels. *Protocol-level* gateways [6] directly map SNMP operations (e.g., Get, GetNext, Set) to corresponding Web Service operations, while *object-level* gateways translate SNMP MIB (Management Information Base) structures to Web Services operations (e.g., GetIfTable). *Service-level* gateways [3], in turn, group sets of MIB variables in service-related operations. In the two last cases (object and service-level gateways) the Web Services traffic can consume less bandwidth than SNMP because the manager calls a single Web Service operation that is translated to several SNMP requests. After receiving all replies, the gateway generates a single response sent back to the caller manager.

In July 2004, Pavlou et al. [7] have not only compared Web Services with SNMP, but also with OSI systems management and CORBA. Considering notifications, the authors show that Web Services notifications are not so well-defined if compared with CORBA, but Web Services notification specifications were quite new by that time. An integrated SNMP and Web Services notification support was not proposed in the Pavlou et al. work, although the authors were confident that new Web Services-based notification system might present notification services as sophisticate as those available in CORBA.

Klie and Strauss [8] have proposed an SNMP/XML gateway where SNMP traps were stored in a trap buffer internal to the gateway. In this solution, XML-based managers interested in receiving notifications must actively poll the gateway trap buffer to check for available notifications. This approach, however, increases the bandwidth consumption because it introduces the necessity of a polling process that generates request-response messages even if there are no traps to be processed. The authors have also proposed an alternative solution where the gateway works as an HTTP client that uses HTTP POST messages [9] to send XML-encoded traps to interested managers. In this case, managers must have the ability to receive HTTP requests by implementing HTTP servers.

Oh et al. [10] have also proposed an XML/SNMP gateway. In their work, an XML-enabled manager could request management information from target SNMP devices and filter the associated results within the gateway. Since XML-based management uses text-encoded messages larger than SNMP messages, Oh et al. approach reduces the XML traffic between a manager and the gateway through the definition of filters that cut down management information not of interest. To forward SNMP traps, the authors have proposed a solution similar to the Klie and Strauss's approach: gateways first receive SNMP traps, apply some filters to cut undesired data, and create a final XML document sent to managers via HTTP. Here again, gateways need to implement HTTP clients, while managers need to implement HTTP servers.

All the above researches have concentrated in observing request-response interactions: they have barely touched the notification support, and no final conclusion has been drawn considering the Web Services against SNMP notifications performance. However, since notifications are a powerful management tool, they should be studied when Web Services begin to be considered as an alternative to SNMP.

III. Directly mapping of SNMP to Web Services notifications

Web Services can be seen as a suite of technologies that enable the integration of systems based on the Web. The use of Web Services allows to group together information retrieved from very diverse sources in order to compute new information from the original set. For example, sophisticate services that can book flights and hotels in an integrated fashion is possible thanks to Web Services. In this case, at least two different bases (flights and hotels) of the same type (booking) are consulted.

Network management information is critical to the operation of businesses. Such information, however, also needs to be manipulated considering other information from different areas of the same business (e.g., backup systems, voice over IP infrastructure, and databases). SNMP is a technology originally focused on network management. Although some may have success on using SNMP on different domains, it does not have the same potential of Web Services as a systems integration solution. On the other side, Web Services, at least in a short term, are unlikely to replace SNMP as core network management technology. We thus believe that both SNMP and Web Services are required but unable to replace each other in their original domains.

In this context, we believe that current network management systems will be integrated with other management systems forming a single yet distributed Web Services-enabled management system. At this point, from the perspective of managed devices, the today SNMP manager would be replaced by a Web Services manager. In this scenario, the clear necessity of

an SNMP to Web Services translation arises. As seen before, intermediate gateways have been investigated as a solution to implement such required translation. However, an SNMP to Web Services gateway may employ different strategies to allow the communication between the SNMP-enabled device and the Web Services-enabled manager. One such a strategy, addressed in this section, is the direct mapping of SNMP operations to Web Services operation, or more specifically to the investigation of this paper, SNMP traps to Web Services notifications.

In this section we first analytically study the traffic of SNMP traps, to then compare it with the traffic of corresponding Web Services notifications generated by an intermediate gateway that directly maps SNMP traps to Web Services notifications.

A. SNMP traps traffic

SNMP traffic has been previously explored by Pras et al. [2], but SNMP traps were not addressed. Inspired by that work, in this subsection we present a complementary study that computes the maximum traffic of SNMP traps. Although SNMPv3 is a full standard, we concentrate our analysis on SNMPv1 and SNMPv2c for the sake of simplicity. Additional information on SNMPv3 traps performance is provided in the subsequent sections.

SNMP messages are a set of ASN.1 [11] elements encoded following the Basic Encoding Rules (BER) [12]. Each ASN.1 element has three parts: type, length, and value. *Type* is encoded in one octet, while *length* requires a varying number of octets depending on the length of the *value* part. For example, if *value* is encoded by less than 128 octets, then *length* is encoded in 1 octet; if *value* is encoded with 128 up to 255 octets, then *length* is encoded in 2 octets; if *value* requires 256 up to 65535 octets, *length* will be encoded by 3 octets.

SNMP has two different trap PDUs, one defined by SNMPv1 [13] and another by SNMPv2c[1] [14], which are composed of three main parts: an authentication header, a PDU header, and a variable binding (varbind) list. Each trap itself is an ASN.1 element whose *length* part is encoded by 1 to 3 octets (for messages smaller than 65535 octets). Added to one octet require to encode the *type* part, the maximum length of an SNMP trap message is expressed as:

$$L_{\text{trap}} = 4 + L_{\text{authentication}} + L_{\text{header}} + L_{\text{varbindlist}} \quad (1)$$

