APPLICATION SCENARIOS FOR DISTRIBUTED MANAGEMENT USING SNMP EXPRESSIONS

Rui PedroLopes
Escola Superior de Tecnologia e de Gestão, Instituto Politécnico de Bragança, Bragança, Portugal
Email: rlopes@ipb.pt

José Luís Oliveira
Departamento de Electrónica e de Telecomunicações, Universidade de Aveiro, Aveiro, Portugal
Email: jlo@det.ua.pt

Keywords: DISMAN, Expression MIB, Network Management, SNMP

Abstract: Management distribution is an, we can say, old topic in terms of the number of proposed solutions and publications. Recently, the DISMAN workgroup suggested a set of MIB modules to address this matter in the context of SNMP. One of the DISMAN modules has the capability of using expressions to perform decentralized processing of management information – the Expression MIB. Although existing for some time now, its capabilities are not very well known. In fact, other DISMAN MIBs, such as the Schedule MIB and the Script MIB already got some attention in several papers and are target of very solid work. There are hardly any papers describing the Expression MIB and its functionality. This paper contributes to eliminate this absence by describing our implementation effort around it as well as some real world applications for it.

1 INTRODUCTION

During the last years the SNMP management framework has strongly guided the development of network systems and management applications. This architecture, regardless of some well-known shortcomings, has managed not only to survive but also to evolve to a rather complete set of features. This fact, combined with its inherent simplicity and coarse APIs availability, has pushed it into a dominant position in today's network management market.

On the other hand, one of the problems associated with SNMP is its centralized architecture, not well suited for offline operation and not scalable on large networks. The solution for this problem is, according to many authors, management distribution, a research topic since early 90's.

The history of management distribution, well discussed by Martin-Flatin on (Martin-Flatin, 1998), started with initial work by Yemini *et al.* in 1991 when features such as scalability, flexibility and robustness where identified as necessary for future developments on network management (Yemini, 1991). Goldszmidt and Yemini early supported a management distribution methodology by delegating management operations near management information (Goldszmidt, 1998). According to this concept, management processing functions are dynamically delegated to the network elements and executed locally. This introduces a shift in the original concept where the information is transported to a central location to be processed. This approach is known as Management by Delegation (MbD) and although the research prototypes did not have the expected community recognition they unquestionably proved the concept.

Other approaches for management distributions suggested using mobile agents to implement and distribute management functions. Many authors supported several usage scenarios, platforms and applications and enforced the concept of a cooperative management effort on the network (Bieszczad, 1997, Pham, 1998, Krause, 1996, Lopes, 1999).

The industry also adopted management distribution by releasing tools, APIs or agents, such as Sun's JMX (Sun Microsystems) or SNMP Research's CIAgent (SNMP Research).

Some of these products, technology and concepts do not easily survive the community resistance

because they are not either compatible or adapted to the management technology of choice – the SNMP. The SNMP community have also suggested, under the DISMAN workgroup of the IETF (DISMAN Charter), some tools for management distribution.

2 DISTRIBUTED SNMP-BASED MANAGEMENT

The typical usage scenario of the DISMAN architecture is based on the distribution of management tasks through a set of mid-level managers known as Distributed Managers (DMs). The main purpose of this approach is to reduce the command exchange with the management station, alleviate the processing load usually residing at a single central point and increase the system robustness by introducing redundancy and by allowing offline operation.

A fundamental piece of management by delegation is the possibility of running programs or scripts remotely, approach implemented by the Script MIB. Schoenwaelder, following an excellent study of distribution models and solutions, presents the distribution of management tasks in the context of the Script MIB (Schoenwaelder, 1997).

Associated with the Script MIB, the IETF Distributed Management charter suggests other modules, namely the Schedule, Expression, Event, Remote Operations, Notification Log, Alarm and Alarm Reporting Control MIB modules (DISMAN Charter). These set of MIB modules provide a rather complete framework for distributing management operations under the SNMP context over a hierarchy of several DMs.

