A Notification Correlation Architecture Based on Policies and Web Services

Evandro Della Vecchia Pereira, Lisandro Zambenedetti Granville,
Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco

Federal University of Rio Grande do Sul (UFRGS) - Institute of Informatics
Av. Bento Gonçalves, 9500 - Bloco IV - Porto Alegre, RS - Brazil
{edvpereira, granville, janilce, liane}@inf.ufrgs.br

Abstract. Currently, Web Services for network management have been an intensive field of investigation. However, the investigations carried up to do not properly address the issue related to event notifications. The Simple Network Management Protocol, which is the widely deployed and accepted solution, has an interesting but also limited support for event notification through its trap messages. In this paper we present a distributed notification correlation architecture that allowed us to closely investigate the use of Web Services as a tool in managing networks considering the specific case of notification support. Other aspect explored in this paper is the use of policies in a notification correlation architecture. This study complements the investigations under development showing aspects of Web Services and Policy Based Network Management (PBNM) for network management that were unknown in the field of event notification.

1 Introduction

Currently, there is an intensive research effort in the network management community concerning the use of Web Services (WS) technology as a management tool. According to Schoenwaelder et al. [1], Web Services represent a revolutionary approach because its usage suggests the substitution of the current SNMP framework [2] that is widely accepted and deployed. However, although said to be revolutionary, the use of Web Services for network management can not be feasible in real systems without concerning the SNMP-enabled devices already in use.

In order to integrate SNMP-enabled devices into WS-based management systems, some integration approaches have been proposed. One common approach is based on the use of SNMP to WS gateways that map SNMP messages, information, and/or services to WS operations. Such gateways are often based on WS that are implemented through SOAP (Simple Object Access Protocol) [3] over HTTP. This SOAP/HTTP binding is currently easily deployed mainly due to the large number of available development and supporting tools.

SOAP over HTTP suggests a synchronous communication where a client, using a SOAP API, calls a WS operation that is executed in a server located in a remote computer. Usually, the client blocks waiting for the server reply. Observing the network traffic, one can check that a SOAP message is first sent from the client to the server to call the WS operation, and a final message is sent back from the server to the client to return

the operation results. These two messages easily remind the request/response messages of SNMP, then indicating that WS client-server interactions are quite similar in nature to the SNMP manager-agent interactions. In this way, SNMP to WS gateways can be built, through the use of the SOAP/HTTP binding, mapping the WS client request to an SNMP request (Get, GetNext, Set, etc.), and mapping back the SNMP response (GetResponse) to a WS server reply. The synchronous nature of the SOAP/HTTP binding, however, is not suitable for the SNMP notification messages (traps), and very few (if any) investigation on this issue has been carried out.

In this context, alternative SOAP bindings could be used to accomplish the necessity for SNMP notification support on the SNMP to WS gateways. This paper presents the investigation of the use of SOAP over SMTP to deliver SNMP notification messages to WS-based managers. To do that, we have defined a distributed notification architecture (which is able to correlate event notifications) composed by SNMP to WS notification gateways and policy-based notification forwarders. As the name suggests, the notification gateways map SNMP notifications to WS asynchronous calls, while forwarders correlate and forward notifications based on policies defined by a network administrator. We evaluate our proposed solution regarding the network bandwidth consumption, as well as discuss the pros and cons in having WS as a tool for network event notification. The contribution of this work is two fold: first because it observes a SOAP binding other than HTTP in the context of network management, and second because our architecture forwards notifications based on a policy-oriented approach. Although notifications based on policies is present in our work (as one will check in the architecture to be presented in next sections), this paper is primarily focused in evaluating the differences between SNMP and SOAP/SMTP binding for events notification.

The remainder of this paper is organized as follows. Section 2 presents related work, reviewing Web Services for network management and event correlation. Section 3 presents the event correlation architecture while section 4 presents a system prototype implementation. Section 5 shows results from the use of the proposed architecture in a test network, and section 6 finally closes this paper with conclusion and future work.

2 Related Work

In this section we review Web Services for network management and event correlation.

