A Web Services Approach to Address Efficient Information Retrieval in Network Management

Aimilios Chourmouziadis, George Pavlou
University of Surrey

Abstract: Management research and standardization has undergone a lot of development during the last 20 years. A lot of attempts have been made towards a technology to address management problems, including protocol based approaches such as SNMP and CMIP or object-based approaches such as CORBA. The quest though for a general enough technology to be used for network, system and service management that is also efficient in terms of information retrieval, computational resources required and development/operational costs still rages. A recent emerging technology is Web Services (WS) and initial research has shown that it can be used relatively efficiently for distributed network management. This paper aims at presenting the means to organize services/objects into hierarchical structures to facilitate bulk information retrieval by a manager system and to also propose a solution for selective retrieval based on preset criteria in order to reduce traffic cost.

1. Introduction.

During the last 20 years, from when SNMP version one was introduced and even until SNMP version three made its appearance a few years back with, 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 has too many inefficiencies that limit its usefulness to only being a monitoring tool. Therefore alternative technologies must be investigated in an effort to design and implement a management framework that is efficient in terms of information retrieval time and traffic, computational resources required and development & operational costs. CORBA was considered in the mid 1990's as a unifying management technology and although it has come a long way since then, it still has some serious inefficiencies. More recently, the introduction of Web Services coupled with the advent of maturing XML technology standards is seen as a promising approach for faster product development, tighter system integration and robust device management [2].

A general technology for network management should reduce the cost and complexity of maintaining the management infrastructure, it should be easily extensible making versioning support easy, but should also be powerful and expressive, allowing flexible queries to be performed and also multipart transactions to commit atomically. A single transaction should allow intermixing of operations that involve changes in configuration state and the retrieval of dynamic/static state information. The key issues thus are:

- Efficient bulk and selective information retrieval, similar in power to CMIP scoping and filtering.
- Scalable and flexible information models for efficient and fast software development
- Powerful and scalable event management.
- Transaction support for configuration management.
- Security, including authentication/authorization and access control.

XML, Web Services and the large collection of tools surrounding these technologies offer to some extent a great opportunity to solve these problems [4]. Web services have many similarities to CORBA [3]. WSDL [6], URIs, SOAP [5], UDDI [7] have all equivalent parts in CORBA, verifying the fact that Web Services can be used as a distributed object technology in the context of network and service management [3]. Of course CORBA and WS also have differences but the key issue is that WS can be used for distributed management.

2. Related work and Scope of this Work

Several researchers in the literature tried to compare the performance of different management technologies to WS. In [3] and [8] two different WS toolkits are used, and the authors conclude that WS is efficient only when transferring large volumes of data. In [9] and [14] two gateway schemes on XML and WS respectively were implemented for backwards compatibility. WS performance though at this stage yields ambiguous results, since as shown in [10], [11], [12] and [13] performance can be really poor as schema-specific parsers are at an early stage,

transport protocols issues have not been very well looked at, and other factors like compression and data serialization have not yet been taken into account.

This work though does not investigate WS performance but has as scope efficient bulk and selective information retrieval, similar to CMIP scoping and filtering. SNMP's limited support for bulk retrieval and its inexistent selective retrieval capabilities and CORBA's lack of explicit relevant features leave CMIP as the only powerful technology for information retrieval but at the cost of complexity and the fact that OSI-based technologies are slowly phased-out in favour of IP-based ones. This work tries to provide solutions to these problems.

3. Association of WS interfaces to support scoping

Scoping in order to be applied to any network management technology necessitates the existence of a way to organize managed objects (MOs) into well-defined structures. In CORBA, object interface association was the idea that researchers came up with to make this possible. MOs at a lower level of the object hierarchy are controlled by MOs at levels before them. This approach, which is similar to the CMIP one although the only type of association in CMIP is containment, is necessary to organise objects in a "searchable" structure.

Our idea to perform scoping relies on a scheme of associating WSDL interfaces/objects exploiting WS characteristics. In order for this to be done though it is necessary to view some of these characteristics. WS consist mainly of two components the web service implementation held within the service instance and the web services endpoints from which WS are accessed. Service Instances consist of an abstract part acting as an access stub, and from a concrete part which affects instance behaviour as can be seen in ***Figure 1***. Most of the service properties and settings are held at the endpoint. If a server's URL is http://localhost:port/ then for each service at ***Figure 2*** the access point is http://localhost:port/Ei. Usually a SI is tied to a single implementation class, and thus endpoints on the same instance share the same data. Two different endpoints though like E_3 and E_4 can offer two different services or, as in the case of E1 and E2, provide a different access point for the same service. Our idea exploits these two cases.