One of the best knew MIB, probably because the early availability of implementations is the Script MIB. In this paper, we describe our ongoing work on another DISMAN MIB module: the Expression MIB.

2.1 Expression MIB

We started working on the Expression MIB implementation when the documentation was still at the Internet draft status. It made the first appearance on (Lopes, 2000) and since then no other implementation has become known to the authors. Meanwhile, some minor details have changed both in the IETF documentation and in the implementation code, particularly the expression parser and the sampling mechanism. We included a more robust expression parser and changed some functions according to the clarifications made by the documentation editors. We also enhanced the sampling mechanism with remote sampling to cope with the Event MIB requirements. This last change was valuable for the work presented on this paper.

The MIB is divided in three main groups:

- *expResource* – this group is related to resource control, with particular incidence on sampling parameters since this operation can have some impact on system resources.
- *expDefine* – is organized in three tables which collect information about the expression definition and about the errors occurred while evaluating it: a) *expExpressionTable*, defines the expression string, the result type as well as the sampling period. b) *expErrorTable* maintains a table of errors' registers gathering information such as: the last time an error occurred on evaluating the expression, the operation in which it occurred, the error type. c) *expObjectTable* controls each element characteristics inside the expression. The expression string may contain variables and each variable may have different sampling types and be or not wildcarded.
- *expValue* – this group has a single table which instantiates the evaluation objects. It is by querying this table that the result from the expression is known.

The values used in the expressions may be absolute (the values of the MIB objects at the sampling time), delta (the difference from one sample value to the next) or changed (a boolean indicating whether or not the object changed its value since the last sample). In addition to sampling, the MIB also defines wildcarding, allowing the use of a single expression over multiple instances of the same MIB object. While regular objects are resolved by a SNMP get operation, wildcard objects are retrieved through a get-next operation. Users are familiar with wildcarding for referencing multiple files (such as the UNIX command "`cp foo.* /tmp`"). If there is more than one wildcard parameter in an expression they all must have the same OID termination (semantics) to obtain a coherent result.

An expression result is retrieved by querying a row in the *expValueTable*. Each row has a single column, formatted according to the result type of the expression. The value is accessed by an OID containing the OID for the data type, the expression name and a fragment.

The expression name has the form of x."owner".y."name" converted to dot separated integers. The integer x is the length of the owner and y is the length of the string which identifies this expression to the particular owner. Each word character is converted to integer and separated from the other integers by a dot.

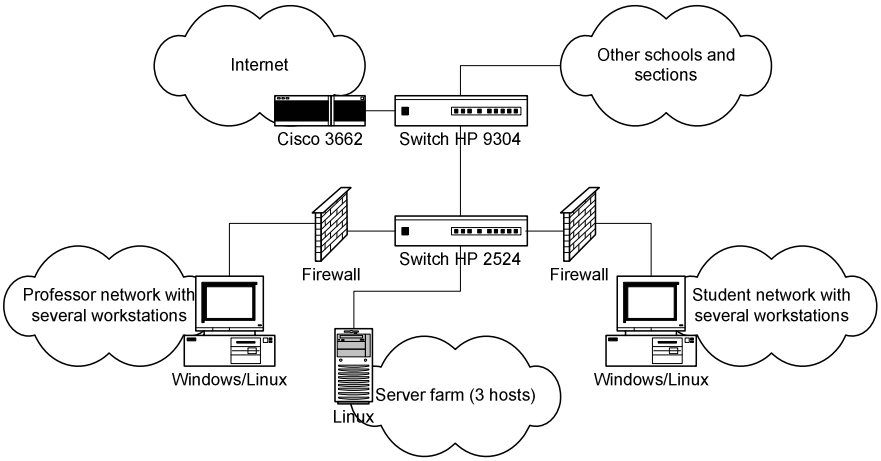

Figure 1: Sections of the local network scenario.

The fragment starts with "0.0." and ends with a zero, if there is no wildcard or, otherwise, with the instance that satisfied the wildcard.