2.1 Web Services for Network Management

Developed and standardized by the World Wide Web Consortium (W3C), Web Services (WS) are independent Web-based application components that other applications can find and use. WS vary from simple to complex and promise flexibility and distributed computing on the Internet [4].

In the context of network management, G. Pavlou et al. [5] define that WS have strong analogies to CORBA. It can support request/response/error remote call interactions. Service interfaces are specified in the Web Services Description Language (WSDL) [6], which constitutes a general XML-based framework for the description of services (equivalent to CORBA IDL).

Other investigations have been carried out in order to observe performance-related aspects of WS for network management. Jeroen van Sloten et al. [7] investigate how SNMP to WS gateways can reduce the bandwidth consumption. Ourselves have evaluated SNMP to WS gateways when mapping original SNMP messages to WS operations [8] [9].

In general, it is possible to state that the current investigations observe the WS traffic and the associated performance when translating and integrating (using some approach) SNMP operations/information into a WS-enabled network management environment. If we divide SNMP in request/reply and asynchronous messages (traps), request/reply messages are far more used to managed deployed networks than traps. Then, it is not a surprise to realize that WS and SNMP investigations are related to SNMP request/reply messages and, as far as the authors of this paper are aware of, specific investigations related to notifications have not been carried out up to today. This paper aims to present an investigation on this issue, showing the collected results obtained from the observation of the SNMP and SOAP/SMTP traffic generated by our implemented distributed correlation architecture.

2.2 Event Correlation and SNMP Traps

A notification is the reporting of an event that can be related to situations considered abnormal. Network device event notifications are essential to help administrators to be aware about network related problems. But, in many cases, multiple events addressing the same problem can be received by a management station. It may happen, for example, because the interval defined for an event to be sent may be too short. This can overload the manager software visualization with several events that are related to the same problem. Another scenario that increases the amount of events is when a single fault triggers several different notifications. For example, if a router is down and a server reached through this router is up, server and router monitoring agents can send notifications indicating that both router and server are down, when actually only the router is not operating. The list below summarizes some common situations related to possible events overload [10]:

- An agent can send lots of notifications related to the same fault;
- An intermittent fault produces a new event each time the fault is detected;
- A faulty device can trigger an event each time the device is required (e.g. a query);
- A single fault can be detected by different agents;
- A fault in some device can affect other devices of the same network.

In this scenario, a reduction on the number of event messages, through event correlation, is needed to make the processes of analyzing and solving problems faster. Some techniques can be used for event correlation, such as rule based reasoning, models based reasoning, case based reasoning, codebook, state transition graph model, and finite state machine. More details about these techniques can be found in [11].

As pointed in the introduction, probably the most used notification mechanism is SNMP traps. This mechanism, however, has several widely known drawbacks, such as lack of acknowledgement and limited size of the information carried by the notification

messages. A complete substitution of SNMP traps is not possible, however, because it would involve the updating of the devices' IOS or even a full device replacement that is not often feasible in the already deployed networks.

Although Web Services are pointed as a revolutionary architecture [1] for network management (then could replace the SNMP traps too), we believe that a mixed approach, combining SNMP and Web Services, is imperative both to achieve the widely proclaimed advantages of using Web Services for network management and to still being able to manage SNMP-based devices. The next sections present our proposed solution and a prototype implementation to support event correlation based on Web Services without excluding the SNMP-enabled devices. We have used the SNMP-related elements of this architecture to evaluate WS as an event notification tool, whose results are presented in section 5.

3 The Web Services-based Notification Correlation Architecture

The goal of the proposed notification correlation architecture is to correlate notifications in a hierarchical way (figure 1). The notifications (SNMP traps) are generated in network devices and sent to an SNMP/WS gateway. In such gateway the traps are converted to SOAP/SMTP messages and sent to a forwarder. Each forwarder is actually a mid-level manager (MLM) where the notifications are correlated and possibly sent to another MLM. This process is repeated until notifications reach the top-level manager (TLM)

