On the Performance of Web Services Management Standards – An Evaluation of MUWS and WS-Management for Network Management

Giovane César Moreira Moura, Giancarlo Silvestrin, Ricardo Nabinger Sanchez, Luciano Paschoal Gasparly, Lisandro Zambenedetti Granville
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brazil
Email: {gcmmoura, gsilvestrin, rnsanchez, paschoal, granville}@inf.ufrgs.br

Abstract—Important steps have been taken in the recent years towards evaluating the performance of Web services for network management. Due to the lack of specific standards for Web services-based management, previous evaluations have been carried out measuring only the performance of SOAP (the basic Web services protocol) running in network management environments. While the conclusions of the papers published so far indicate the feasibility of employing Web services for network management, there is no evidence that these conclusions also hold for solutions developed according to recent Web services management standards. In this paper we go a step further and present the results of a set of experiments carried out in order to compare the specifications OASIS Management Using Web Services (MUWS) and DMTF Web Services for Management (WS-Management) against the de facto network management standard, i.e., the Simple Network Management Protocol (SNMP). The performance metrics investigated were network usage, response time, and CPU usage.

I. INTRODUCTION

During the last four years, the network management community has been investigating the performance of Web services technologies for network management [1], usually comparing Web services (WS) with the de facto management standard for TCP/IP networks, i.e., the Simple Network Management Protocol (SNMP) [2]. The main motivation for these investigations has been the concern that Web services messages – extensively based on verbose, textual eXtensible Markup Language (XML) documents – could negatively impact the performance of management tasks traditionally accomplished via SNMP, which employs more compact messages, encoded following the Basic Encoding Rules (BER) [3].

The main conclusion to be drawn from previous evaluations is that the performance of Web services does not prevent their use in the network management arena. For example, Pras et al. have shown that, depending on the number of management objects to be retrieved from a network device, SNMP may perform worse than Web services [4]. Similarly, it has been presented by Neisse et al. [5] and by Fioreze et al. [6] that with proper manipulations, Web services can have improved performance over SNMP even in management environments where “legacy” SNMP devices cannot be replaced.

Although these investigations have clarified that the Web services performance issue is not as critical as initially supposed, all evaluations have been performed using Web services that did not follow any specific Web services-based management standard. Meanwhile, however, the industry has been working on defining Web services specifications for service management, which encompasses the management of network devices as well. Nowadays, the two main standards in this area are the Management Using Web Services (MUWS) [7], [8] by the Organization for the Advancement of Structured Information Standards (OASIS), and the Web Services for Management (WS-Management) [9] by the Desktop Management Task Force (DMTF).

Different from the Web services created to carry out the first performance evaluations, management Web services defined according to either MUWS or WS-Management follows specific rules that are imposed with the objective of achieving system interoperability. The performance of these standard-compliant Web services, however, has not been evaluated so far.

In this paper we present the results of a set of experiments conducted to evaluate whether or not MUWS and WS-Management can be efficiently adopted for managing networks. We measured the network traffic generated by MUWS and WS-Management, as well as the response time observed by a network manager when interacting with three management agents that support MUWS, WS-Management, and SNMP. In order to perform a fair comparison, all these management agents have been implemented using the same programming language (in this case, Java) running in the same hardware infrastructure. In addition to network usage and response time, we also evaluated how much CPU each implementation uses to process management requests issued by the network manager.

The reminder of this paper is organized as follows. In Section 2 we review the related work, as well as MUWS and WS-Management specifications. In Section 3 we introduce the mapping from SNMP operations to MUWS and WS-Management. This mapping is used to carry out the evaluation presented in Section 4, where the management testbed is
characterized, and the experiments and results are discussed. Finally, Section 5 closes the paper with concluding remarks and perspectives for future work.

II. BACKGROUND

In this section we discuss the most relevant research work carried out so far to evaluate the pros and cons of using Web services for network management, and briefly review both MUWS and WS-Management standards.

A. Related Work

Several research work has been carried out in the past to assess the performance of Web services when applied to network management. These investigations differed in several aspects, including their focus, methodology, and results. In the next paragraphs we spot some of the most relevant work on this matter and the respective conclusions achieved so far.

