Efficient Information Retrieval in Network Management Using Web Services

Aimilios Chourmouziadis and George Pavlou

Center of Communications and Systems Research, Department of Electronic and Physical Sciences, University of Surrey,
GU27XH Guildford, United Kingdom
{A.Chourmouziadis, G.Pavlou}@surrey.ac.uk
http://www.ee.surrey.ac.uk/CCSR/

Abstract. Web Services is an XML-based technology that has attracted significant attention for building distributed Internet services. There have also been significant efforts trying to extend it to become a unifying management technology. An all-encompassing management technology needs to support efficient information retrieval, scalable event management, transaction support for configuration management and also security. Previous technologies, such as CMIP, SNMP and CORBA have addressed these aspects poorly, partially or at a high cost. This paper proposes an approach to address efficient information retrieval in terms of both bulk and selective data transfer. In order to achieve this, services modelling management information need to be organized in a hierarchy through service association. In order to achieve service association, information metadata are defined in secondary endpoints compared to the ones where services are deployed and accessed. We have defined a language for expressing arbitrarily complex information retrieval expressions and implemented a parser at the object level that evaluates these expressions, navigates arbitrary service associations and returns the results. We demonstrate the use and usefulness of the approach in an example usage scenario.

1 Introduction

Since the introduction of the Simple Network Management Protocol (SNMP) in the early 1990's and the versions of it that followed, its wide deployment for sophisticated network management still raises a lot of concerns. In the 2002 IAB Network Management Workshop [1] it became evident that SNMP can not be used for sophisticated management since its inneficiencies limit its potential usage to relatively simple monitoring. Therefore, alternative technologies are required to meet management goals such as efficiency in information retrieval, transaction support, security and also reduced development & operational costs. Distributed object technologies and, in particular, the Common Object Request Broker Architecture (CORBA) was considered as a unifying management technology and, although it has come a long way since then, it still has serious inefficiencies. In Corba federation and bulk retrieval are not supported, filtering capabilities lack expressiveness, scalability

R. State et al. (Eds.): DSOM 2006, LNCS 4269, pp. 1 – 12, 2006.

is an issue in addition to the large agent footprint. More recently, the introduction of Web Services, coupled with the advent of maturing eXtensible Markup Language (XML) technology standards, is seen as a promising approach for faster product development, tighter system integration and robust device management [2].

Web Services (WS) is an XML technology that encompasses W3C standards such as the Simple Object Access Protocol (SOAP) [3], the Web Services Definition Language (WSDL) [4] and the Universal Discovery Description and Integration protocol (UDDI) [5]. Since all these have their CORBA equivalents [7], WS can be used for distributed network management in a similar fashion to CORBA. But can they address this goal efficiently? Researchers in [6] and [7] compared the performance of WS, CORBA and SNMP. The conclusion was that when the amount of information to be retrieved increases, so does the efficiency of WS in comparison to SNMP. Smaller amounts of data though results in higher traffic for WS. The performance of WS, in terms of coding and latency, is poor in comparison to CORBA and SNMP.

Though the measurements in [6] and [7] show that WS could only be used in management scenarios where large amounts of data need to be exchanged, this is not necessarily true. WS performance at this stage can yield ambiguous results. As discussed in [8] and [9] approaches to resolve issues such as parsing, tranport problems, compression and data serialization etc, are still immature. Moreover the support WS provide to create sophisticated requests needs also to be investigated. Emulating the behavior of SNMP's operations such as GetNext and GetBulk is not a good practice when using WS. Such practices deprive any WS-based framework from the ability to use alternative sophisticated approaches to perform operations such as complex information retrieval. Performance and capabilities are thus inhibited.

In this paper we introduce a sophisticated approach to achieve true bulk or selective information retrieval, a capability that only CMIS/P offers among all management technologies, albeit at the cost of complexity and adherence to OSI upper layers that are not used widely anymore. In comparison, SNMP has limited support for bulk retrieval, mainly due to its mapping to UDP, and has no selective retrieval capabilities. Finally, CORBA lacks explicit support for such functionality.