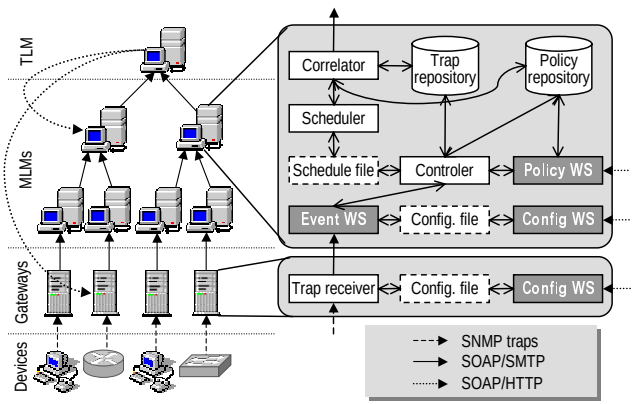

Fig. 1. Notification Correlation Tree and Gateway and MLM Structures

Besides receiving notifications on the top-level of the architecture, the TLM is the element responsible to configure both gateways and forwarders (MLMs) via configuration Web Services that are reached, in this case, via SOAP/HTTP messages. The HTTP binding has been used because configuration actions are typically accomplished by the

request/reply kind of interaction. The following subsections present the details of the architecture paying closer attention to the asynchronous interactions typically required for an event notification support.

3.1 SNMP Traps to SOAP/SMTP Translation

Due to some drawbacks, traps should be treated inside network administrative domains. For example, SNMP traps hardly cross border firewalls, but usually can reach other hosts on the local network. In the proposed architecture, once a trap is issued in the local network, it must be translated to SOAP messages over SMTP. Once translated, traps (which are now on the form of SOAP/SMTP messages) can be forwarded having associated acknowledged replies because SMTP uses TCP as the transport protocol. Also, SOAP/SMTP traffic may more easily cross border firewalls, which enables the elements of the proposed architecture to be spread along several different administrative domains.

In order to provide a trap to SOAP/SMTP translation, both network device and gateway need to be properly configured. On the network device, the SNMP agent must point to a corresponding gateway when issuing new traps. That is done changing the configuration options of the device's SNMP agent. On the gateway, a configuration file (remotely managed thanks to the **config WS** element) lists the devices from where the gateway is allowed to receive traps (via the **trap receiver** element), and the MLMs to where SOAP/SMTP messages must be forwarded.

3.2 Receiving and Correlating Notifications

Each MLM has an internal **event WS** element that is an SMTP server responsible for receiving the notifications on SOAP/SMTP messages. The event WS element - which has some operating parameters (to be addressed in the next section) specified in a local configuration file - extracts from the SMTP message the SOAP content and calls an operation on the **controller** element that will treat the just received notification. The controller then scans the local **policy repository** to check those policies that can handle the received notification. Every policy related to the notification is then evaluated to check whether its associated actions should be executed. If no policy related to the received event is found in the repository, then the notification is further delivered to another default MLM (optimistic approach) or it is discarded at all (pessimistic approach). The selection of the optimistic or pessimistic approach is based on the internal configuration of each MLM.

Often, a policy just evaluated needs to be evaluated in the future again, due to the temporal aspect of some event correlations described in a policy. In this way, temporal events need to be notified as well. For example, if a policy correlation rule states that if event X is followed by event Y in less than 30 seconds then the action Z must be executed, once the notification describing event X evaluates the policy, there must be a way to reevaluate the policy after 30 seconds if no notification describing event Y arrives, i.e. there should be a clock event notifying the 30 seconds timeout. This time-related events are accomplished in the proposed architecture through policy scheduling.

The **schedule file** is the one that stores schedule information, and its content is managed by the controller element.

Once a notification is received and it is associated to a policy, this notification is stored in the **trap repository**. This repository has a table of traps and associated policies under evaluation. This information is used when a policy reevaluation is triggered by the **scheduler** element (which monitors the schedule file). Policy reevaluation is performed by the **correlator** element that proceeds checking all notifications that have been received and are related to the policy under reevaluation. If the set of events makes the policy conditions evolve to true, then the policy actions are executed.