Choi et al. [10] assessed the performance of XML-based management and compared it to SNMP. By developing an XML to SNMP gateway, the authors measured the network usage imposed by the communications (i) between the XML-based manager and the gateway, and (ii) between the gateway and the agents, concluding that the former performs better than the latter, i.e., the overhead between manager and gateway is half as much as between gateway and agents. The authors have also measured response time and computing resources usage (such as CPU and memory), showing – for the configured setup – that every gateway can handle 70 to 80 agents and that CPU is a more important computing resource than memory.

One of the first comparisons between the performance of SNMP and Web services for network management has been presented by Neisse et al. [5]. The authors presented two approaches for SNMP to Web services gateways: protocol-level and object-level gateways. While the former provided the mapping of SNMP primitives to Web services operations (e.g., Get, GetNext, and Set), the latter mapped high level Web services operations (e.g., GetIfTable) to multiple SNMP requests. Adopting a methodology similar to that of Choi et al. [10], the authors concluded that Web services always consume considerably more bandwidth than SNMP when protocol-level mapping is used. Object-level gateways, however, perform better than SNMP when the amount of instances to be retrieved is high.

Pavlou et al. [11] evaluated the performance of SNMP, CORBA, and Web services for management. To do that, the authors have developed their own agents using Orbacus CORBA platform, WASP Web services platform, and NET-SNMP and AdventNet SNMP platforms. They measured the response time and traffic for (i) a method returning the value of a single object of the TCP-MIB, (ii) a method returning the value of the eight dynamic attributes of the same MIB, and (iii) retrieving the information for all the TCP connections as a list of records (tcpConnTable). The conclusions were that Web services, being XML-based, incurred up to eight times the management traffic demanded by CORBA, and information retrieval times were approximately twice those observed for CORBA. In relation to SNMP, Web services performed even poorer.

An evolution of the aforementioned evaluation has been carried out by Pras et al. [4]. The authors have measured the performance of SNMP and Web services, trying – in addition to the previous work – to discover how much SNMP or Web services processing contributes to the total performance of the agent. To be able to determine that, the agents developed not only decoded and encoded messages, but also fetched data from within the system. The performance metrics investigated were network usage, response time, and computing resources usage. The results obtained showed that SNMP is more efficient in cases where a single object is retrieved, while for larger number of objects Web services may be more efficient. Moreover, the authors revealed that the CPU time demanded to code and compress messages is less than the time required to retrieve current data from the system (and can be neglected if the number of objects to be retrieved is high). These results substantiated the claim made by the authors that “there is no convincing reason to refuse Web services for network monitoring.”

Recently, investigations have been carried out to assess Web services performance to manage particular scenarios. For example, Fioreze et al. [6] reported the results of an investigation about the use of Web services in the context of management by delegation. In other work, Lima et al. [12] evaluated the performance of Web Services and SNMP-based notifications taking network usage and notification delivery delay into account.

There is no doubt that the previous investigations on Web services performance were very clarifying and helped to demystify that Web services are prohibitive for network management. However, the results achieved until now must be confirmed taking into account the new standards upon which depends the widespread adoption of Web services in the network management arena. As far as we are aware of, this is the first research work to present an objective analysis on this particular topic.

B. Web Services-Based Management Standards

The increasing interest of the community to integrate management applications through Web Services has demanded efforts to specify and standardize vendor-neutral, platform-independent protocols. Two specifications have been standardized recently: Web Services Distributed Management [7], [8], [13] by OASIS on March 2005 and Web Services for Management [9] by DMTF on April 2006. Although it may be too early to predict the adoption of these standards by the industry, the fact that both of them are being supported by important players suggests that they have the potential to become de facto standards for service management. In the following sub-sections we present a general view of both standards. A reduced set of them has been used later on in our experiments.

1) Web Services Distributed Management: The Web Services Distributed Management (WSDM) is a specification
developed and standardized by the OASIS Web Services Distributed Management Technical Committee. It is organized in two sets of specifications: Management Using Web Services (MUWS) [7, 8] and Management of Web Services (MOWS) [13]. The former enables management of distributed resources using Web services, whereas the latter addresses management of Web services endpoints using Web services protocols. In this paper we focus on MUWS, since it is the specification that provides functionality comparable to SNMP and WS-Management.