Since one of our goals is to provide solutions for real management information retrieval scenarios, we have used SNMP MIBs modeled as web services to which retrieval scenarios are applied. In order to facilitate information retrieval it was important to come up with a way to organize data and services in a hierarchy that allows navigation of the information being held. To do this, we came up with a scheme to associate services and define arbitrary relationships between them. This hierarchical organization allows us to employ schemes of selective or bulk retrieval. This is done by deploying a parser at the object level on the agent side that accepts requests in the form of queries from a manager expressed in a language we designed. The agent uses the parser to interpret these queries and respond to the manager with the data collected from a list of management web services the agent has access to.

The remainder of this paper is structured as follows. In section II, we provide an analysis of our system model. In section III we present details on how service association is performed and how arbitrary service relationships can be defined. Section IV discusses details about the information retrieval grammar and the parser we developed. In section V we present a usage scenario that demonstrates the use and usefulness of our approach. Finally, in section VI we present our conclusions.

2 Service Design

Since the appearance of distributed object technologies, researchers realized the importance of converting SNMP and CMIS/P management information models. This led to the establishment of the Joint Inter Domain Management taskforce (JIDM) that produced a translation between the SNMP SMI / CMIS/P GDMO and CORBA's Interface definition Language (IDL). A JIDM-like translation though of SNMP SMI to services would be problematic for the same reasons encountered with its mapping to IDL. First bulk or selective retrieval is not supported. Second, scalability problems may arise when vast amounts of dynamic entities, i.e. SMI table rows, are modeled as separate services. As such, we considered an emerging approach for semantic translation of SNMP MIB's [7]. In such a translation, there may exist service methods for returning commonly accessed groups of single instanced objects, e.g. packet counters. In addition, tabular information is accessed collectively through a method that returns a table as a list of records. Additional methods may support SNMP-like "get next" or "get N next, i.e. get bulk" functionality. This type of modeling adds support for bulk data transfer for multiple instanced objects. Still, selective retrieval is not supported.

2.1 Information Retrieval Approaches

In WS bulk and selective information retrieval operations could be performed in two ways. The first method involves performing filtering actions at the SOAP level with a tool such as the XML Path Language (XPath) or similar. Since SOAP's body is in XML, XPath can be used to selectively pick portions of it. Such an approach is not problem-free though. Initially, extra latency is added in the retrieval process because more data need to be accessed, retrieved and coded in the SOAP body. Moreover, according to views expressed in the Network Configuration protocol mailing list, XPath is a very heavy tool for filtering purposes. Even a cut down version of XPath may still be resource intensive. A second approach to address selective retrieval is to perform it within the object code that represents a WS. As such we perform filtering before formation of the SOAP body, avoiding extra latency and keeping resource usage low by binding selective retrieval to the needs of the information model.

2.2 Modeling Approach

Supporting bulk retrieval for both single instance and multiple instance entities requires a collective view of management data across all levels of the data structure. To achieve this for SNMP, every MIB is modeled as a service. A level higher from the service level an agent has a collective view of all services, organized in a hierarchy through service association. The association scheme allows for arbitrary relationships to be defined. The agent in the scheme uses a parser to decide based on string expression queries it receives, the services from which data must be retrieved. Thus the agent has both a collective view and a selective capability over the services underneath it. At object code level, every service contains single instance and multiple instance entities of SNMP data modeled as simple values and tables. Bulk retrieval is achieved by three methods with different views on data. One method has access to single instance data, the other has view on table data and the third method has view of

all the object data. All methods receive string arguments that represent command line queries that are interpreted by a parser we have built which decides which data will be sent to the manager requesting them. As such, all methods have both bulk and selective access to the data in their view. In Fig. 1 the translation of the SNMP's data into services is given. Modeling information this way initially allows a collective view of information at various levels (low level of granularity). At the same time a selective capability upon any data structure (service, simple data, tables) is offered (high level of granularity). Thus, our approach not only allows us to perform selective or bulk information retrieval but also to keep the number of services representing management data low. Therefore, it tries to avoid complexity and scalability problems.