The authentication header is composed of two fields: the protocol *version* (3 octets), and a *community string* ($2 + L_{\text{community}}$ octets, if $L_{\text{community}} < 128$). Thus, the authentication header length of an SNMP trap is expressed as:

$$L_{\text{authentication}} = 5 + L_{\text{community}} \quad (2)$$

SNMPv1 trap PDU header has five fields: *enterprise* ($1 + L_{\text{enterprise}}$ octets), *agent address* (6 octets), *generic-trap* (3 octets), *specific-trap* (3 octets to encode a trap value up to 127), and the trap *timestamp* (7 octets max). SNMPv2c trap PDU header has three fields: *request_id* (6 octets), and two unused zeroed integer fields (3 octets each). Again, each SNMP PDU header is an ASN.1 element itself, whose *length* part is also encoded by 1 to 3 octets (for messages smaller than 65535 octets). Considering the additional octets for the *type* part, the maximum length of an SNMP PDU header is:

$$\begin{aligned} L_{\text{headerSNMPv1}} &= 24 + L_{\text{enterprise}} \\ L_{\text{headerSNMPv2c}} &= 16 \end{aligned} \quad (3)$$

A single variable binding, or simply varbind, is composed of an *object identifier* (OID) and an associated *object value*. To encode an OID, 2 plus the OID length (L_{OID}) minus 1 octets are required. That is so because the two integers (usually "1.3") present in the beginning of an OID is encoded in a single octet. The number of octets required to encode the varbind value (L_{value}) depends on the value type. For example, integers are usually encoded in 3 octets. Finally, a varbind needs two additional octets to be properly encoded. Thus, the total octets required to encode a varbind is:

$$L_{\text{varbind}} = 3 + L_{\text{OID}} + L_{\text{value}} \quad (4)$$

To encode a varbind list, further initial octets are needed, according to the length of the list in bytes. For example, if the length of the varbind list is less than 128 bytes, then 2 octets are used. From 128 to 255 bytes, 3 octets are used, and from 256 up to 65535 bytes, 4 octets. SNMPv2c traps have two fixed initial varbinds: the trap *timestamp* and the trap *type*. The *timestamp* varbind consumes up to 19 octets, while the trap *type* consumes up to 15 + L_{trapOID} octets. If one needs to notify n varbinds ($n < 65535$), all of them with the same L_{OID} and L_{value}, the total maximum number of octets is:

$$\begin{aligned} L_{\text{varbindlistSNMPv1}} &= 4 + n \cdot L_{\text{varbind}} \\ L_{\text{varbindlistSNMPv2c}} &= 38 + L_{\text{trapOID}} + n \cdot L_{\text{varbind}} \end{aligned} \quad (5)$$

Considering 1, 2, 3, 4, and 5 we have that SNMP trap messages will be no longer than:

$$\begin{aligned} L_{\text{trapSNMPv1}} &= 37 + L_{\text{community}} + L_{\text{enterprise}} + n \cdot (3 + L_{\text{OID}} + L_{\text{value}}) \\ L_{\text{trapSNMPv2c}} &= 63 + L_{\text{community}} + L_{\text{trapOID}} + n \cdot (3 + L_{\text{OID}} + L_{\text{value}}) \end{aligned} \quad (6)$$

Now, let's consider an SNMP `linkDown` trap (as defined in the Interface MIB module [15]), which is sent by a managed device if one of its links goes down. The identification of the unavailable link (`ifIndex`) and associated status (`ifAdminStatus` and `ifOperStatus`) are sent to the manager in three varbinds, each with OID encoded in 11 octets ($L_{\text{OID}} = 11$), and value sent in an integer ($L_{\text{value}} = 3$). In addition, let's consider the following usual values for the other SNMP trap fields:

- Community string: *public* $\rightarrow L_{\text{community}} = 6$
- Enterprise: *1.3.6.1.4.1* $\rightarrow L_{\text{enterprise}} = 6$
- Trap OID: *1.3.6.1.6.3.1.1.5.3* $\rightarrow L_{\text{trapOID}} = 10$

[1] Although SNMPv2c also defines the InformRequest PDU to notify remote entities, we do not address such message because InformRequet is less frequent in management scenarios than traps

If we suppose that a single trap reports m unavailable links (i.e., $n = 3 \cdot m$), the L_{trap} would be up to:

$$\begin{aligned} L_{\text{linkdownSNMPv1}} &= 49 + 51 \cdot m \\ L_{\text{linkdownSNMPv2c}} &= 79 + 51 \cdot m \end{aligned} \quad (7)$$

B. WS-Notification messages

WS-Notification [5] is a specification that defines a complete architecture to support complex notification environments based on Web Services. In some senses, WS-Notification is an evolution of the WS-Event [4]. We do not aim here at providing a discussion about the pros and cons of each solution, or compare them two. That is available in several white papers on the Internet. However, in general, the WS-Notification standardization efforts seem to be more active, and provide enough definitions to be used in this investigation.

In such architecture, a *publisher* is the entity that reports notifications about some *topics* to *consumers*. *Brokers* are used between publishers and consumers to control which notifications should be forwarded to which consumers based on the notifications topics. Interested consumers must register in a broker to state which topics they are interested in, while publishers must also register to state which topics they are supporting. Playing with WS-Notification elements, quite sophisticate notification setups can be defined.

WS-Notification also defines the Web Services messages required by the architecture components (publishers, consumers, brokers, etc.) to communicate with each other (e.g., messages to register publishers and consumers within brokers). In this paper we do not address such auxiliary Web Services message, but will focus on the notification message `Notify` only, in order to compare its performance with SNMP. The `Notify` message is issued by the producer in a *Document Style* SOAP communication. *Document Style* allows asynchronous SOAP communications proper for notifications, which contrasts with the popular synchronous SOAP Remote Procedure Call (RPC). Figure 1 shows the structure of the `Notify` message.

```
... SOAP header ...
<wsnt:Notify>
   <wsnt:NotificationMessage>
      <wsnt:Topic Dialect="xsd:anyURI">
         {any} *
      </wsnt:Topic>?
      <wsnt:ProducerReference>
         wsa:EndpointReference
      </wsnt:ProducerReference> ?
      <wsnt:Message>
         {any}
      </wsnt:Message>
   <wsnt:NotificationMessage> +
</wsnt:Notify>
```

Fig. 1. The format of a `Notify` message [5]

Each `Notify` is a collection of `NotificationMessages`. Each `NotificationMessage`, on its turn, contains the identification of the message topic, and a reference to the producer end-point. The `Message` element contains the actual notification message payload. Other optional elements may also be found in a notification, for example, metadata about a subscription. We will not deal with these other elements because in direct mappings of SNMP to WS-Notification such elements are not required.