MUWS defines how to make an arbitrary resource manageable via Web services. To accomplish this objective, it relies on a number of Web services specifications such as XML Schema [14, 15], Web Services Description Language (WSDL) [16], Web Services Addressing (WS-Addressing) [17], Web Services Resource (WS-Resource) [18], Web Services Notification (WS-BaseNotification) [19], and Web Services Security (WS-Security) [20].

A Web service endpoint provides access to a manageable resource. A manageability consumer (e.g., a management system) discovers the Web service endpoint and exchanges messages with it. Table I summarizes the set of Message Exchange Patterns (MEPs) that MUWS provides for a manageability consumer to interact with a manageable resource. Three types of interaction are covered. A manageability consumer can either retrieve management information about a manageable resource or affect the state of a manageable resource by changing its management information. In addition, a manageable resource may inform, or notify, a manageability consumer of significant events.

2) Web Services for Management: The Web Services for Management (WS-Management) is a specification developed by a consortium formed by Sun, Microsoft, NEC, Intel, Dell, and other companies. The specification has been submitted to the Distributed Management Task Force (DMTF) in September 2005 and released as a preliminary standardization effort by the DMTF in April 2006.

As opposed to WSDM, WS-Management is organized in a single document. It covers only part of what WSDM addresses: how to identify a manageable resource and communicate with it. This overlapping area is within MUWS. In consonance to the intrinsic composability nature of Web services, WS-Management also depends on a set of specifications such as XML Schema, WSDL, WS-Addressing, Web Services Eventing (WS-Eventing) [21], Web Services Enumeration (WS-Enumeration) [22], Web Services Transfer (WS-Transfer) [23], and WS-Security.

WS-Management exposes a common set of operations, which are relevant to systems management. Table II enumerates some of these operations. As one can observe, the specification provides the ability to get, put, create and delete management resources, enumerate the contents of containers and collections, subscribe to events generated by managed resources, and execute customized methods.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>WSDM MESSAGE EXCHANGE PATTERNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests for Property Information</td>
<td></td>
</tr>
<tr>
<td>GetResourceProperty</td>
<td>Retrieves the value of a specified resource property for a resource</td>
</tr>
<tr>
<td>GetMultipleResourceProperties</td>
<td>Retrieves values for a specified set of resource properties for a resource</td>
</tr>
<tr>
<td>QueryResourceProperties</td>
<td>Retrieves part of the resource properties document from a resource (using e.g., XPath)</td>
</tr>
<tr>
<td>QueryRelationshipsByType</td>
<td>Retrieves information on a particular type of relationship in which the resource participates</td>
</tr>
</tbody>
</table>

| Commands to the Resource |
| SetResourceProperties | Modifies (inserts, updates, and/or deletes) the specified properties for a resource |

| Subscriptions and Notifications |
| Subscribe | Requests that notifications be sent to a manageability consumer |
| GetCurrentMessage | Requests that a notification producer for a resource return the last notification on a topic |
| PauseSubscription | Requests a temporary hold on an existing subscription |
| ResumeSubscription | Requests the re-activation of a paused subscription |
| Notify | Receives notifications on behalf of a consumer |
| RegisterPublisher | Creates the registration of a resource as a notification publisher at a NotificationBroker |
| Destroy | Destroys the registration of a resource at a NotificationBroker |

The main difference between MUWS and WS-Management specifications stands in their focus. While the former is a richer solution that is geared to manage distributed systems, composed of resources of very dissimilar types, the latter is a more lightweight specification. Work is currently underway to converge WSDM with WS-Management [24]. The objective is to develop a common set of specifications for resource management that can be broadly supported across multiple
platforms. This paper, in this context, provide useful information that can be used to substantiate the reconciliation and convergence of the current specifications.