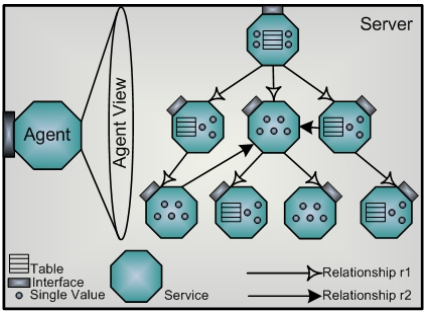

Fig. 1. Modeling approach

3 WS Association

From the information modeling approach presented in the previous section, it is evident that our scheme has two requirements. Firstly, it requires some means to define logical or physical relationships between services. These relationships will make navigation and selection of services feasible. These relationships provide an agent with a hierarchical view of the services underneath it and easy access to their data. A second requirement is that these relationships must be arbitrary so that traversal can be based on any relationship. The idea of navigation of arbitrary relationships between entities modeling management data is originated in [10]. The authors there propose to make data retrieval more efficient in CMIS/P by allowing traversal of objects based on any relationship and not only containment. Our concept is to define different relationships between services and make navigation possible based on them. Services though have different access requirements than objects (i.e. more latency). Thus, we decided that it is more efficient to make service selection first and only then to apply selective actions on the data, in a similar fashion to what CMIS/P does with scoping and filtering. With these two requirements satisfied, our agent can selectively pick up services according to string expression queries it receives from a manager.

The remainder of part III continues as follows. In section 3.1 we present the concept of navigating the relationships that are defined between services. In section 3.2 we present how to define such relationships.

3.1 Navigation and Selection of Services

The common view for a hierarchical organization of entities is that of a tree, where elements in the previous level of the tree are connected with containment relationships with the ones in the next level. Navigation among the elements in this tree is based on containment. If these elements were services capturing SNMP MIBs and the relationships between these services were arbitrary, then a tree such as the one depicted in Fig. 2 can be defined.

Navigation of this tree and selection of services by an agent is possible in our scheme, by defining simple expressions that identify a starting point for the search, level restrictions for service selection and relationship patterns to follow. A simplified Backus Naur Form (BNF) [11] syntax for such an expression is the following:

<path_selection_exp>::={< startpoint_tag> , <minlevel_tag> , <max-level_tag> , <pattern_tag>} . (1)

<pattern_tag>::=<identifier> | <pattern_tag> **.** <identifier> | (<pattern_tag>)! . (2)

<min_level_tag>::=<integer> . (3)

<max_level_tag>::=<integer> . (4)

<startpoint_tag>::=<identifier> . (5)

The path selection expression is dispatched from a manager to an agent in the form of a "command line" query expression. The agent uses it to select services based on relationship patterns, the level restrictions and the starting point tag. Selective retrieval of data from these services is performed only after selection of services has been completed. This is achieved by using the parser developed to interpret several other expressions that we will present later on. In order to demonstrate the usage of the path selection expression, some examples are given.

1) Path selection expression with no Restriction: A path selection expression such as the one in equation 6, if used with the information tree of services shown in Fig. 2, will cause a number of actions. Upon receiving the expression, the agent will use the parser to evaluate its validity. If the expression is valid then the agent will evaluate the expression and will start searching the sub-tree defined from the starting point (Root in this case) for services which can be reached by following relationships first of type r1 and then of type r2. The services selected are highlighted in Fig. 3. If selective retrieval expressions are also dispatched, the agent will only return the values that match the criteria posed by these expressions, as we will see later.