3 MANAGEMENT TASKS IN A REAL SCENARIO

To better understand the effects of using the Expression MIB in every day management operations, wee used some elements and hosts of the network installed in our workplace (Figure 1).

As a teaching institute, it has several workstations specific for student work as well as several hosts to provide professors with individual working stations, both protected with different firewalls. In this scenario, we picked up both switches, ten workstations from the student section and three servers. The tasks we wanted to achieve where:

Task 1 - Rate of lost forwarding database entries

This task applies to the switches and calculates the rate of the total number of forwarding database entries, which have been or would have been learnt, but have been discarded due to a lack of space to store them. This information is stored in the managed object *dot1dTpLearnedEntryDiscards*. If this counter is increasing, it indicates that the forwarding database is regularly becoming full, which may result in unpleasant performance effects on the network (Expression 1). The object is defined in the Bridge MIB (Decker, 1993).

$$\text{lost entries rate} = \frac{\Delta \text{dot1dTpLearnedEntryDiscards}}{\text{number of seconds in } \Delta}$$

Expression 1: Rate of lost forwarding database entries.

To calculate the expression it is necessary to poll the device twice, at time t1 and t2. If the rate is higher than zero, it indicates that a problem exists and that the user should know about it.

After having defined the expressions, it is necessary to translate it to a form that the Expression MIB understands. It is necessary to describe the expressions in terms of MIB objects according to the SNMP syntax. In this case the mapping will be:

```
expExpression.3."adm".4."lost" =  "$1"
expExpressionValueType.3."adm".4."lost"  =  integer32
expExpressionDeltaInterval.3."adm".4."lost" =  5
expExpressionRowStatus.3."adm".4."lost"  =  'active'

expObjectID.3."adm".4."lost".1 =
   dot1dTpLearnedEntryDiscards.0
expObjectSampleType.3."adm".4."lost".1   =  'deltaValue'
expObjectRowStatus.3."adm".4."lost".1 =  'active'
```

It is not necessary to normalize the expression, i.e. divide the difference between successive samples by the sampling interval, because the period does not change – the x axis delta interval is always the same. Thus, the expression is composed only of a single parameter with delta sampling. Moreover, there are no wildcarded objects because the sampling object is a leaf object.

Task 2 - Percentage of active processes

This task is valuable in workstations to have an idea of resource occupation. This task evaluates the percentage of current system processes (*hrSystemProcesses*) to the maximum number of processes that a system can support (Expression 2). The objects are defined in the Host Resources MIB (Waldbusser, 2000).

$$\text{process percentage} = \frac{\text{hrSystemProcesses}}{\text{hrSystemMaxProcesses}} \times 100$$

Expression 2: Percentage of active processes.

The correspondent Expression MIB configuration is:

```
expExpression.3."adm".3."sys"   =  "$1/$2*100"
expExpressionValueType.3."adm".3."sys"   =  unsigned32
expExpressionDeltaInterval.3."adm".3."sys"  =  5
expExpressionRowStatus.3."adm".3."sys"   =  'active'

expObjectID.3."adm".3."sys".1   =  hrSystemProcesses.0
expObjectSampleType.3."adm".3."sys".1 =  'absoluteValue'
expObjectConditional.3."adm".3."sys".1   =
   hrSystemMaxProcesses.0
expObjectRowStatus.3."adm".3."sys".1  =  'active'
expObjectID.3."adm".3."sys".2   =
   hrSystemMaxProcesses.0
expObjectSampleType.3."adm".3."sys".2 =  'absoluteValue'
expObjectRowStatus.3."adm".3."sys".2  =  'active'
```

The conditional object is responsible for invalidating the expression if the `hrSystemMaxProcesses` is not defined for the given host. In this case, the expression should not be evaluated (divide by zero). Also, by invalidating a single parameter, the whole expression is invalidated, so it is not necessary to set this object for the second parameter.