### III. WEB SERVICES, SNMP, AND NETWORK MANAGEMENT

Although both MUWS and WS-Management have been originally designed for general service management, they can be specifically employed for network management as well. In such a context, Web services specifications can be compared with SNMP, the de facto standard for the management of TCP/IP networks. In this section we present how MUWS and WS-Management can perform operations similar to those offered by SNMP to retrieve information stored in conceptual management tables exposed by agents located inside managed devices. We restrict our study by neglecting the management operations that write information on a table because the performance of such operations is usually a less critical issue. That is so because changing the internal status of a managed device normally occurs in an on-demand basis, while periodical retrieval of status is much more frequent in monitoring processes.

#### A. Table Operations

Tables are quite common in SNMP Management Information Bases (MIBs), and thus can serve as a proper subject for our evaluation. In an abstract view, a table of management information is composed of columns that comprise different attributes of the same managed resource. The rows of a table describe multiple resources of a same type or class. For example, a device’s routing table is composed of several rows, each one presenting a different route. Each route, in its turn, is described according to the routing table columns that define, for example, the destination network, the network mask, and the route cost.

![Fig. 1. Operations over tables of management information](image)

Over a management table one can perform a basic operation of retrieving a set of cells given the following input parameters: (i) a list of columns of interest, (ii) a pointer to the first row of interest, and (iii) the number of rows of interest that follows the first row. All other retrieval operations (e.g., retrieving the whole table, retrieving a single cell, etc.) are specializations of this basic one. Figure 1 exemplifies possible operations over a management table.

In the following sub-sections we present how SNMP, MUWS, and WS-Management realize the basic retrieval operation over a concrete management table, assuming that a network manager wants to access a managed device in order to download the device’s routing table defined according to the IETF IP Forwarding Table MIB [25].

#### B. SNMP Support for Table Manipulation

SNMP request messages to retrieve information from a managed device are *get*, *get-next*, and *get-bulk*. The *get* message is more frequently used for scalars (i.e., objects with a single instance associated to them), while *get-next* and *get-bulk* are more frequently used for tables. We will thus continue our explanation considering only *get-next* and *get-bulk*.

Objects defined in a MIB table model the columns of that table. Table rows have no specific denomination, while cells of a table are object instances, which are essentially variables. When sending a *get-next* or *get-bulk* request, the network manager selects the columns of interest by listing them in a sequence of variable bindings attached to each SNMP message. The columns in the list do not need to be consecutive. In order to select the first row of interest, the manager appends to each column the identification of the predecessor row. If the row of interest is also the first row of a table, the identification of a predecessor is not provided.

One important difference between *get-next* and *get-bulk* is related to the number of rows of interest. In *get-bulk* messages there is a specific field (*max-repetitions*) used by the manager to inform this value to the agent. In *get-next*, such a value is not supported, and the agent assumes that only one row is of interest. This restriction forces the manager using *get-next* to repeatedly issue new requests until all desired rows have been retrieved, which increases the bandwidth consumption and response time. Figure 2 depicts the message exchange between a manager and an agent using SNMP’s *get-next* and *get-bulk*.

![Fig. 2. get-next versus get-bulk](image)

It is important to highlight that with proper use of either *get-next* or *get-bulk* an SNMP manager can perform several different retrieval operations such as those previously presented in Figure 1. This is important because not all technologies addressed in this paper provide this flexibility.
C. MUWS Support for Table Manipulation

MUWS is based on the concept of manageable resource and resource property, which we use to model tables of management information. Each row of a table is mapped to a MUWS resource, while each column defines a resource property. Considering this mapping, MUWS supports manipulation of tables as follows.

To retrieve a single cell, i.e., a specific property of a single MUWS resource, the network manager issues a GetResourceProperty request message. To retrieve multiple columns (properties) of a single row (resource), one must use a GetMultipleResourceProperties request. Properties are listed in the XML document that forms the request message and, as in SNMP, do not need to be consecutive. If the manager is interested in all columns of a row, it must list all columns one by one in the GetMultipleResourceProperties request message.

In order to retrieve several rows with multiple columns, the manager needs to send various GetMultipleResourceProperties request messages, each one informing all properties (columns) of interest. In MUWS, there is no message able to request multiple rows (resources) from a table of management information. It means that there is no SNMP get-bulk alike message in MUWS. In this sense, GetMultipleResourceProperties operates similarly to the SNMP get-next message.