$$path\_sel\_\exp = \{Root,,,r1.r2\} \quad (6)$$

2) Path selection expression with single level Restriction: For the path selection expression in equation 7 the agent will start searching the sub-tree defined from the starting point (Root) for services in level 2 to which you can reach following relationships first of type r1 and then r2 . The selected services are highlighted in Fig. 4

$$path\_sel\_\exp = \{Root, 2, 2, r1.r2\} \tag{7}$$

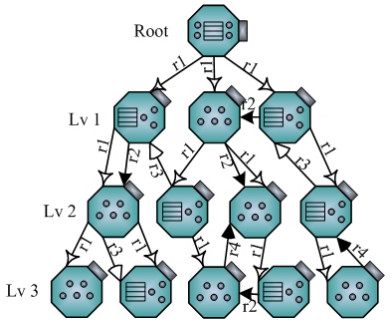

Fig. 2. General relationship tree

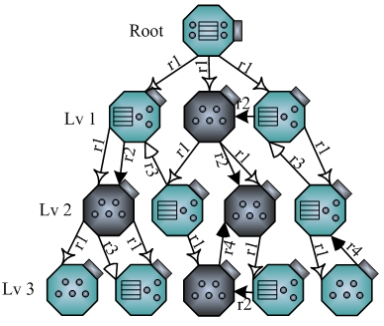

Fig. 3. Service selection no Restriction

3) Path selection expression with multiple level Restriction: In the case where the path selection expression has multi-level restriction, as in equation 8, the agent will search the sub-tree defined from the starting point (Root) for services in level 2 and 3 which can be reached by first following relationships of type r1 and then r2. The selected services are highlighted in Fig. 5

$$path\_sel\_\exp = \{Root, 2, 3, r1.r2\} \tag{8}$$

4) Fringe Services: In all the above service selection examples the agent visits one after the other all services included in the sub-tree whose head node is the start node tag. For every selection the agent makes, it evaluates for every service node whether each pattern tag in the relation pattern can be followed or not, thus there is a recursive evaluation of the binary state of each pattern tag in the relation pattern. The recursive evaluation of each pattern tag on a sequence of pattern tags can allow detection of services where the relation pattern cannot be followed. These services are called

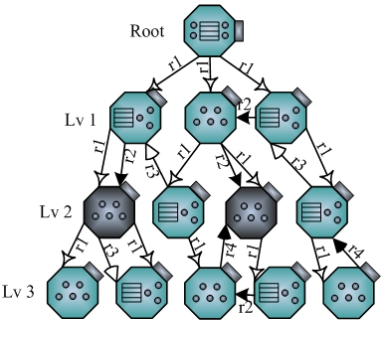

Fig. 4. Service selection single restriction

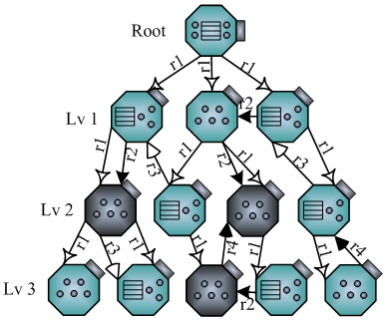

Fig. 5. Service selection multiple restriction

fringe services. An example of a path selection expression that captures services where type r1 relationships cannot be followed is given in equation 9. The services that are selected are highlighted in Fig. 6

$$path\_sel\_\exp = \{Root,,,(r1)!\} \tag{9}$$

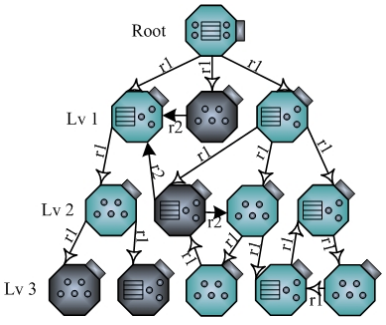

Fig. 6. Service selection for fringe Services

3.4 Service Association

In order to use the selection scheme described previously, a method to define relations between services is required. A simple scheme to define such relationships if containment was the only option would be to use a simple naming scheme to define the endpoint URIs where services are deployed. Such a scheme would be to use the number of slashes "/" in the endpoint URI to denote the level in a hierarchy where a service is offered, the tag after the last slash to denote the name of the service and the tag before it to denote its parent. However, relationships between data may not only be containment. The definition of other relations must also be possible, so that the definition of actions between data and conversation scenarios between services is feasible.