Notification messages are encapsulated by SOAP in order to be delivered. SOAP itself is further layered on top of HTTP, SMTP, or BEEP, although SOAP over HTTP is a far more common implementation. Due to that, in our evaluations to be presented in the next sections, we concentrate in observing WS-Notification messages running over the SOAP/HTTP binding.

Although SOAP and WS-Notification are clearly specified, achieving a final formula to express the length of a WS-Notification compliant message is quite difficult because the number of octets to encode several elements may vary a lot. In addition, there is no specification that defines how to encode the notification message payload, leaving this decision to the WS-Notification user. This prevents a reasonable analytical study of Web Services messages. In order to compare WS-Notification with SNMP, we then take the approach of implementing Web Services and check their performance related to bandwidth consumption and delivery delay. Thus, in our evaluation, we study the WS-Notification performance by implementing gateway prototypes that map SNMP traps to WS-Notification messages. The first two gateway, to be presented next, directly maps SNMP traps to WS-Notification messages; the third gateway provides more sophisticated mapping, as it is going to be presented in Section 5.

C. Strategies in mapping SNMP to WS-Notification

We assume in this work that SNMP to Web Services gateways are used in management environments similar to the one presented in Figure 2.

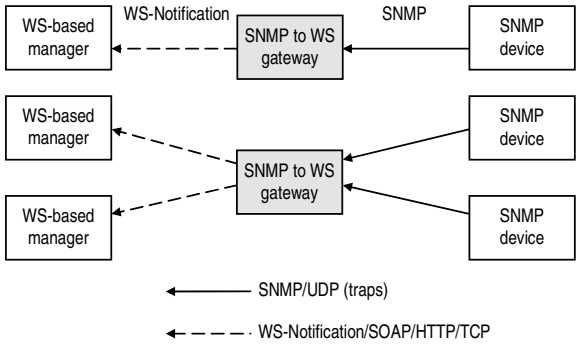

Fig. 2. Management environment with SNMP to WS gateways for notification mapping

In such environment, SNMP agents are configured to send trap messages to specific SNMP to Web Services gateways. Each gateway then maps just received traps to WS-Notification messages and forwards them to interested managers registered in each gateway. Managers registration is manually handled configuring gateway registries, but more dynamic registration

using WS-Notification subscriptions could be employed (although it is out of the scope of this paper).

Considering an initial direct mapping of SNMP traps to Web Services notifications, we first evaluate the performance of WS-Notification running on top of SOAP over HTTP using two mapping strategies: SNMP to XML-encoded payload, and SNMP traps tunneled in the notification payload.

In the *SNMP to XML-encoded payload* mapping strategy, the contents of the fields of an SNMP trap are input data to generate both WS-Notification message fields and XML-encoded payload (i.e., the `Message` element contents). Table I presents the mapping between SNMP fields to WS-Notification message fields and payload, using this first strategy.

TABLE I
SNMP TO XML-ENCODED PAYLOAD MAPPING STRATEGY

SNMP field (version)	WS-Notification element
SNMP version (v1, v2c)	(unmapped)
Community string (v1, v2c)	Payload in `Message`
PDU type (v1, v2c)	(unmapped)
Request ID (v2c)	(unmapped)
Agent address (v1)	`ProducerReference`
Timestamp (v1)	Payload in `Message`
1st varbind - timestamp (v2c)	Payload in `Message`
Enterprise (v1)	`Topic`
Generic trap type (v1)	`Topic`
Specific trap type (v1)	`Topic`
2st varbind - trap type (v2c)	`Topic`
Remaining varbinds (v1, v2c)	Payload in `Message`

SNMP version and PDU type do not need to be mapped because both SNMP trap PDUs and protocol versions are transformed to the same WS-Notification structure, in a way that a Web Services manager does not need to deal with different notification versions. Request ID from SNMPv2c is also not mapped because notifications are supposed to be asynchronous messages without a reply associated. Even if it were the case, Request ID is not needed in the gateway/manager communication because TCP is used as the connection-oriented transport protocol.

The WS-Notification `topic` element is used to inform the trap being transmitted. For SNMPv1, enterprise, generic trap, and specific trap are mapped to a single topic; for SNMPv2c, `topic` is defined according to the second varbind, which carries the trap type [16]. The agent address is mapped in the `ProducerReference` element. For SNMPv2c, which does not include an agent address, this information is taken from the IP header of the network datagram. Here we assume that there is no SNMP proxy between the network device and the intermediate gateway, which is a requirement to preserve the correct mapping of the agent address. Finally, all other SNMP fields (community, timestamp, and remaining varbinds) are included in the `Message` element.

Figure 3 shows the final WS-Notification message produced from mapping an original SNMP `linkDown` trap.

```
... SOAP header ...
<wsnt:Notify>
   <wsnt:NotificationMessage>
      <wsnt:Topic
      Dialect="http://www.inf.ufrgs.br/traps">
         npex:linkDown
      </wsnt:Topic>
      <wsnt:ProducerReference>
         <wsa:Address>
            snmp://143.54.47.253
         </wsa:Address>
      </wsnt:ProducerReference>
      <wsnt:Message>
         <timestamp xsi:type="xsd:long">
            1284967225</timestamp>
         <ifIndex xsi:type="xsd:unsignedInt">
            3</ifIndex>
         <ifAdminStatus xsi:type="xsd:unsignedInt">
            2</ifAdminStatus>
         <ifOperStatus xsi:type="xsd:unsignedInt">
            2</ifOperStatus>
      </wsnt:Message>
   </wsnt:NotificationMessage>
</wsnt:Notify>
```

Fig. 3. A `linkDown` trap mapped to a WS-Notification message

In the case of the *SNMP traps tunneled in the notification payload* strategy, the gateway only pushes the SNMP message binary contents into the `Message` element without further processing it except by using the base 64 encoding algorithm [17]. All remaining WS-Notification fields turn to be unneeded. Web Service manager, in this second direct mapping strategy, must have the ability to deal with SNMP trap messages on its side, since the WS-Notification support is merely used as a tunneling mechanism between the notifying agent and the notified manager.