Comparing MUWS messages with SNMP messages, the former are far more complex than the latter because they need to follow a set of rules defined in different standards, while SNMP messages just need to follow the Basic Encoding Rules (BER) [3]. In addition, different XML message formats can be employed to achieve the same MUWS behavior. The real final format used in a management environment will strongly depend on the MUWS framework used to implement managers and agents. For example, in our evaluation we have used the Apache Muse [26], which enforces its own message format.

The XML document presented in Figure 3 is a MUWS request assembled by Muse to retrieve the destination (ipCidrRouteDest), the network mask (ipCidrRouteMask), and the type of service (ipCidrRouteTos) of a routing table row of a device whose IP address is 143.54.12.157.

As one can observe, this Muse XML document is not optimized because, for example, it repeats several namespaces along the message. Although the MUWS specification is flexible enough to define similar but optimized documents with the same semantic of the previous one, the current Muse implementation does not provide any means to produce optimized messages.

Figure 4 presents an excerpt of the MUWS response informing network address, mask, and type of service for the requested row in the managed device.

D. WS-Management Support for Table Manipulation

Similarly to MUWS, WS-Management also models management information as resources and resource properties, which are retrieved using a single Get request message. The basic Get semantic does not allow the manager to indicate which properties (columns) are of interest, thus implicitly selecting on the agent all properties by default. An additional mechanism called fragment-level transfer allows a manager to select the columns of interest inserting XPath expressions in the header of the Get message, but this mechanism is optional and may not be implemented by a WS-Management agent.

In our evaluation of WS-Management, we have used the Wiseman framework [27], which, unfortunately, does not support the fragment-level transfer mechanism. This prevents the manager from selecting the properties (columns) of interest. However, since in our experiments we were interested in all columns of each resource, the absence of fragment-level transfer did not represent a drawback. Quite the opposite, this made even more evident the restriction of MUWS to retrieve all properties only by listing all of them in the request message.

Comparing Wiseman and Muse in terms of ease of use, Wiseman requires harder coding efforts from the agent developer. Little automation is present in Wiseman, forcing the developer to manually handle core XML elements that are, on the other hand, provided by the Muse API. That, however,
brings more flexibility to the developer that can optimize the final XML messages if required. Some optimizations, in fact, are provided by Wiseman on its own. For example, the SaveAggressiveNamespaces option reduces the size of WS-Management messages decreasing the number of namespace repetitions. This can be observed comparing Figure 5 (that shows a Get request message) and Figure 3 (that previously presented a MUWS GetMultipleResourceProperties request message).

![SOAP message for WS-Management Get request](image)

**Fig. 5. SOAP message for WS-Management Get request**

Like in MUWS, WS-Management resources are accessed one by one, because Get does not support the retrieval of several different resources (rows) in a single request. Figure 6 complements Figure 5 presenting the response sent from the WS-Management agent to the requesting manager.

**IV. EXPERIMENTAL EVALUATION**

In this section we present the experimental evaluation carried out to compare the performance of MUWS, WS-Management, and SNMP. We first describe the test environment and then address the methodology employed to assess network usage, response time, and CPU usage, together with the results obtained.

**A. Measurement Setup and Prototypes**

The testing environment was composed of two Linux PCs connected through a 3Com Superstack 3 3300XM switch at 100Mbps. The PCs have been used to execute the three pairs of management applications/agents, each pair corresponding to a different protocol being tested, i.e., MUWS, WS-Management, and SNMP. Table III summarizes the hardware and software configuration of the environment.