Task 3 - Interface utilization for half-duplex links

Calculate the network utilization per element (Expression 3). The best way to do this is by measuring the interface utilization, as described in (Leinwand, 1996) and (Cisco). The objects are defined in the MIB-II (McCloghrie, 1991).

$$\text{Interface utilization} = \frac{(\Delta\text{ifInOctets} + \Delta\text{ifOutOctets}) \times 8}{(\text{number of seconds in } \Delta) \times \text{ifSpeed}} \times 100$$

Expression 3: Interface utilization for half-duplex links.

The *ifInOctets* managed object has the count of all the octets received in a given interface; *ifOutOctets* represents the count of the transmitted octets; *ifSpeed* has the speed of the interface in bits per second.

If the link is full duplex, the equation evaluates to 200%, so it is necessary to get the maximum of the `ifInOctets` and `ifOutOctets`. However, this approach hides the direction that has lesser value and provides less accurate results. It is better to calculate the interface utilization in each direction by dividing the equation in two. For simplicity, we only measure half-duplex links.

The translation to MIB objects for this expression is already described in (Kavasseri, 2001), so it is straightforward.

Task 4 – Interface accuracy

We also decided to measure the interface traffic that does not result in error (expressed in percentage), also known as the interface accuracy (Cisco). The result of Expression 4 compares errors to total packets seen and sent and subtracts this percentage from 100, to get the accuracy. An accuracy of 100% implies that no errors have occurred and an accuracy of 97% means that in every 100 packets, 3 where lost due to errors.

$$\text{accuracy} = 100 - \frac{\Delta\text{ifInErrors}}{\Delta\text{ifInUcastPkts} + \Delta\text{ifInNUcastPkts}} \times 100$$

Expression 4: Interface accuracy.

ifInErrors has the count of the received octets that resulted in error; the sum of *ifInUcastPkts* (the number of received unicast octets) with *ifInNUcastPkts* (the number of received octets which are not unicast) gives the total number of received packets.

```
expExpression.3."adm".4."cond" =  "$1==1"
expExpressionValueType.3."adm".4."cond"  =  unsigned32
expExpressionRowStatus.3."adm"4."cond"   =  'active'

expExpression.3."adm".4."accu" =  "100-$1/($2+$3)*100"
expExpressionValueType.3."adm".4."accu"  =  integer32
expExpressionDeltaInterval.3."adm".4."accu" =  5
expExpressionRowStatus.3."adm"4."accu"   =  'active'

expObjectID.3."adm".4."cond".1  =  ifConnectorPresent
expObjectWildcard.3."adm".4."cond".1  =  'true'
expObjectSampleType.3."adm".4."cond".1   =
   'absoluteValue'
expObjectRowStatus.3."adm".4."cond".1 =  'active'

expObjectID.3."adm".4."accu".1  =  ifInErrors
expObjectWildcard.3."adm".4."accu".1  =  'true'
expObjectSampleType.3."adm".4."accu".1   =  'deltaValue'
expObjectConditional.3."adm".4."accu".1  =
expValueUnsigned32Val.3."adm".4."accu".0.0
expObjectConditionalWildcard.3."adm".4."accu".1=
   'true'
expObjectDiscontinuityID.3."adm".4."accu".1 =
   ifCounterDiscontinuityTime
expObjectDiscontinuityIDWildcard.3."adm".4."accu".1   =
   'true'
expObjectRowStatus.3."adm".4."accu".1 =  'active'

expObjectID.3."adm".4."accu".2  =  ifInUcastPkts
expObjectWildcard.3."adm".4."accu".2  =  'true'
expObjectSampleType.3."adm".4."accu".2   =  'deltaValue'
expObjectRowStatus.3."adm".4."accu".2 =  'active'

expObjectID.3."adm".4."accu".3  =  ifInNUcastPkts
expObjectWildcard.3."adm".4."accu".3  =  'true'
```

```
expObjectSampleType.3."adm".4."accu".3   =  'deltaValue'
expObjectRowStatus.3."adm".4."accu".3 =  'active'
```

Just like in Expression 3, we use an equal operation to set the conditional object for the expression evaluation. It should only be evaluated if the interface is connected (`ifConnectorPresent==1`). The following parameters directly correspond to delta values from the `ifTable`.