Figure 4 presents the final WS-Notification message used to tunnel SNMP traps.

```
... SOAP header ...
<wsnt:Notify>
   <wsnt:NotificationMessage>
      <wsnt:Topic
      Dialect="http://www.inf.ufrgs.br/traps">
         npex:tunneled
      </wsnt:Topic>
      <wsnt:ProducerReference>
         <wsa:Address>
            snmp://143.54.47.253
         </wsa:Address>
      </wsnt:ProducerReference>
      <wsnt:Message>
         <SNMPtrap xsi:type="xsd:base64Binary">
            (trap encoded by the base64 algorithm)
         </SNMPtrap>
      </wsnt:Message>
   </wsnt:NotificationMessage>
</wsnt:Notify>
```

Fig. 4. SNMP trap tunneled in a WS-Notification message

IV. EVALUATION

In order to evaluate SNMP and Web Services notifications, we have set a testing network composed of four Linux PCs running the Slackware distribution. The first one has the role

of an SNMP-enabled device that, through the NET-SNMP [18] `snmptrap` tool, simulates link problems generating `linkDown` traps. The SNMP to WS gateway, running in the second PC, receives SNMP traps via the NET-SNMP `snmptrapd` daemon, and calls the gateway software to handle each received notification. The gateway and manager have been coded using the gSOAP library for C++ [19]. All these three PCs (SNMP device, gateway, and manager) are all connected in an isolated 100 Mbps switch, with no background traffic. A fourth PC, connected in a switch port that mirrors all network traffic, runs the `tcpdump` tool [20] to monitor the traffic of SNMP and Web Services notifications. Table II summarizes the hardware and software resources used in the testing environment.

TABLE II
HARDWARE AND SOFTWARE FOR THE EVALUATION ENVIRONMENT

	SNMP device	gateway	manager	monitor
Processor	Pentium 4 (2.4Ghz)	Pentium 4 (2.4Ghz)	Pentium 4 (2.4Ghz)	Pentium 4 (2.4Ghz)
Memory	256MB	256MB	256MB	256MB
OS	Slackware 10.1(2.4.28)	Slackware 10.1(2.4.28)	Slackware 10.1(2.4.28)	Slackware 10.1(2.4.28)
SNMP support	NET-SNMP (v 5.2.1.2)	NET-SNMP (v 5.2.1.2)	–	–
WS support	–	gSOAP (v 2.7.5)	gSOAP (v 2.7.5)	–

A. Network usage

Our first evaluation checks the network usage when the notifying traps have an increasing number of objects. To achieve that, we simulate a `linkDown` trap that would be able to report in the same message more than one unavailable network interfaces. Although actual `linkDown` traps notify only one single interface per message, we simulate here the behavior of carrying more interfaces for the purpose of our evaluation. Each interface being reported in a `linkDown` is addressed by three varbinds (`ifIndex`, `ifAdminStatus`, and `ifOperStatus`).

In the evaluation, we consider not only the notification messages themselves (SNMP and WS-Notification), but also the overhead of the whole protocol stack, thus including transport layer, network layer, and data link layer protocols. We observe both regular and (`zlib`) compressed Web Services messages. In addition, we also check how the security support of SNMPv3 and HTTPS affects the network usage in the end-to-end communication. The use or not of security is treated in the following way. When security is not used, the SNMP device sends its traps using SNMPv2c and the gateway forward such traps using HTTP as the protocol to carry the WS-Notification messages. When security is present, the SNMP device sends traps using SNMPv3 and the gateway uses HTTPS to contact the manager. Figure 5 presents the network usage for the gateway that uses XML to encode the original SNMP trap fields in the WS-Notification message payload.

Although not perceptible in Figure 5, in our experiments SNMPv3 traps consumed few more bytes than SNMPv2c, around 70 bytes. As expected, uncompressed WS-Notification (WSN) messages are greater than SNMP (v2c and v3). More important, the greater the number of object, the greater the difference between SNMP and WS-Notification traffic. The use of HTTPS to provide a secure communication between gateway and manager only increases such difference, since additional security information is required in the communication. When we compress (via `zlib`) the WS-Notification messages, however, the angle of the traffic line using conventional HTTP transport falls significantly at a level that it crosses the SNMP line when notifying around 175 objects. When HTTPS is used, the network consumption increases, but the traffic line angle is the same as the HTTP line, crossing the SNMP line around 375 objects. This confirms that compressed SOAP can perform better than SNMP. However, in the case of notifications, this is true for a large number of objects (greater than 175 for HTTP and greater than 365 for HTTPS), which is quite rare in actual management environments.

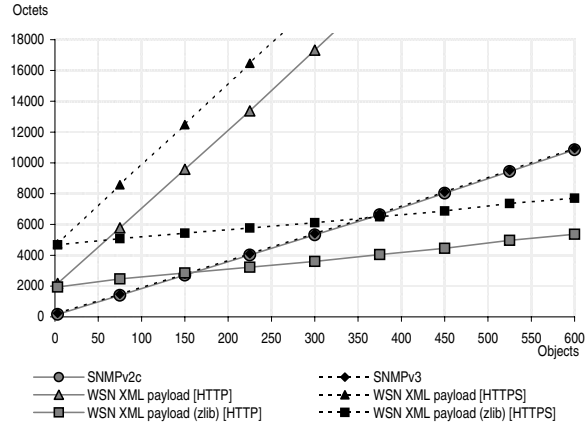

Fig. 5. SNMP and Web Services notification network usage - WS-Notification messages with XML-encoded payload

In the case of tunneling SNMP traps in WS-Notifications (Figure 6), HTTP consumes more bandwidth than SNMP, but the difference is relatively constant. Using HTTPS, again, more bytes are needed, but the traffic consumption grows like in the HTTP case. The use of compression produces a traffic line that crosses the SNMP line when 145 objects are carried using HTTP. The compressing of SNMPv3 traps is less effective, however, as can be seen in the case where `zlib` tries to reduce the size of SNMPv3 messages before using HTTPS to deliver to the manager the just received notification. In this case the traffic consumed when employing compression over encrypted messages are more similar to the traffic consumption presented by uncompressed messages than to the traffic consumed by compressed but not encrypted messages. This least to the conclusion that, for the tunneling approach, the use of compressed, not encrypted messages is more interesting, and the only able to perform better than SNMP if more than 150 objects are transmitted. Even in

this case, however, SNMP still remains as the lighter solution because is fact the number of object carried in traps is often far less than 150.