**TABLE III**

<table>
<thead>
<tr>
<th>Management Application</th>
<th>CPU</th>
<th>RAM memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUWS</td>
<td>Pentium 4</td>
<td>256 MB</td>
</tr>
<tr>
<td>WS-Management</td>
<td>Pentium 4</td>
<td>256 MB</td>
</tr>
<tr>
<td>SNMP</td>
<td>Pentium 4</td>
<td>256 MB</td>
</tr>
</tbody>
</table>

For our study three different agent prototypes have been implemented. The first was a MUWS agent implemented with Apache Muse [26]. The second was a WS-Management agent developed with Wiseman [27]. The third – an SNMP agent – was built using the SNMP4J Agent Toolkit. Java was used as the programming language to develop the three agents because the only open source reference implementations of both MUWS and WS-Management are coded in Java and we wanted to evaluate all agents under similar conditions.
The agents were built to retrieve rows from the \texttt{ipCidrRouteTable} [25]. Each row is comprised of 16 columns, as follows:

- \texttt{RouteDest}: destination IP address of the route;
- \texttt{RouteMask}: mask to be ANDed with destination IP;
- \texttt{RouteType}: type of service;
- \texttt{RouteNextHop}: address of the next system en route;
- \texttt{RouteIfIndex}: local interface through which the next hop of the route should be reached;
- \texttt{RouteProto}: how the route was learned;
- \texttt{RouteAge}: when this route was last updated;
- \texttt{RouteInfo}: reference to MIB definitions;
- \texttt{RouteNextHopAS}: Autonomous System of next hop;
- \texttt{RouteMetric1}: primary routing metric for the route;
- \texttt{RouteMetric2}: alternate metric for the route;
- \texttt{RouteMetric3}: alternate metric for the route;
- \texttt{RouteMetric4}: alternate metric for the route;
- \texttt{RouteMetric5}: alternate metric for the route;
- \texttt{RouteStatus}: row status variable.

The code used to fetch routing information from the system was similar in Web services and SNMP prototypes. Whenever the agents receive a request, they access the table, perform a search to find the correct route, and return the values of the objects requested. This process is repeated for every object informed within the same request. Regarding routing table, it was populated with 128 routes used by one of the point of presence routers of the Brazilian Education and Research Network (RNP).

### B. Network Usage

To measure network usage we have configured the management applications to retrieve a varying number of \texttt{ipCidrRouteTable} rows (2, 4, 8, 16, 32, 64, and 128) from the agents. In the case of SNMP, the management application has been configured to execute \texttt{get-next} operations\(^1\), one for each row to be retrieved. Each message consisted, in addition to the header, of a varbind list containing 16 varbinds (all columns from \texttt{ipCidrRouteTable}). The MUWS management application, in its turn, has executed \texttt{GetMultipleResourceProperties} operations, one for each row. The request and response messages were similar to those illustrated in Figures 3 and 4, with the difference of requesting/informing the 16 properties (columns) of the row. Finally, the WS-Management application has issued \texttt{Get} operations. However, since it by default retrieves all management information associated to the resource (a row of the table), the request has not indicated any property, while the response message contained the values of the 16 properties of the requested row.

We have used Ethereal [28] to capture the messages exchanged between management applications and agents. To compute the network usage we have considered the octets generated by all layered protocols from the application layer down to the data link layer. The results shown in Figure 7 confirm that, as expected, SNMP consumes less bandwidth than both MUWS and WS-Management regardless of the employment of gzip compressed messages. It remains a factor 9 better than “pure” WS-Management and a factor 16 better than “pure” MUWS, independent of the number of transferred rows. This is mainly due to the fact that SNMP performs binary encoding of its messages, while MUWS and WS-Management employ textual, XML messages. Besides, MUWS and WS-Management, which rely on TCP, demand extra traffic related to connection, disconnection, and acknowledgment signalling.

Regarding the difference between the Web services-based management standards, WS-Management performed better than MUWS. As one can see in Table IV, the explanation for this is that MUWS request messages were 142% larger than the ones employed by WS-Management (2,354 x 969 octets). For the response messages, the difference was of about 72% (2,404 x 1,390 octets).

![Network usage](image)

**TABLE IV**

<table>
<thead>
<tr>
<th>MUWS x WS-MANAGEMENT MESSAGES IN NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
</tr>
<tr>
<td>MUWS</td>
</tr>
<tr>
<td>WS-Management</td>
</tr>
</tbody>
</table>

| Specification | Header | Body | Envelope | Total |
| MUWS          | 529    | 1,679| 196      | 2,404 |
| WS-Management | 329    | 667  | 394      | 1,390 |

\(^1\)It has been already demonstrated by another research [4] that \texttt{get-bulk} operations are always more efficient than \texttt{get-next}. Hence, we have carried out the SNMP experiments using only \texttt{get-next} because it represents the worst case scenario.
the properties to be retrieved. In such situations, the network traffic generated by WS-Management would get closer – but still be lower – than that observed for MUWS, since the response of MUWS is much more verbose (2,404 x 1,390 octets).