Several different actions might be executed. In this work only two actions are explored, that are the definition of the destination MLM and variables value to be addressed in the new correlation. That is accomplished when the correlator calls another external MLM via a new SOAP/SMTP message. After actions execution, all stored traps related to the evaluated policy are removed from the trap repository.

4 Prototype Implementation, Deployment, and Results

We have implemented a system prototype that follows our proposed distributed correlation architecture. The following subsections show the implementation of each element of the architecture in a bottom-up fashion.

4.1 Network Devices and SNMP Agents

We have used three different trap sources: a Windows based PC, a Linux based PC and a Cisco router. All of them had the corresponding SNMP agents activated and configured in order to send their SNMP traps to associated gateways. In this case, the associated gateways, from the SNMP agent point-of-view, are managers responsible for receiving and handling SNMP traps.

4.2 Gateways implementation

Gateways were implemented using the NET-SNMP toolkit [12], PHP [13] scripts and the PEAR::SOAP library, in a Linux environment. In order to receive the traps issued by the managed devices, the `snmptrapd` daemon from the NET-SNMP toolkit has been used. Once a trap is received, the daemon forwards it to a PHP script (`gateway.php`) that translates the received trap to a WS operation call using SOAP over SMTP.

Each received trap is logged into a temporary file just before the `gateway.php` script is executed. Then the gateway script reads the several trap parameters from the temporary file and builds up a SOAP message sent to a remote manager via SMTP, i.e. a SOAP/SMTP call. The following information, however, is required to be present in order to proceed with such call:

– **Destination e-mail**: since we are dealing with an SMTP transfer, the e-mail of the target entity, which can be either a MLM or the TLM, have to be determined;

– **Web Service operation**: since at the target entity different WS operations can be invoked, there is the need to determine which one is responsible to handle the notification call.

Destination e-mail and WS operation are both defined in the gateway configuration file. Although they can have very different value from gateway to gateway. In our default configuration file the WS operation is always defined as "TrapHandler" and the destination e-mail is configured as manager_name@manager_domain.

4.3 Mid-Level Manager (MLM) implementation

The MLM has also been implemented in a Linux platform. An SMTP server is responsible for receiving WS notifications via SOAP/SMTP, while an HTTP Apache server is responsible for receiving policies from the TLM. Inside the MLM both the policy repository and trap repository are implemented through a MySQL database.

Three main PHP scripts implement the Event WS, Config WS, and Policy WS elements. Config WS and Policy WS are reached via SOAP/HTTP, while Event WS (which exposes the TrapHandler WS operation) is the only one reached via SOAP/SMTP. The remaining elements (Controller, Scheduler, and Correlator) were also implemented in PHP scripts. The Scheduler is constantly called to evaluate the policies deployed. This is accomplished with an infinite loop that verifies if there is some correlation to be done. The interval time of each verification is configured by the TLM. The scheduler script is executed in background.

4.4 Top-Level Manager (TLM) implementation

The TLM has also been implemented with PHP scripts and for the interface it was used HTML. In the TLM, the operator creates policices and can send them to the MLMs using HTTP based WS. To create a policy, the operator fills the policy fields (figure 2) and save it in the Global Policy Repository. An unique identification number (id) is created for each policy and trough this id it is possible to choose witch policy will be sent to each MLM. Before the transferring, the policy is converted to XML and when received by the MLM, a parser reads the policy and save it in the Local Repository.

Besides the policy inclusion mentioned above, the implemented prototype also has policy query, exclusion, MLM sending, configuration of MLMs and a screen showing the correlated traps.

5 Evaluation

The proposed architecture presents two points of observation: policy-related issues and network management based on WS notifications. At this moment we are more concerned about the second point because, as discussed in Introduction, there is no study on the impact of using WS as a notification support. Thus, we concentrate our initial observation in checking the network usage associated to the SNMP traffic between the network devices and gateways, the corresponding SOAP/SMTP traffic between gateways and MLMs and traffic generated by the TLM to configure MLMs.