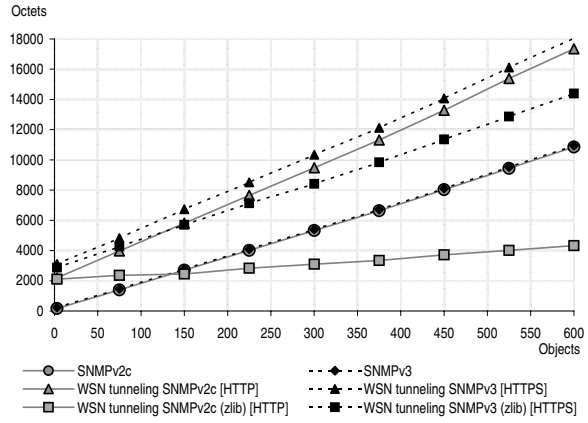

Fig. 6. SNMP and Web Services notification network usage - WS-Notification messages tunneling SNMP traps

B. Delivery delay

Our second evaluation considers the notification delivery delay. In this case, SNMP traps delay is taken as the reference point. We measure the delay of each Web Services notification timestamping, in the monitoring PC, the last packet of the original SNMP trap (if more than one IP packet carries the SNMP trap) and the last packet seen in the TCP connection used to deliver the corresponding WS-Notification message generated by the intermediate gateway. Here, we are interested in checking how much additional delay is introduced by the intermediate gateway. Figure 7 presents the measured delays for the various Web Services notification messages. For each number of notified interfaces, we have repeated the measurement 30 times. The results presented in Figure 7 have been computed with a confidence interval of 95%.

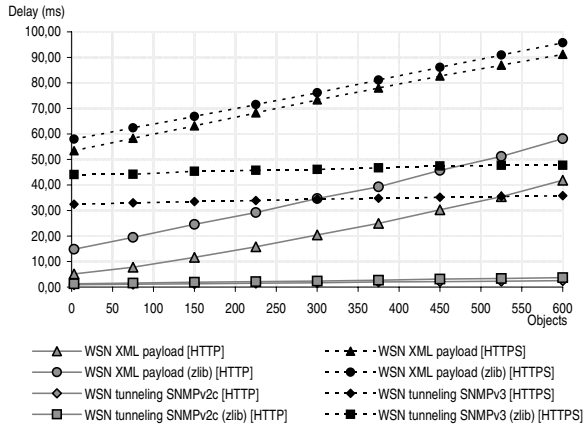

Fig. 7. Web Services notification delay

As can be observed, the use of HTTPS is responsible for increasing the delay for all WS-Notification messages, at least carrying less than 300 objects. As can also be observed, the use of (`zlib`) also increases the delivery delay for each scenario. In the case of the gateway that uses XML documents in the WS-Notification payload, the delivery delay grows according to the number of notified objects. With the introduction of (`zlib`) compression, such delay is slightly greater (around 9 ms). In the case of the tunneling approach, the delivery delay is almost constant, regardless the number of notified objects. With the introduction of compression, there is no significant increase in the delivery delay, only being greater than 3 ms when notifying 450 objects.

The tunneling option has significant less overhead than the messages with payload encoded in XML because no transformation from SNMP to XML is executed in the gateway. This also affects how the overhead increases when carrying more objects. In the tunneling strategy, with additional objects the overall overhead is almost constant while in the approach using XML the overhead increases much more significantly.

It is important to notice that at the manager side the decoding effort required to handle the receiving notification is different according to the mapping strategy used in the gateway. In the case of XML in the notification message, the manager only needs to manipulate the received XML documents because both the notification itself and its contents are encoded in XML. In the case of the tunneling strategy, part of the processing is spent decoding the Web Services notification and part is spent decoding the encapsulated SNMP trap. In this second case, the manager not only needs to handle the Web Services notification but it also must be able to decode SNMP traps. In the implementation of a manager, the XML in the payload may be more convenient because it does not force the manager to deal with SNMP details. However, the increased delay shown but that strategy may require its substitution for the tunneling approach, in environments where a fast reaction is needed at the manager site.

These results show that, for the direct mapping of SNMP traps to Web Services notifications, the use of compressed WS-Notification messages over HTTP tunneling original SNMP traps is the solution that consumes less bandwidth without introducing prohibitive delay in the intermediate gateway. However, the tunneling approach with compressed messages begins to be better than SNMP in terms of network usage only when notifying 175 or more objects, which is in fact a rare situation because SNMP traps are likely to carry very few objects. Thus, in practical situations, since SNMP traps with few varbinds are still very lightweight, SNMP-based notification remains a solution that performs better than Web Services notifications.

V. ENHANCED SNMP TO WS GATEWAY

As presented before, the WS-Notification traffic and associated delivery time performs poorer than SNMP in real cases (although compressed messages perform better than SNMP in hypothetical situations with a larger number of objects). The previous results indicate that WS-Notification, compared with SNMP, can not provide proper notification-based management without consuming far more network resources. On the other

hand, totally discarding the use of Web Services for notifications may not be possible in some situations, for example and as discussed before, if the manager to be notified is a Web Services-enabled manager instead of an SNMP-enabled manager.

In this situation, a mixed solution seems to be a feasible alternative: SNMP should be used near to the notifying device, while WS-Notification should be used to reach the notified manager. The intermediate point, i.e., an SNMP to Web Services gateway, would be then responsible for integrating both sides. Although the direct mapping-based gateways presented in the previous sections helped to understand the impact of Web Services notifications, the associated results indicate that the use of such gateways would be, due to the observed performance, inadequate to provide the desired integration. Thus, alternative solutions are required.

In this section we present an enhanced SNMP to WS gateway that aggregates management information when reacting to SNMP traps in order to deliver richer Web Services notifications to managers.