The difference between the network usage of MUWS and WS-Management is virtually suppressed when gzip compressing is introduced. Both MUWS and WS-Management compressed traffic consumes almost the same bandwidth. On the other hand, the employment of secure messages exacerbates the difference: MUWS secured messages consume far more bandwidth than WS-Management messages. If compression and security are combined and employed in the same messages (first compressing then securing them), the bandwidth consumption is higher, but the difference between MUWS and WS-Management is very low, again due to the compression. One interesting point to observe in Figure 3 is that “pure” WS-Management bandwidth consumes very similar to compressed and secured MUWS, as compressed and secured WS-Management. This inc that, concerning bandwidth consumption only, the “pure” WS-Management has no advantage over compressed and secured WS-Management.

We wanted our evaluation to reflect what is possible to achieve with current implementations of both WS-based management standards. Since Wiseman – the only open implementation of WS-Management – still does not support fragment-level transfer, we have not measured its impact on network usage.

C. Response Time

We have measured the average response time for the same management operations used in the previous experiment. To do that we have instrumented the management applications with two calls to the method System.currentTimeMillis, which have been inserted before (t1) and after (t2) the loop to request n rows from the agents. By subtracting t1 from t2, the time needed to retrieve the requested rows could be determined (\(\Delta t\)). In addition, we have inserted calls to System.currentTimeMillis inside the agents before (t3) and after (t4) the procedure to fetch information from the system in order to determine how much time is consumed by this task (\(\Delta Fetching = \sum_{i=0}^{num.objects} (t_{i+4} - t_{i+3})\)). The difference between \(\Delta ResponseTime\) and \(\Delta Fetching\) corresponds to the delays associated to:

- processing/dispatch of the request messages by the management application;
- processing of the requests when they arrive at the agent;
- codification of the results – BER (SNMP) or XML (Web services);
- dispatch of the response messages by the agent;
- receipt of the response messages by the management application.

To obtain statistically sound results, each experiment was repeated 30 times. Adopting degree of confidence of 95%, the confidence interval seen for any experiment was no larger than 105 milliseconds.

Figure 8 summarizes the results obtained. Examining the figure, one can notice that WS-Management has performed worse than SNMP and much better than MUWS. Taking into account that the hardware used to execute the management applications/agents was the same (see Table III) and the time consumed by the agents to fetch information from the system was removed from the calculations, one can infer that the overhead to process/encode MUWS messages is much higher than the one imposed by both WS-Management and SNMP. Although the Application Programming Interfaces (APIs) used by the MUWS management application/agent have not been analysed, they may have influenced these results.

![Response time](image.png)

**Fig. 8. Response time**

To make a fair comparison between the Web services and SNMP, the SNMP agent has been developed in Java. The results obtained (specially for WS-Management and SNMP) suggest that the overall computational cost of the Web services and SNMP prototypes is similar.

Once compressed and secured messages are employed, the response time is affected. The introduction of gzip increases the response time proportionally to the original response time of “pure” MUWS and WS-Management, i.e., using MUWS with gzip is more expensive than using WS-Management with gzip. That is so because the larger the messages, the slower the compression process. The difference in the response time of MUWS and WS-Management with the introduction of SSL, however, is lower. That is so because the encryption process is less dependent of the message size, which turns the response time of secure MUWS and WS-management closer to each other.

D. CPU Usage

To determine CPU usage we have patched the `ps` utility to output CPU time in millionths of a second. Adopting a methodology similar to the one used in the previous experiment, we have executed `ps` – in the PC where the agents
were running – before receiving the first request \( (t_1) \) and after sending the last reply message \( (t_2) \) related to a sequence of request messages issued by the management application to retrieve 2, 4, 8, 16, 32, 64, and 128 rows. We have derived the amount of CPU time consumed subtracting \( t_2 \) from \( t_1 \). Notice that the CPU time consumed to fetch information from the system was included (but was the same for the three agents).