One way to define relationships between services is to provide metadata about them. Initially we considered certain WS standards for the job. WS-Addressing [14] and WS-MetadataExchange (WS-MEX) [15] are such standards. WS-MEX specifies the messages that applications exchange in order to retrieve service metadata. WS-MEX is intended though as a retrieval mechanism for only service description data. Because introduction of metadata services will increase unnecessarily latency and memory requirements WS-Addressing was considered as another solution. WS-Addressing was initially designed to provide service endpoints with capabilities such as multi protocol and interface information support. This is achieved by adding service endpoint references (ER) to endpoint's Uniform Resource Identifiers (URIs). ERs are adding information to WS messages so that applications can support various exchange patterns than simple request responses. WS-addressing allows notification standards to provide asynchronous event reporting and subscription. It can be used though in a different way. WS-Addressing could be used to add information about service relationships. The addition of a metadata element in the standard to provide a consuming application with WSDL information also allows the provision of other metadata about a service. Still work on WS-Addressing is not finalized, while the

standard and the metadata element is not supported by many open source toolkits. Thus, we had to find other means to support metadata information about service relationships until work in WS-addressing is finalized and open source toolkits support it.

Providing metadata about service relationships requires a simple and flexible scheme. In WS, WSDL allows to define the interface that services expose and the access point where they are offered. A WSDL document consists of an abstract part acting as an access stub and a concrete part affecting its behavior. The interface tag of the abstract part describes sequences of messages a service sends and/or receives. It does this by grouping related messages into operations. An operation is a sequence of input and output messages, and an interface is a set of operations [4]. The endpoint component of the concrete part defines the particulars of a specific endpoint where a service is accessed [4]. A service consists of its instance and its endpoint and the later is as important as the former. Most service properties and settings reside there.

The organization of WSDL and the structure it enforces on its constituent parts allows us to do three things. First, it allows manipulation of the level of transparency [12]. Secondly, the granularity of services can be altered [12]. Third it permits three distinct ways to deploy services. The most common way of deployment is by allowing access to an entire object through an interface. Service WS0 in Fig. 7 shows this deployment scenario. The WSDL document in this case contains one service tag referring to one binding and one endpoint tag. The second deployment scenario seen in Fig. 7 is for service WS1 where access to a service is through multiple access points. In this case the WSDL document for this service contains one service tag which includes multiple endpoint tags (two in this case) referring to the same binding tag. The third deployment scenario is seen for services WS2 and WS3. In this scenario two interfaces to the same object are offered by defining multiple endpoint elements for the same service element, which refer to different binding elements.

For our association scheme use of the second deployment scheme was made. The multiple access points of this scheme provide us with the means to define metadata about service relationships. In our proposal, every service has a primary access point to provide access to it. For every relationship a service has with another service, the latter will define a secondary URI. This secondary URI provides metadata about the relationship that the two services share with a syntax that complies with RFC 3986 [13] about URIs. The syntax for the primary and the secondary URIs is given in (10) and (11). Parsing secondary URIs provides agents with a view of the association tree.