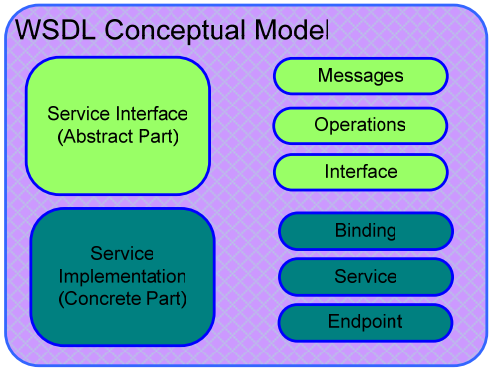

Figure 1
The WSDL conceptual Model

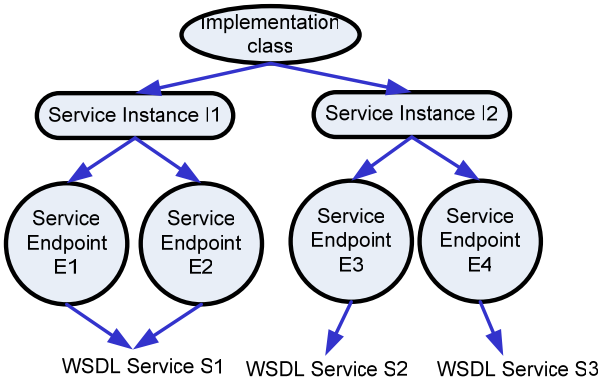

Figure 2
Instances VS Endpoints

We exploit the containment hierarchy of endpoints to conceptually deploy MOs at different levels of the hierarchy. This means that the lowest level of services are being deployed first at the last level of the endpoint hierarchy, with each level specified by a "/"and then the next level of services above it are deployed by discovering the services directly underneath them; this continues until the highest level of services are deployed. So if a highest level service was deployed at *"/xxxxx"*, the one under it would be at *" /xxxxx/yyyyy"*, but the lower level service would be deployed first and then the highest level one. Services at higher levels though, should discover only services which lie one level deeper from their endpoint. In our case three SNMP MIB groups have been implemented TCP, IP and ICMP, each deployed at a different endpoint. Since TCP and IP include tables, a Table Service has been implemented that associates them. For ICMP a NonTable Service has been implemented to play the same role. ***Figure 3*** shows the Registry after the deployment of all services. Initially the TCP, IP and the ICMP services are deployed at the lowest level and then the federating services are deployed using Java's reflection API to discover dynamically the services underneath them in order to avoid the need to update higher level services when new services are added at lower levels.

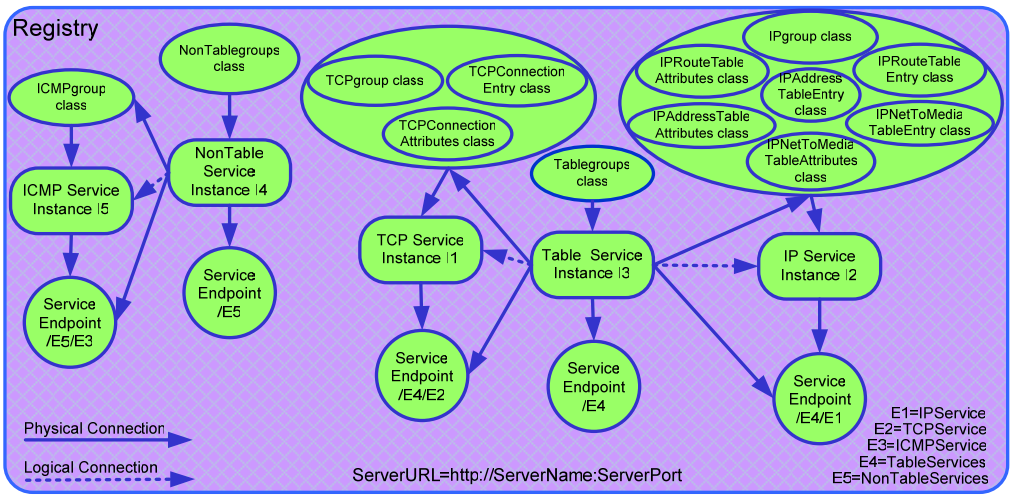

Figure 3
Registry's view after the Deployment of Services

Having organized services in this way all that a client needs to do to perform scoping is to choose the endpoint at which a service is deployed. Conversation with the lower level services is handled by the service selected. In our case, conversation is performed only between methods that have functions that perform filtering. Three common methods amongst all objects which allow filtering as well exist. These functions are:
String [] getSglInstObj(String expression) #returns part or all the single-instance values of the object
String [] getTable(String expression) #returns parts or all the tables of the object
String [] getObj(String expression) #returns parts or all the tables and single instance values of the object
Apart from these methods the objects may offer specific methods to retrieve specific single instance values and methods to retrieve a specific column of a specific table or the rows of a specific table.