Table V shows the average values obtained after 30 executions of each request sequence. One can observe that both MUWS and WS-Management have consumed more CPU than SNMP. SNMP remains a factor 2 to 3 better than WS-Management and a factor 3 to 4 better than MUWS. Actually, this factor increases with the number of retrieved rows. Furthermore, the high amount of CPU consumed by WS-Management did not follow the tendency observed in the previous experiment in which the response time measured for WS-Management was lower than the one measured for SNMP.

![Table V](image)

<table>
<thead>
<tr>
<th>Rows</th>
<th>SNMP</th>
<th>MUWS</th>
<th>WSM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(gzip)</td>
<td>(ssl)</td>
<td>(gzip+ssl)</td>
</tr>
<tr>
<td>2</td>
<td>0.04</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>0.07</td>
<td>0.09</td>
<td>0.08</td>
</tr>
<tr>
<td>8</td>
<td>0.10</td>
<td>0.21</td>
<td>0.13</td>
</tr>
<tr>
<td>16</td>
<td>0.16</td>
<td>0.41</td>
<td>0.25</td>
</tr>
<tr>
<td>32</td>
<td>0.30</td>
<td>0.89</td>
<td>0.55</td>
</tr>
<tr>
<td>64</td>
<td>0.61</td>
<td>2.24</td>
<td>1.35</td>
</tr>
<tr>
<td>128</td>
<td>1.19</td>
<td>5.11</td>
<td>3.59</td>
</tr>
</tbody>
</table>

We should point that a percentile of the amount of CPU is consumed by the Java Virtual Machine, upon which the three agents were executed. In the case of MUWS and WS-Management agents, there is the additional consumption required by Tomcat.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have compared the performance of two Web services-based management specifications – Management Using Web Services (MUWS) and Web Services for Management (WS-Management) – against SNMP. Although several experiments have been carried out so far to evaluate whether or not Web services can be applied for network management, this is the first investigation to address the performance of these specifications. We believe the results provide guidance and insights for the convergence effort currently underway.

The experiments related to network usage have shown that management Web services compliant to either MUWS or WS-Management are much more verbose than those used in previous investigations (9:1 and 16:1 in relation to SNMP). Although there is some room to lower network usage (e.g., by choosing shorter attribute names), the size of SOAP request and response messages will still be much larger than SNMP messages, given the huge differences of size observed. Previous investigations [5], [4], however, have demonstrated that XML-encoded management data can be efficiently compressed, which has been confirmed in our investigation.

Regarding response time, we have been able to isolate and determine the overhead of processing request/response messages in the Web services and SNMP prototypes. WS-Management has performed about 38% worse than SNMP, while MUWS presented half the performance of SNMP. In the worst case scenario, MUWS incurred 40.38 milliseconds (5.169/128) overhead to retrieve a table row (plus fetching). As expected, the introduction of security and compressing, however, increases the response time but not at a point that would prevent their use.

The adoption of Web services come at the price of demanding much more computing resources such as CPU. Although we have not evaluated the amount of memory consumed, we have observed that the WS-Management agent has allocated 35 MB of RAM memory at startup, while the MUWS agent has allocated 52 MB. Most of this memory is used by the Java Virtual Machine, Tomcat, and the agent.

Putting MUWS and WS-Management in perspective, WS-Management presented the best results for the three metrics investigated: network usage, response time, and CPU usage. These results reflect the nature of WS-Management, which is informally declared to be more lightweight than MUWS. Although WS-Management imposes less bandwidth than MUWS, it is still very verbose. That may be a major concern if it is considered to operate in restricted bandwidth environments; fortunately, such environments are becoming rare today. Furthermore, the available implementations of WS-Management and, particularly, MUWS consume many computing resources and, therefore, should be diminished/ported if it is supposed to run in network devices with scarce resources.

As future work we intend to investigate how to improve the performance of the Web services compliant with the management standards by tuning the execution environment, at both manager and agent sides. In addition, we also plan to assess the impact of introducing security support based on the WS-Security specification.

REFERENCES