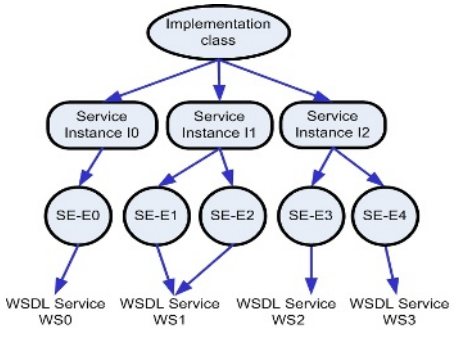

Fig. 7. Service deployment scenarios

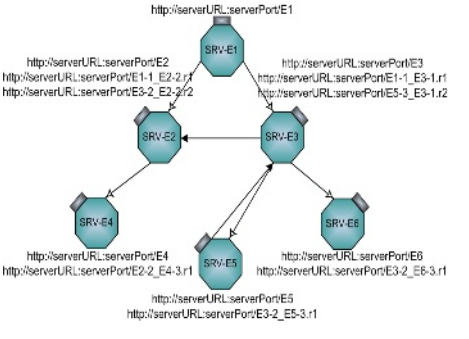

Fig. 8. Association scenario with endpoints

In Fig. 8 primary and secondary URIs for all services sharing two types of relations are provided. Service ST-E1 has only one URI to allow access to it. Services ST-E2 and ST-E3 have 3 URIs, the primary and two secondary ones for both the r1 and r2 types of relations they share with other services. All third level services contain one primary one secondary URI to show an association of type r1 with other services.

Primary_URI =http://serverURL:serverPort/primaryServiceTag (10)

Secondary_URI=http://serverURL:serverPort/sendingServiceTag-serviceLevel_recipientServiceTag-serviceLevel.relationTag (11)

4 Selective Retrieval at Service Level

So far we have explained how bulk data retrieval is possible by providing a collective view upon data accessible by services at the agent level through service association and at the service level through specific methods that have collective access to the data. Path selection expressions allow our agent to deploy a selective capability on the services modeling the management information. To add filtering as selective retrieval capability on the management data within a service, our parser also evaluates data selection expressions sent to it by the manager. These expressions mandate which information from the service should be selected by the agent.

4.1 Data Selection Expressions

SNMP contains two types of data, single and multiple instance (tabular) ones. The BNF syntax for the data selection expression for single instance data is the following:

<sgl_slct_exp>::={<identifier > | <sgl_slct_exp >, <identifier> } . (12)

An example expression for retrieving the ipIndelivers, tcpOutSegs and the tcpInSegs from the TCP and IP MIB would be the following:

sgl_slct_exp={tcpInSegs,tcpOutSegs,ipInDelivers} . (13)

Retrieving multiple instance entities is a bit more complex since it requires an expression to define which entities need be retrieved and a filtering expression to retrieve only part of the data that meet specific criteria. The BNF syntax for the multi-instance selection expression and the filter expression is the following:

<mult_slct_exp>::={<mult_inst_tag> | <mult_slct_exp>, <mult_inst_tag >} . (14)

<mult_inst_tag>::=<identifier>([] | [<integer>-<integer>] | [< integer>] | [< integer>(< | >)><letter>(> |<)< integer>]) . (15)

<flt_exp>::={<mult_inst_tag ><relational operator> <value>| <flt_exp > <space><logical_operator><space> <flt_exp>} (16)

<value>::=<integer>|<string> (17)

A usage example for retrieving all TCP connections is given in (18).

ml_slct_exp={tcpConnEntry[ ]} (18)

The filter expression for retrieving only FTP and HTTP connections is given in (19).

flt_exp= {tcpConnLocalPort = 22 OR tcpConnLocalPort =80} (19)

5 Usage Scenario

To demonstrate a case where fairly complex network management data must be retrieved, we present a use case scenario. In the configuration of Fig. 9, consisting of five IP routers and 6 Local Area Networks (LAN), LAN N2 receives substantial traffic from an HTTP server it hosts. Both N2 and R1 are able to cope with this traffic. At some point though, a user in LAN4 creates more traffic by transferring large files from an FTP server in LAN3. As a result R1 and LAN2 exceed their handling capacity. The cause of congestion in router R1 must be detected by the Central Management System (CMS) responsible for managing the overall network.