4 EVALUATION

The most important obstacle in accomplishing the requirements is the practical impossibility to upgrade the switches' SNMP agent to support the Expression MIB. To solve this problem it would be necessary to:

a) upgrade the operating system flash with a version supporting the Expression MIB;
b) use some kind of extension mechanism to add Expression MIB support to the existing agents. The AgentX standard does not allow for sub-agents to retrieve values from the master agent (Daniele, 2000).

Neither of these options is available at the moment so we had to go for a different approach: introduce changes to the Expression MIB so that it allows retrieving values from remote agents (Lopes, 2003). This approach uses the concept of SNMP URLs to store the information required to communicate with remote agents (Lopes, 2002). For example, to select the *sysUpTime* instance of the router.ipb.pt to be used in an expression we would use the following URL:

```
snmp://senior@router.ipb.pt/sysUpTime/0??v2c
```

This URL is stored in the *expObjectTable* together with regular variables and so it is considered as a remote variable. The expression syntax does not change.

This approach allows us to consider even further management tasks based on the definition of expressions:

Task 5 – Service correlation

The final interesting task is the possibility to correlate different but related services. For example, we have all the user account information in an OpenLDAP (OpenLDAP) directory service and the file server is based on SAMBA (SAMBA). The latter retrieves the user account information from the LDAP server and so it is dependent on it (Figure 2).

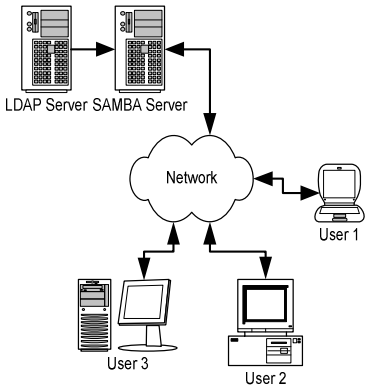

Figure 2: LDAP based file server.

From the management point of view, if either service fails then the access to files from the users also fails.

The goal of this task is to check if the overall service intervenient are working properly. For this we also defined the function `checkService($1)`, where $1 is of type URL. This function returns the boolean TRUE if the service responds correctly (value 1) and FALSE if the service does not respond (value 0). By summing several functions we can get the overall response (Expression 5).

$$total = checkService(\$1) + checkService(\$2)$$

Expression 5: Service correlation expression.

The SNMP description of the expression is:

```
expExpression.3."adm".4."srvc" =
   "checkService($1)+checkService($2)"
expExpressionValueType.3."adm".4."srvc"  =  Unsigned32
expExpressionDeltaInterval.3."adm".4."srvc" =  20
expExpressionRowStatus.3."adm".4."srvc"  =  'active'

expObjectID.3."adm".4."srvc".1  =  ldap://ldap.ipb.pt
expObjectSampleType.3."adm".4."srvc".1   =
   'absoluteValue'
expObjectRowStatus.3."adm".4."srvc".1 =  'active'

expObjectID.3."adm".4."srvc".2  =  smb://samba.ipb.pt
expObjectSampleType.3."adm".4."srvc".2   =
   'absoluteValue'
expObjectRowStatus.3."adm".4."srvc".2 =  'active'
```

This implementation is available online as open source and it is also associated with preliminary work on the Event MIB. The agent as well as the source code may be retrieved from http://nms.estig.ipb.pt/.

5 CONCLUSIONS

Management distribution is a requirement to modern networks. As features appear and technology evolves, better tools are needed to maintain the network in excellent working condition.

The DISMAN workgroup have defined a rather complete set of MIB modules to ease the distribution of management tasks under the context of SNMP, which become compatible with the vast majority of installed systems.

Mathematical expressions are fundamental to process and somehow filter the knowledge behind the evolution of network working parameters. The Expression MIB is responsible for these tasks and we have presented also several scenarios where the concept of management task provided by the Expression MIB can be applied with success.