A. Gateway operation

In traditional management, SNMP traps not rarely trigger in the notified manager a sequence of SNMP requests back to the notifying agent in order to collect additional information related to the reported event. Such additional information is used by the manager in deciding about the actions to take in response to the original SNMP trap (Figure 8).

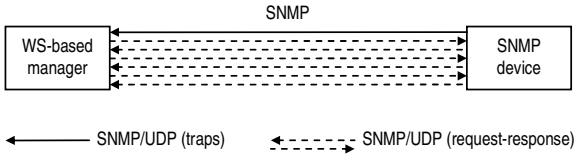

Fig. 8. Manager-agent SNMP direct interactions

In this scenario, the distance between manager and agent drastically impact on the delay between issuing a trap and having the manager finally reacting to it: the more distant the manager, the slower the reaction. That is so because SNMP forces quite interactive communications between manager and agent that can not be easily overlapped in time, i.e., to retrieve a new information the manager needs to receive the response of the previous request. This situation is particularly critical in the Internet, where the distance between the notifying device and the interested manager can be quite large when inter-domain management is required.

The goal of the proposed gateway is to move the intense SNMP interactions closer to the notifying device, and contact the distant manager only after having collected all relevant information related to the event being reported. In such a scenario the intermediate gateway works as a mid level manager (MLM) from the management by delegation (MbD) model [21]. In our case, however, we consider only a subset of the operations usually offered by a MLM: we assume that the MLM is focused in receiving traps, retrieving additional information from the network device, and finally notifying the remote manager via WS-Notifications (Figure 9).

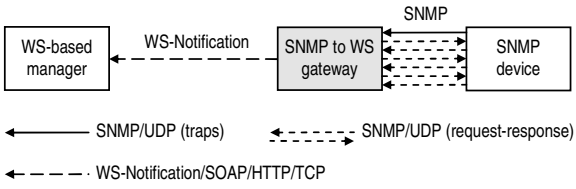

Fig. 9. Employment of an intermediate gateway

In order to determine which additional information is relevant to each notified event, the gateway consults an internal trap description database. This database is remotely feed by the Web Services manager at a setup phase by transferring to the gateway XML documents that describe the traps of interest. Another technology that could be used in this scenario is the IETF Script MIB [22]. In this case, a management script responsible for performing the previous actions would be transferred to the MLM (i.e., gateway) using the SNMP protocol. Additional parameters of execution of such script would be also transmitted via SNMP, such as the additional information required when receiving a specific trap. Although the Script MIB is a powerful tool for MbD, it is based on SNMP to support the management-MLM communication. Since we are assuming a situation where the manager-gateway communication is based on Web Services, and the expressiveness of management scripts is less important for this study than the performance of notifications, we keep our gateway simple dealing with also simple XML descriptive documents.

B. Gateway architecture

Figure 10 presents the architecture of the proposed gateway, highlighting the components related to the notification support.

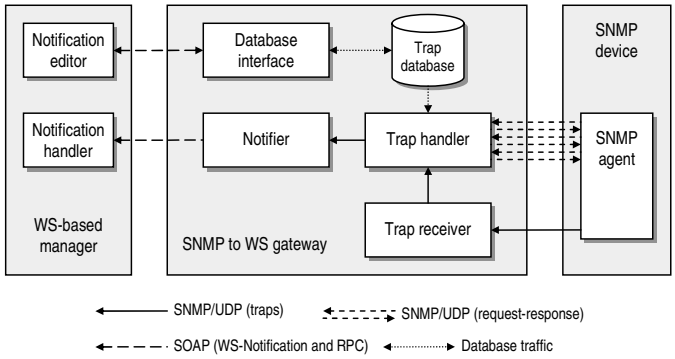

Fig. 10. Enhanced SNMP to Web Services gateway architecture

The *trap receiver* component is responsible for receiving SNMP traps issued by SNMP agents from managed devices. This component is implemented by the `snmptrapd` daemon from the NET-SNMP package [18], as cited before. Each received trap is internally forwarded to the *trap handler* component that, on its turn, queries the *trap database* to discover which additional information from the notifying device is

required before building up a Web Services notification. The trap database is maintained in memory, with a copy of its registries in the file system for the case of a gateway reboot. The trap handler has been also implemented in C++.

Once the trap handler discovers the additional information, it contacts the SNMP agent in the managed device to retrieve such additional information using (potentially) several SNMP request messages. To query the managed device, the trap handler uses the functions provided in the NET-SNMP API.

After receiving all additional information from the managed device, the trap handler uses the *notifier* component to build up the final Web Services notification to be sent to the end manager. The notifier is also a C++ module that implements a Web Services client using the gSOAP library [19]. All final notifications are encoded as WS-Notification documents whose payload carries an XML document. This is similar to the first approach presented in the first gateway evaluated before. Although the second approach (tunneling) presented better results, tunneling cannot be used in this third gateway because the original SNMP trap is not preserved when the additional information (retrieved from the managed device) is incorporated in the final WS-Notification message.

At the manager side, the *notification handler* component implements, also using gSOAP, a Web Services server that receives WS-Notifications. Also at the manager side, the notification editor is used by the network operator to specify how SNMP traps must be handled in the intermediate gateway. The operator writes *trap treatment descriptions* that are simple XML documents expressing the addition information associated to each trap of interest.

Figure 11 presents an example of a trap treatment description for the `linkDown` trap. In this case, the operator is explicitly stating in the description that he/she is interested in reading four columns (`ipCidrRouteDest`, `ipCidrRouteMask`, `ipCidrRouteIfIndex`, `ipCidrRouteNextHop`) of the routing table of the device that reports a `linkDown`. In the description document, the additional information may be defined providing either the object name (e.g., `ipCidrRouteDest`) or the object ID (e.g., 1.3.6.1.2.1.4.24.4.1.1) of the SNMP MIB.

```
<trapTreatmentDescription>
   <trap>linkDown</trap>
   <additionalObjects>
      <object>ipCidrRouteDest</object>
      <object>ipCidrRouteMask</object>
      <object>ipCidrRouteIfIndex</object>
      <object>ipCidrRouteNextHop</object>
   </additionalObjects>
</trapTreatmentDescription>
```

Fig. 11. Example of a *trap treatment description*

Each description edited by the operator can be deployed in the SNMP to WS gateway by calling, via SOAP Remote Procedure Call (RPC), the gateway operation (`deployTrapDescription`). Additional operations are available at the gateway to manage the set of trap descriptions: `listTrapDescriptions` and `removeTrapDescription`.