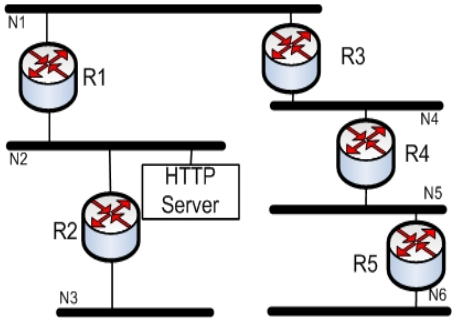

Fig. 9. LAN configuration

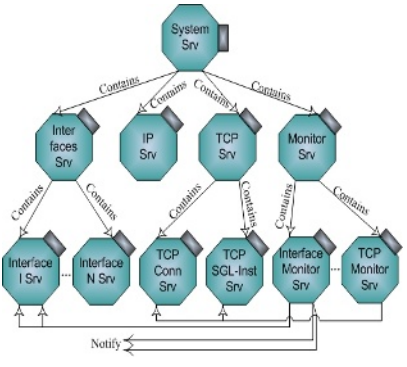

Fig. 10. Service association for usage scenario

Assuming the tree of services in Fig. 10 deployed in every host and router on the network, the CMS should be notified when congestion builds up and take appropriate actions. On notification the CMS must determine the cause of congestion and take actions to alleviate it. In Fig. 10, services IP, TCP, and Interfaces model the corresponding SNMP MIB groups. Other MIBs may also be captured as services but they are omitted. The TCP Service contains two services that model TCP single instance data and TCP connections. The Interfaces service breaks into a number of services representing the interface MIB data of every interface of the managed device. In Fig. 10, other generic services such as an event service or a logging service might also be present. To keep things simple we have omitted a WS-based notification service and assume that notifications to the CMS are sent from the services that produce them. In the future we will investigate creation of a flexible and scalable WS-based notification service. At the moment when the traffic congestion threshold is crossed on interface N2 the relevant interface monitoring service notifies the CMS; we

assume here that the CMS has activated the interface monitoring service in R1 Upon receiving this notification, the CMS must try to determine the cause of congestion.

The CMS should retrieve information about the load that TCP connections between the various subnetworks are imposing on the interface N2 of router R1. Such information is not provided by SNMP MIBs. Such an option would be available, if MIBs would be capable to capture incoming and outgoing traffic per TCP connection or use the RMON MIB to capture link layer traffic. In our case, let's assume that the TCP MIB is equipped with such data in the TCP connection table under two variables called tcpConnInSegs and tcpConnOutSegs. In this case, the CMS should start monitoring hosts with Web or FTP servers and try to identify TCP connections with high segment counters and also high segment rates.

Path_slct_exp={TCP Monitor Srv,NULL,NULL,empty} (20)

flt_exp={ (tcpConnLocalPort = 22 OR tcpConnLocalPort =80) AND
(tcpConnInSegs>thres_value OR tcpConnOutSegs>thres_value) (21)

For the CMS to inform the agent which data it requires to retrieve from the TCP monitoring service, it should dispatch to the agent the expressions in (20) and (21). On receiving these two expressions, the agent picks up from the TCP monitoring service only TCP connections for applications on well known ports, usually known to produce traffic. Such applications are FTP and HTTP on ports 22 and 80 respectively. In addition, the filter expression tells the agent to only retrieve connections whose incoming or outgoing traffic exceeds a certain threshold. This way the manager will acquire a few possible candidates responsible for creating congestion. With further monitoring on these connections, it can determine the behavior of each one in terms of segment rate and possibly identify remote hosts through tcpConnRemAddress that cause the extra traffic. Without the functionality we support, the whole of the remote TCP connection table would need to be retrieved, and this would incur a lot of additional traffic to the already congested network. This is a relatively simple scenario but other scenarios also exist where a lot more information that belongs to different services must be selectively retrieved. One such case may be the selective retrieval of data from the logging service in order to trace particular series of events.