The Expression MIB does not allow retrieving values from remote agents and it is not practical to upgrade existing SNMP agents with this functionality. To solve this problem we have developed and extended the Expression MIB functionality with the possibility to retrieve values from remote agents. We have found this functionality most valuable because it allows not only to retrieve values from other locations but also to correlate information from different sources.

REFERENCES

Martin-Flatin, J., Znaty, S., Hubaux, J., 1998. A Survey of Distributed Network and Systems Management Paradigms. In *Technical Report SSC/1998/024*, Swiss Federal Institute of Technology Lausanne, August 1998.

Yemini, Y., Goldszmidt, G., Yemini, S., 1991. Network Management by Delegation. In *proc. IFIP 2nd Int. Symposium on Integrated Network Management - ISINM'91*, Washington, DC, USA, April 1991.

Goldszmidt, G., Yemini, Y., 1998. Delegated Agents for Network Management. In *IEEE Communications Magazine, Vol. 36 No. 3*, March 1998, pp. 66-71.

Bieszczad, A., Pagurek, B., White, T., 1997. Mobile Agents for Network Management. *Carleton University*, Canada, 1997.

Pham, V., Karmouch, A., 1998. Mobile Software Agents: An Overview. In *IEEE Communications, Vol. 36, No. 7*, July 1998, pp. 26-37.

Krause, S., Magedanz, T., 1996. Mobile Service Agents enabling Intelligence on Demand in Telecommunications. *In proc. of IEEE GLOBCOM'96*, 1996.

Lopes, R., Oliveira, J., 1999. Software Agents in Network Management. In *proc. of the 1st International Conference on Enterprise Information Systems – ICEIS'99*, March 1999, Setúbal, Portugal.

Sun Microsystems. Java™ Management Extensions Instrumentation and Agent Specification, v1.0. http://www.javasoft.com/.

SNMP Research. http://www.snmp.com.

DISMAN Charter. http://www.ietf.org/html.charters/disman-charter.html

Schoenwaelder, J., 1997. Network Management by Delegation: from Research Prototypes towards Standards. In *proc. 8th Joint European Networking Conference - JENC8*, Edinburgh, Scotland, UK, May 1997.

Lopes, R., Oliveira, J., 2000. Distributed Management: Implementation issues. In *proc. of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet – SSGRR 2000*, August 2000, L'Aquila, Italy.

Decker, E., Langille, P., Rijsinghani, A., McCloghrie, K., 1993. Definitions of Managed Objects for Bridges. *Internet Request for Comments 1493*, July 1993.

Waldbusser, S., 2000. Host Resources MIB. *Internet Request for Comments 2790*, March 2000.

Leinwand, A., Conroy, K., 1996. Network Management. A Practical Perspective – 2ed. September 1996.

Cisco Systems, How To Calculate Bandwidth Utilization Using SNMP.

McCloghrie, K., Rose, M., 1991. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. *Internet Request for Comments 1213*, March 1991.

Kavasseri, R., Stewart, B., 2000. Distributed Management Expression MIB. *Internet Request for Comments 2982*, October 2000.

Cisco Systems, Performance Management: Best Practices White Paper.

Daniele, M., Wijnen, B., Ellison, M., Ed., Francisco., D., Ed., 2000. Agent Extensibility (AgentX) Protocol Version 1. *Internet Request for Comments 2741*, January 2000.

Lopes, R., Oliveira, J., 2003. Delegation of Expressions for Distributed SNMP Information Processing. Submitted to *Integrated Network Management – IM 2003*, Colorado Springs, Colorado, USA 2003.

Lopes, R., Oliveira, J., 2002. A Uniform Resource Identifier Scheme for SNMP. In *proc. Workshop IP Operation & Management – IPOM 2002*, Dallas, TX, USA 2002.

OpenLDAP. http://www.openldap.org/.

SAMBA. http://www.samba.org/.