The final WS-Notification, as presented before, incorporates both the original SNMP trap fields and additional information retrieved from the managed device according to the trap description written and deployed in the intermediate gateway by the network operator. Figure 12 presents an excerpt of an WS-Notification message generated after processing a `linkDown` trap based on the trap description presented in Figure 11.

```
... SOAP header ...
<wsnt:Notify>
   <wsnt:NotificationMessage>
      <wsnt:Topic
      Dialect="http://www.inf.ufrgs.br/traps">
         npex:linkDown
      </wsnt:Topic>
      <wsnt:ProducerReference>
         <wsa:Address>
            snmp://143.54.47.253
         </wsa:Address>
      </wsnt:ProducerReference>
      <wsnt:Message>
         <timestamp xsi:type="xsd:long">
            1284968409</timestamp>
         <ifIndex xsi:type="xsd:unsignedInt">
            3</ifIndex>
         <ifAdminStatus xsi:type="xsd:unsignedInt">
            2</ifAdminStatus>
         <ifOperStatus xsi:type="xsd:unsignedInt">
            2</ifOperStatus>
         <ipCidrRouteDest xsi:type="xsd:string">
            143.54.47.1</ipCidrRouteDest>
         <ipCidrRouteMask xsi:type="xsd:string">
            255.255.255.128</ipCidrRouteMask>
         <ipCidrRouteIfIndex xsi:type="xsd:string">
            1</ipCidrRouteIfIndex>
         <ipCidrRouteNextHop xsi:type="xsd:string">
            0.0.0.0</ipCidrRouteNextHop>
         ... Other destination information ...
      </wsnt:Message>
   </wsnt:NotificationMessage>
</wsnt:Notify>
```

Fig. 12. Example of a WS-Notification message with both trap and additional information

C. Gateway evaluation

In order to perform the evaluation of this third proposed gateway, we have used the same testing scenario described for the first two gateways. We have measured the traffic at both sides of the gateway to compute the network usage between the notifying agent and gateways, and between the gateway and the notified manager.

In the previous evaluations we have increased the number of objects carried in an SNMP trap. For this new evaluation we consider the trap description of Figure 11 and vary the number of routing destinations in the managed device. We simulate a `linkDown` trap to trigger in the gateway a sequence of SNMP messages to download the routing table.

Figure 13 presents the network usage for SNMP (in the device-gateway communication) and Web Services (in the gateway-manager communication).

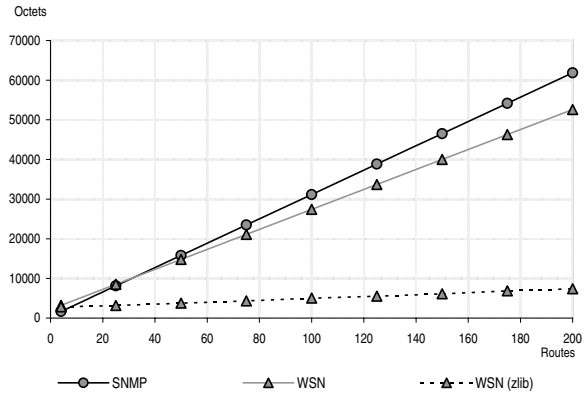

Fig. 13. Network usage of the enhanced SNMP to Web Services gateway

As can be seen in Figure 13, the compression process drastically decreases the bandwidth consumption mainly because more information is now available to compress.

For the evaluation of the notification delay, we simulate a long distant connection between the manager and the agent by delaying SNMP messages at the gateway. In this case, the gateway also works as an intermediate route that links manager and agent. Considering an introduced delay of 10 ms between manager and agent, Figure 14 presents the performance of both SNMP and Web Service when transferring to the manager all information required to handle the `linkDown` trap as described in Figure 11.

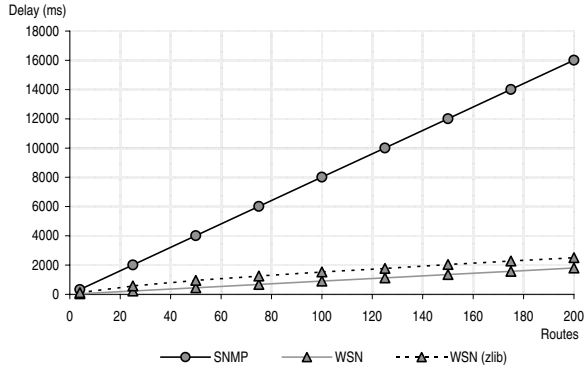

Fig. 14. Enhanced SNMP to Web Services gateway notification delivery delay

SNMP presents here worst delivery time because the final delay is the sum of the delays of each SNMP message between manager and agent. Web Services perform better because they do not accumulate the SNMP delay, which in this case is confined to the communication between agent and gateway.

After receiving the notification, the manager typically analyzes the notification contents and proceeds with a corrective action if required. In this reverse communication initiated at the manager side, different approaches for mapping Web Services to SNMP has been already investigations as cited before [6] [3] [7] [9] [10]. If further contacts with the managed device are needed to lead the managed device to consistent state, the manager must then use an intermediate gateway built using one of such approaches to translate the Web Services requests to SNMP operations forwarded to the managed device.

VI. Conclusions and Future Work

In this paper we have evaluated the performance of Web Services and SNMP-based notifications, considering network usage and delivery delay. We have compared SNMP traps with WS-Notification messages initially observing two gateways that, using different strategies, directly map SNMP to WS-Notification.

The first gateway transforms binary encoded trap into XML-encoded document inserted into the payload of WS-Notifications. This first approach performs poorer than SNMP, but when WS-Notification messages are compressed via `zlib`, the bandwidth consumption falls at a level that if the number of objects to be notified is greater than 175, then the WS-Notification traffic consumes less bandwidth than SNMP. With the introduction of security support via HTTPS and SNMPv3, however, more octets are required in the communications, then consuming more network bandwidth.