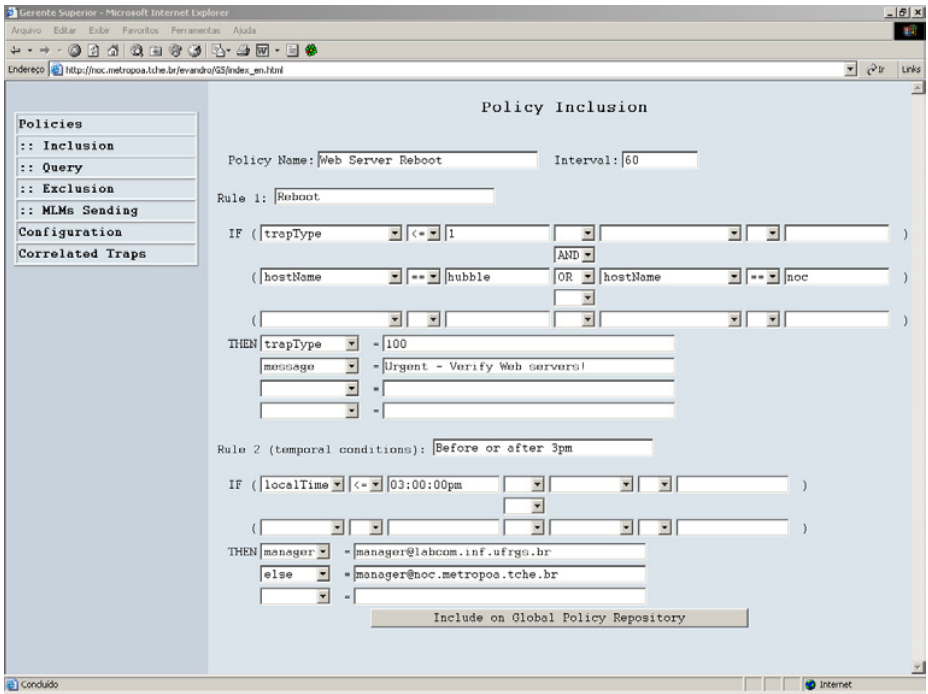

Fig. 2. Including policies in the Global Repository.

5.1 Test Environment and Network

To verify the network usage associated to SNMP and WS traffic we monitored the links of a test network (figure 3). In this test environment, traps were generated by a host named Hubble. Those traps were sent to a gateway named Labcom. After translating the SNMP traps, Labcom started notifying, via SOAP/SMTP, a MLM named Noc. Hubble was used also as the TLM, so policies and were sent by Hubble to Noc.

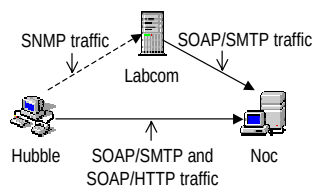

Fig. 3. Test network

5.2 Trap Control

At Hubble we have controlled the triggering of SNMP traps via a simple bash script. This script also includes as many integer SNMP variables in the traps messages as determined in an input parameter. The traps issued inform a link up event, and the list of variable describes the hypothetical interfaces of Hubble that went up again.

5.3 SNMP versus SOAP/SMTP traffic

SNMP traffic includes regular header and a list of additional variables. Initially we generated a trap with no additional variables, checked the size of the generated SNMP trap and the volume of WS traffic to transfer the corresponding SOAP/SMTP message that was generated by the gateway. Then we progressively increased the number of fake interfaces up to 90. Figure 4 shows the measured network usage by both SNMP and WS.

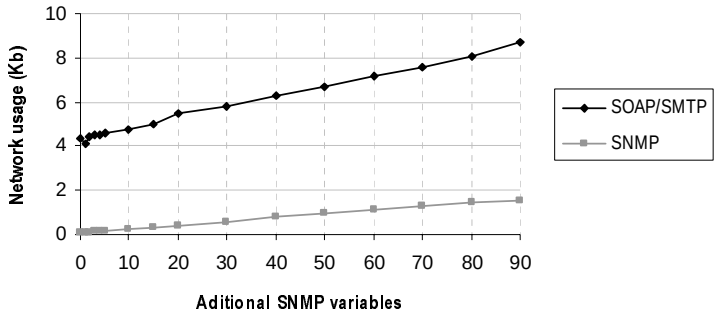

Fig. 4. SNMP versus SOAP/SMTP traffic

SNMP generates quite less traffic than SOAP/SMTP, as can be observed. For messages with fewer variables this is a quite straightforward. Increasing the number of variables makes network consumption increasing as well. The important point here, however, is the observation that not only the SOAP/SMTP traffic is greater than the SNMP one, but that the difference in the traffic increases when more variables are carried.