6 Conclusion

This paper presents an approach to address efficient information retrieval using Web Services. Viewing management information collectively at various levels of the management hierarchy addresses the problem of bulk retrieval. Using a parser to interpret expressions that highlight only specific data to be retrieved solves the problem of selective retrieval. The collective and selective capability we provide in our approach allows manipulation of management information at a fine level of granularity.

In order to support collective view of data at the object level, we deployed methods that view SNMP single instance or multiple instance data as a whole. To do the same at the service level we developed a scheme that defines arbitrary relationships between services. In order to allow the navigation and selection of services based on any relationship we developed a parser that interprets appropriate expressions. In order to

perform selective retrieval at the object level, the same parser interprets another series expressions to highlight the data that will be selected.

We have developed both the parser which performs the path and data selection expression interpretation and the service association usage scenario to test its applicability. This was all done in Java 1.4.2.10 using the regex machine it provides. We used two WS toolkits to deploy these services, Apache AXIS and WASP, and we also plan to provide performance data in the future. Until now we have only used custom-made events that are dispatched directly from the services that produce them to the same "hardwired" manager. We will be investigating a proper notification service in the near future, tracking also work that has taken place in relevant standards bodies.

Acknowledgements

The research work in this paper was partly supported by the EU EMANICS Network of Excellence on the Management of Next Generation Networks (IST-026854).

References

1. J. Schönwälder, "Overview of the 2002 IAB Network Management Workshop," RFC 3535.
2. L. E. Menten, Bell Laboratories, "Experiences in the application of XML for Device Management," *IEEE Communications Magazine,* Vol. 42, no. 7, pp. 92-100 July 2004.
3. W3C, "The Simple Object Access Protocol 1.2 (SOAP)", http://www.w3.org/TR/soap12-part1,h ttp://www.w3.org/TR/soap12-part2
4. W3C,"The Web Services Description Language 1.1 (WSDL)", http://www.w3.org/ TR/ wsdl/
5. OASIS, "The Universal Discovery Description and Integration Technical Committee Draft V 3.02 (UDDI)," http://uddi.org/pubs/uddi_v3.html.
6. A. Pras, et al, "Comparing the Performance of SNMP and WS-Based Management," *IEEE Electronic Transaction on Network and Service Management,* eTNSM, FALL 2004.
7. G. Pavlou, P. Flegkas, and S. Gouveris, "On Management Technologies and the Potential of Web Services," *IEEE Communications Magazine,* Vol. 42, no. 7, pp. 58-66 July 2004.
8. D. Davis, M. Parashar, "Latency Performance of SOAP Implementations," 2nd *IEEE ACM International Symposium on Cluster Computing and the Grid,* 2002, p 407.
9. R. van Engelen, "Code Generation Techniques for Developing Light-Weight XML WS for embedded devices," ACM symposium on applied computing, 2004, pp.854-861
10. G. Pavlou et al, "CMIS/P++: extensions to CMIS/P for increased expressiveness and efficiency in the manipulation of management information.," 7th Annual joint conference of the IEEE Computer and Comms Societies, INFOCOM 98, Vol 2, April 1998 ,pp.430 - 438
11. P. Naur, "Revised Report on the Algorithmic Language of ALGOL 60," Communications of the ACM, May 1960, pp. 299-314.
12. J. Schonwalder, A. Pras, and J. P. Martin-Flatin, "On the Future of Internet Management Technologies," *IEEE Communications Magazine.*, Oct. 2003, pp. 90–97.
13. T. Berners-Lee, R. Fielding, L. Masinter, "The Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.
14. D. Box, F. Curbera, "WS-Addressing specification" , W3C submission 10 August 2004.
15. F. Curbera, J. Schlimmer, "WS-Metadata Exchange specification", Sept. 2004.