The second gateway works as a tunneling end-point, where SNMP traps are tunneled over WS-Notification messages via the base 64 algorithm. Since this approach does not use verbose XML documents as in the first gateway, the final bandwidth consumption is reduced. Compressed tunneled messages consume less bandwidth than SNMP if more than 145 objects are carried in the notification messages.

For the delay observation, we have taken SNMP message as a reference point and measured how the processing overhead introduced by the first two gateways has increased the final delivery delay. The first gateway that employs XML-encoded payload introduces a greater delay than the gateway used only for tunneling. One interesting aspect is that the introduction of compression affects more the first gateway than the second one. This is due to the large number of octets to be processed by the compressing algorithm in the first gateway. Another important observation is that the use of HTTPS is the fact most responsible for the increase in the delivery delay, with an impact more significative, for example, than the use of compression via `zlib`.

From the evaluation of the first two gateways, we can conclude that the performance of Web Services notifications, when using direct mapping, does not help to argue in favor of it. Although WS-Notification messages perform better than SNMP for a large number of objects, Web Services are not a feasible solution because this situation is quite rare, since SNMP traps usually carry few objects.

However, in some environments SNMP is not a possible notification tool at all. Here, Web Services could be a feasible option, despite its more resource consuming nature if compared with SNMP. For this scenario of integrated Web Services managers and SNMP devices we have introduced a third gateway that interacts with the notifying device to retrieve further information besides that already available from original traps. The goal of this gateway is to isolate intensive and bandwidth consuming SNMP messages near to the notifying

agent, while generating fewer WS-Notification messages to report events to Web Services managers.

The evaluation of the third gateway shows that confining SNMP traffic near to the notifying device in fact promotes a better bandwidth usage. In addition, it reduces the delivery time by placing the intermediate gateway closer to the SNMP agent. Our evaluation has been carried out considering that a network operator is interested in downloading the routing table of devices that report a `linkDown` problem. The gateway is flexible to deal with different traps and different additional information by defining trap treatment descriptions. For different descriptions, different performance results will probably appear. In this sense, the proposed gateway works as a specialized mid level manager when we consider the management by delegation approach.

Our main objective in this work was to cover the notification issue not addressed as a primarily component of SNMP-based management in the comparisons of Web Services against SNMP. We can conclude with this study that notifications are a more sensible question than regular request-response interactions addressed in previous work from the network management community. As a future work, we have been investigating the performance of Web Services in alarm correlation systems, since alarm correlation is a feasible approach to decrease the number of notifications and then also decreasing the network resources consumption.

REFERENCES

[1] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. Internet Engineering Task Force (IETF), RFC 3411, STD 62, December 2002.

[2] A. Pras, T. Drevers, R. v.d. Meent, and D. Quartel. Comparing the Performance of SNMP and Web Services-Based Management. *IEEE eTNSM - eTransactions on Network and Service Management*, 1(2):11, December 2004.

[3] T. Fioreze, L. Z. Granville, M. J. B. Almeida, and L. M. R. Tarouco. Comparing Web Services with SNMP in a Management by Delegation Environment. In *9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005)*, pages 601–614, Nice, France, May 2005.

[4] N. Catania et al. *Web Services Events (WS-Events)*. Hewlett-Packard Specification, 2.0 edition, July 2003. http://devresource.hp.com/drc/specifications/wsmf/WS-Events.pdf.

[5] G. Steve, D. Hull, and B. Murray. *Web Services Base Notification (WS-BaseNotification)*. IBM, Tibco, Hewlett-Packard Company, 1.3 edition, July 2005. (work in progress).

[6] R. Neisse, R. L. Vianna, L. Z. Granville, M. J. B. Almeida, and L. M. R. Tarouco. Implementation and Bandwidth Consumption Evaluation of SNMP to Web Sevices Gateways. In *9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*, pages 715–728, Seoul, South Korea, April 2004.

[7] G. Pavlou, P. Flegkas, S. Gouveris, and A. Liotta. On Management Technologies and the Potential of Web Services. *IEEE Communications Magazine*, 42(7):58–66, July 2004.

[8] F. Strauss and T. Klie. Towards XML Oriented Internet Management. In *8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003)*, pages 505–518, Colorado Springs, USA, March 2003.

[9] T. Klie and F. Strauss. Integrating SNMP agents with XML-based management systems. *IEEE Communications Magazine*, 42(7):76–83, July 2004.

[10] Y. J. Oh, H. T. Ju, M. J. Choi, and J. W. Hong. Interaction Translation Methods for XML/SNMP Gateway. In *13h IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2002)*, pages 54–65, Montreal, Canada, October 2002.

[11] International Organization for Standardization. *Information processing systems - Open Systems Interconnection, Specification of Abstract Syntax Notation One (ASN.1)*, December 1987. International Standard 8824.

[12] International Organization for Standardization. *Information processing systems - Open Systems Interconnection, Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)*, December 1987. International Standard 8825.

[13] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). Internet Engineering Task Force (IETF), RFC 1157, STD 15, May 1990.

[14] R. Presuhn. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). Internet Engineering Task Force (IETF), RFC 3416, STD 62, December 2002.

[15] K. McCloghrie and F. KastenHolz. The Interfaces Group MIB. Internet Engineering Task Force (IETF), RFC 2863, June 2000.

[16] R. Frye, D. Levi, S. Routhier, and B. Wijnen. Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework. Internet Engineering Task Force (IETF), RFC 3584, August 2003.

[17] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. Internet Engineering Task Force (IETF), RFC 2045, STD 62, November 1996.

[18] Net-SNMP. http://net-snmp.sourceforge.net/, 2005.

[19] Genivia Consulting. gSOAP Web Services toolkit. http://sourceforge.net/projects/gsoap2, 2005.

[20] tcpdump. http://www.tcpdump.com/, 2005.

[21] Y. Yemini, G. Goldszmidt, and S. Yemini. Network Management by Delegation. In *International Symposium on Integrated Network Management*, April 1991.

[22] J. Schnwlder, J. Quittek, and C. Kappler. Building Distributed Management Applications with the IETF Script MIB. *IEEE Journal on Selected Areas in Communications*, 18(5):702–714, May 2000.