The observation is critical because other SNMP against WS investigations have shown that although WS traffic is greater than SNMP when few variables are present, WS traffic increases less than the SNMP traffic when more variables are present, leading to a situation that eventually the WS traffic for a great number of variables will be less than the associated SNMP traffic. But this observation is correct for request/reply messages, which is not the case in our trap investigation.

5.4 SOAP/SMTP versus SOAP/HTTP traffic

The configuration actions and policy tranfers made by the TLM are done through WS calls based on HTTP, as mentioned before. HTTP is used because it has request/reply communication, what is generally required by network administrators. However, these actions also can be done through WS calls based on SMTP.

To compare the traffic generated between SOAP/HTTP and SOAP/SMTP messages, policies with different size were sent by the TLM to a MLM. First it was used the prototype implementation with no modification. After, some modifications were made in the WS call on the TLM, and an alias and a script were created on a MLM to receive the WS call based on SMTP.

Six policies were sent, with size from 536 to 1122 bytes (policy content in XML format, without headers inserted in the WS call). Table 1 shows the size of policies in XML format and the amount of traffic generated by SOAP/HTTP and SOAP/SMTP messages in each one. The metric unit used is byte.

Table 1. Size of policies and the traffic generated.

Policy	Size	SOAP/HTTP Traffic	SOAP/SMTP Traffic
1	536	4451	4664
2	644	4602	4844
3	796	5849	5118
4	966	6990	5698
5	994	7476	5768
6	1122	8060	5862

As it can be observed in the table, the traffic generated by SOAP/SMTP messages is greater than the traffic generated by SOAP/HTTP messages when policies are smaller. When sending bigger policies the SOAP/SMTP messages increase slower than SOAP/HTTP messages. In the example of six policies presented, the size difference from the smallest policy to the biggest one was about 109%. The traffic generated by WS calls based on HTTP grew about 81% while the traffic generated by WS calls based on SMTP increased only about 25%. To a better view of the difference of traffic generated between the SOAP/HTTP and SOAP/SMTP messages, a graph is showed in figure 5.

In systems that more complex policies are used and the return of WS calls do not need to be instantaneous, WS based on SMTP seem to be a good solution. However, it must be observed the transmission delay. SMTP takes much more time than HTTP, because of SMTP servers delay.

6 Conclusion and Future Work

An important aspect of using WS for network management is that they are easy to implement, to support and they are good to interconnect different systems. Other important

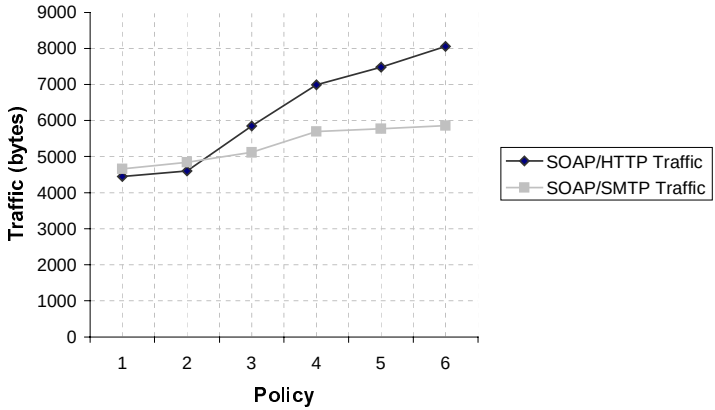

Fig. 5. SOAP/HTTP versus SOAP/SMTP traffic.

aspect is the delivery warranty of notification messages, making systems more trustable. SOAP over HTTP for network management have been investigated before, and could be considered relatively well known today. On the other hand, few is known about network management using SOAP over SMTP. There are two main reasons for that: support for SOAP over HTTP is far more deployed than support for SOAP over SMTP, and SMTP is suitable for asynchronous messages such as traps, but traps are far less used in SNMP than other synchronous messages such as GetRequest and GetResponse.