Problems exist though when a hierarchical model of organizing WSDL interfaces/objects allows traversal of containment relationships by organizing the endpoints at which services are offered. Sometimes it is necessary to traverse objects in an arbitrary way and not just on containment relationships. To meet the need of arbitrary traversal all that a WSDL Service has to do is offer extra different access points that allow objects at a different level of the endpoint hierarchy to access them as if they are at the next level of the hierarchy. Thus a service at *"/xxxxx"* can be accessed by a service at *"/xxxxx/yyyyy"* through an access point at *"/xxxx/yyyyyqqqqq"*. This way access points act as pointers allowing definition of *contains* and *containedIn* relationships, thus allowing traversal of objects in an arbitrary manner. Contains and containedIn relationships though raise some issues. In the previous example the service at *"/xxxxx/yyyy/qqqqq"* may not need to access the service *"/xxxxx"* except in specific cases. A way to control this behaviour is by forcing endpoints subparts *"/xxxxx"*, *"/yyyyy"*, and *"/qqqq"* to be unique. Uniqueness of names can by forced by checking endpoint subparts before deploying new services and by using the server's URL to form unique endpoint references across differrent administrative domains.The benefits of such a decision are two. First of all, services on different implementations cannot be mistakenly acknowledged as different services from the same implementation. The second benefit in the previous example comes when we force the access point at *"/xxxxx/yyyy/qqqqq"* *qqqqq* to be xxxxx. This way the lower level service can recognise that the service at *"/xxxxx/yyyy/qqqqq"* is a higher level service by just checking its endpoint structure, and by also providing to it knowledge of the client's service endpoint selection, it can decide on its course of action . Another issue exists as well. A client's selection of service determines the starting point of scoping but not a stopping point. If a stopping point is necessary, the manager must specify this as the number of relationships to be traversed.

4. Filtering for WS Management

Selective retrieval or filtering is yet another issue that must be solved and this can be done in two ways:

- Filtering at the Simple Object Access Protocol (SOAP) level through XPath or through some other approach.
- Filtering at the interface/object level through a custom solution

Filtering at the SOAP level though creates problems. In order for the SOAP body to be available, an expensive search to more than the necessary values for filtering must be made, which is expensive in terms of CPU load and

memory as well as latency. If XPath in its full form is used to perform filtering, this is also expensive. Even a cut down version of XPath may not be appropriate. Thus a custom solution at the object level through filtering expressions dispatched from the Manager to the agent when service calls to specific methods are made has been designed and implemented. These string expressions are parsed at the agent side by a custom parser which understands the tokens in the following table:

Expression	Description
elementToken	The name of a column or a row of a table or of a single instance entity that matches this string
valueToken	By default equals to value pointing the column or the row of a table whose value is …, but if a row is selected it can either be value or take the name of the elementToken of the column
/$elementToken	Select the single instance entity of whose name is equal to elementToken
/$elementToken[]	Select all the members of a column or the rows in a table with such a column/row name
/$elementToken[i]	Select member i of a column or rows of a table with such a column/row name
/$elementToken[i-j]	Select members i to j of a column or rows of a table with such a column/row name
/$elementToken[k<i<j]	The same as the previous one
/@valueToken	Selects the column whose value is… or the row whose value is… or whose column value is…
!=,=,<=,>=,>,<	The operator that follows every value token
and/or	Perform anding or oring of valueToken expressions
+	Connect elementToken of single instance entities if more than one is needed to be retrieved
&&	Connect expressions of single instance entities or table expressions or a combination of them
( )	Alter the precedence of valueToken expressions with parentheses

Table 1

Parser syntax tokens

For example, if a manager system wants for example to retrieve all the TCP connections the expression is:

expression=/$ tcpConnectionTableEntry [ ]

In the case where parameters like tcpRtoMax and tcpMaxConn and tcpCurrEstab are needed the expression is:

expression=/$ tcpRtoMax + tcpMaxConn + tcpCurrEstab

In the case where both single instance values like tcpRtoMax and tcpMaxConn and tcpCurrEstab and all the TCP connections are needed the expression is:

expression=/$ tcpConnectionTableEntry [ ] && /$ tcpRtoMax + tcpMaxConn + tcpCurrEstab

If all TCP Connections in established mode whose local addresses are greater than 100.100.100.100 and whose remote addresses are smaller than 150.150.150.150 are needed for retrieval the expression is:

expression=/$ tcpConnectionTableEntry [ ] /@ tcpConnLocalAddress>=100.100.100.100 and tcpConnRemAddress<=150.150.150.150 and tcpConnState=1