In this paper we have presented an architecture for distributed notification correlation where SNMP traps are translated to SOAP over SMTP asynchronous requests. Such translation is performed by gateways that forward notifications to MLMs, responsible to correlate events based on policies defined by the network operator. This architecture has been implemented and in this paper we have focused our attention in evaluating the traffic generated by SNMP, SOAP/SMTP and SOAP/HTTP.

The traffic generated by WS messages is much greater than the bandwidth consumed by SNMP messages. This is an important observation because previous investigations, focused on synchronous SNMP messages, have shown that in some situations the WS traffic can be less than the SNMP traffic. In this paper we show, however, that not only WS traffic is greater than SNMP, but also that the WS and SNMP traffic difference increases even more if more variables are present.

This fact could lead to the conclusion that WS-based notifications are not worthwhile. This is true if one considers only the traffic generation aspect. However, WS have integration facilities absent in SNMP. In addition, WS-based notification can cross different administrative domains because WS is based on Web protocols, while SNMP traffic is almost always restricted to local networks. These facts, on the other hand, does not help at all if the notification traffic imposed by WS makes a corporate network congested. In our opinion, there are competing forces related to WS notification issues, and bandwidth consumption, ease of use, and integration are some of them.

In the comparison of WS using different protocols for transport, WS based on HTTP showed to be better when few data has to be sent. The more the amount of data increase, SMTP seems to be better to use.

Future work in this direction includes further investigation on how notification correlation can reduce the traffic generation associated to WS. While we think that WS are required for inter-domain notification integration, we also believe that traffic generation can clearly present WS usage. In this scenario, reducing the traffic generation and still using an integration force such as WS could be reached via proper notification correlation.

References

1. J. Schönwälder, A. Pras, and J. P. Martin-Flatin, "On the future of internet management technologies," *IEEE Communications Magazine, Vol. 41, No. 10*, pp. 90–97, October 2003.
2. J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (snmp)," May 1990, IETF RFC 1157.
3. WWW Consortium, "Xml protocol working group," 2004, http://www.w3c.org/2000/xp/Group/.
4. J. Roy and A. Ramanujan, "Understanding web services," *IEEE Computer Society - ITPro*, pp. 69–73, Nov 2001.
5. G. Pavlou, P. Flegkas, S. Gouveris, and A. Liotta, "On management technologies and the potential of web services," *IEEE Communications, special issue on XML-based Management of Networks and Services*, vol. 42, no. 7, pp. 58–66, July 2004.
6. R. Chinnici, M. Gudgin, J. J. Moreau, and S. Weerawarana, "Web services description language (wsdl) version 1.2 part 1: Core language," June 2003, W3C Working Draft.
7. J. v. Sloten, A. Pras, and M. J. v. Sinderen, "On the standardisation of web service management operations," *EUNICE 2004 - Proceedings of the 10th Open European Summer School and IFIP WG6.3 Workshop*, pp. 143–150, June 2004.
8. 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 *IEEE/IFIP Network Operations and Management Symposium, NOMS*, Apr. 2004, pp. 715–728.
9. 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 (to be published*, May 2005.
10. T. de Castro and J. Nogueira, "An alarm correlation system for sdh networks," in *Proceedings Telecommunications Symposium - SBT/IEEE International ITS '98*, Aug 1998, pages 492 - 497, Vol. 2.
11. J. Zupan and D. Medhi, "An alarm management approach in the management of multilayered networks," in *Proceedings 3rd IEEE Workshop on IP Operations and Management (IPOM)*, Oct 2003, pages 77 - 84.
12. Sourceforge.net, "The net-snmp project home page," 2004, http://net-snmp.sourceforge.net/.
13. TP Group, "Php: Hypertext preprocessor homepage," 2004, http://www.php.net/.