5. Summary and Conclusions

Organizing WSDL objects into a hierarchical structure according to their endpoint definition and deployment allows us to deal with two serious problems in object-based network management. The first one is extensibility of the management solution through ways that are both practical and economical. The second benefit of structure is scoping capabilities. By exploiting the fact that a service with the same implementation can have different access points and that endpoints can denote containment relationships in a pre-described manner, one can organize WSDL objects into containment hierarchies that can be traversed in an arbitrary way. Scoping allows the manager to pick up only the object values that are necessary or to perform filtering operations on them to retrieve partial information in order to reduce the management traffic in comparison with retrieving all the information. So it remains to be seen whether interface association and filtering will increase latency and management traffic due to an increase of remote calls and see whether the increased extensibility as well the ability to retrieve specific data in a bulk and selective manner through scoping and filtering can outweigh these shortcomings.

6. References

[1] J.Schonwalder, “Overview of the 2002 IAB Network Management Workshop”, May 2003, IETF RFC3535
[2]Lawrence E. Menten, Bell Laboratories, “Experiences in the Application of XML for Device Management”, XML Based Network Management in IEEE communications Magazine, July 2004.

[3]George Pavlou, Paris Flegkas, Stelios Gouveris, and Antonio Liotta, “On Management Technologies and the Potential of Web Services” University of Surrey, XML Based Network Management in IEEE communications Magazine, July 2004.
[4] Jean Philippe Martin-Flatin, “Web Based Management of IP Networks and Systems”, Wiley Series in Communications Networking and Distributed Systems, Copyright 2003
[5] Don Box , David Ehnebuske, Gopal Kakivaya, Andrew Layman , Noah Mendelsohn , Henrik Frystyk Nielsen ,Satish Thatte , Dave Winer, Simple Object Access Protocol (SOAP) 1.1 ,W3C Note 08 May 2000
[6]Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, “Web Services Description on Language (WSDL) 1.1”,W3C Note 15 March 2001
[7]Douglas Bryan, Vadim Draluk, Dave Ehnebuske,Tom Glover, Andrew Hately, Yin Leng Husband, Alan Karp, Keisuke Kibakura, Chris Kurt, Jeff Lancelle, Verisign, Sam Lee, Sean MacRoibeaird, Anne Thomas Manes, Barbara McKee, Joel Munter, Tammy Nordan, Chuck Reeves,Dan Rogers, Christine Tomlinson, Cafer Tosun, Claus von Riegen, Prasad Yendluri, “UDDI Version 2.04 API Specification “UDDI Committee Specification, 19 July 2002
[8]Aiko Pras, Thomas Drevers, Remco van de Meent, Dick Quartel, Comparing the performance of SNMP and Web Services Based Management, E transactions on network and service management, fall 2004
[9]Ricardo Neisse, Lemos Vianna, Lisandro Zabenedetti Granville, Maria janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco, “Implementation and Bandwidth Consumption Evaluation of SNMP to Web Services Gateways”, Institute of Informatics, Federal University of Rio Grande do Sul, , 2004 IEEE, DSOM
[10]Dan Davis and Manish Parashar , “Latency Performance of SOAP Implementations”, Compaq Computer Corporation, Manalapan, NJ 07726, USA, Department of Electrical and Computer Engineering, Center for Advanced Information Processing (CAIP) Rutgers, The State University of New Jersey, Piscataway, NJ 08855-1390, USA,{dand,parasharg@caip.rutgers.edu}
[11]Alex Ng, Shiping Chen, Paul Greenfield2, “An Evaluation of Contemporary Commercial SOAP Implementations”, Department of Computing, Macquarie University, North Ryde, NSW 2109, Australia, CSIRO ICT Centre, PO Box 17, North Ryde, NSW 1670, Australia, alexng@ics.mq.edu.au, shiping.chen@csiro.au, paul.greenfield@csiro.au
[12]Kenneth Chiu, Madhusudhan Govindaraju, Randall Bramley, “Investigating the Limits of SOAP Performance for Scientific Computing”, Department of Computer Science, Bloomington, Indiana University, chiuk@cs.indiana.edu, mgovinda@cs.indiana.edu, bramley@cs.indiana.edu
[13]Robert van Engelen, “Code Generation Techniques for Developing Light-Weight XML web services for embedded devices” Department of Computer Science, Florida State University, Tallahassee, FL 323064530, engelen@cs.fsu.edu
[14]Mi-Jung Choi, James W. Hong, Hong-Taek Ju, “XML-Based Network Management of IP Networks” ETRI journal, Volume 25, Number 6, December 2003