DELEGATION OF EXPRESSIONS FOR DISTRIBUTED SNMP INFORMATION PROCESSING

Rui Pedro Lopes and José Luís Oliveira
Polytechnic Institute of Bragança, ESTiG; 5300-302 Bragança; Portugal (rlopes@ipb.pt)
University of Aveiro, DET; 3810-193 Aveiro; Portugal (jlo@det.ua.pt)

Abstract: Due to the scalability problems of SNMP, management distribution has been an important topic during the last years. The DISMAN workgroup propose a set of MIB modules to address this matter. One of the DISMAN modules has the capability of using expressions to perform decentralized processing of management information – the Expression MIB.

Although it is essential to network management, the Expression MIB is not as well known as other DISMAN modules, such as the Script MIB, and not as available in terms of implementations. This paper focuses on the Expression MIB features, its implementation details and it also discusses, from a critical point of view, its functionality. It also proposes minor changes which can boost its application range and importance.

Key words: DISMAN, Expression MIB, Network Management, SNMP.

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 APIs wide 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 to offline operation and not scalable on large networks.

According to many authors, the solution to this problem is management distribution, a research topic since the early 90's[1]. More recently, mobile agents have also been a hot research topic for management applications[2-5]. Considering the standardization of a common model for wide adoption in IP-networks, the IETF have also endorsed a working group to study this matter - the IETF Distributed Management charter, or DISMAN[6].

This group has set up a solid framework composed of several MIB modules, namely the Script, Schedule, Expression, Event, Remote Operations, Notification Log, Alarm and Condition MIB[6]. This set of MIB modules provides a rather complete framework for distributing management operations over a hierarchy of several midlevel managers – distributed managers (DMs).

The Script MIB is one of the best known, probably because of the early availability of implementations. Schoenwaelder, following an excellent study of distribution models and solutions, presents the distribution of management tasks in the context of the IETF Script MIB[7]. However, some other DISMAN modules seem to have captured less interest from the research community and have been left outside development plans. Our aim to evaluate management distribution in SNMP has also led us to the development and assessment of the Expression, Schedule and Event MIB.

In this paper, we present our work on the Expression MIB. We describe some implementation details, which are considered relevant in this paper context, and we evaluate the advantages and disadvantages of the current specification[8]. Based on the development experience and on its trial in the network, we also propose adaptations to the MIB that enhance significantly its functionality with minor modifications in its structure.

The paper is structured as follows: Section 2 presents a general description of management distribution in SNMP. Section 3 describes the Expression MIB model and our open source architecture. It point out the assessment of this management agent, and suggests some minor changes which can boost its applicability and potentiality. The paper ends with some conclusions.

2. MANAGEMENT DISTRIBUTION

The history of management distribution, well discussed by Martin-Flatin in[9], started with early work by Yemini *et al.* in 1991 when features such as scalability, flexibility and robustness were identified as necessary for future developments on network management[10]. Goldszmidt and Yemini early supported a management distribution methodology by delegating management operations near management information[1]. 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[2-5].

The industry also adopted management distribution by releasing tools, APIs or agents, such as Sun's JMX[11] or SNMP Research's CIAgent[12].

Some of these products, technology and concepts do not easily survive the community resistance because they are neither compatible nor adapted to the management technology of choice – the SNMP. The SNMP community has also proposed, under the DISMAN workgroup of the IETF[6], several tools for management distribution.

2.1 Distribution under DISMAN

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

Several MIB modules were proposed to address different but complementary issues of management operations distribution. The Event MIB allows monitoring the real-time evolution of specific MIB objects either locally or remotely, and takes an action when a trigger condition occurs (a value outside a range limit for instance)[13]. The action can result in a set operation or in a notification.

The Notification Log MIB is intended mainly for notification providers but consumers may also use it. It defines a mechanism to cope with lost notifications by recording each notification data[14].

The Remote Operations MIB modules (`ping`, `traceroute`, `lookup`) enable the corresponding network-checking operation to be performed at a remote location. It provides a standard way to perform remote tests, to issue periodical sets of operations, and to generate notifications with test results[15].

The Schedule MIB provides the definitions to perform the scheduling of actions periodically or at specific times and dates. The actions are modelled by SNMP set operations on local MIB variables (restricted to `INTEGER` type). More complex actions can be performed by triggering a management script, which is responsible for carrying out complex state transitions[16].

The Script MIB module allows the delegation of management functions over distributed managers. Management functions are defined as management scripts written in a language supported by the managers. It may be a scripting language (such as TCL) or native code, if the implementation is able to execute it under its control. The module does not make any further assumptions on the language. The distributed manager may be decomposed into two blocks: the SNMP entity, which implements this MIB, and the runtime system, capable of executing the scripts. The Script MIB sees the runtime system as the managed resource, which is controlled by the MIB[17].

The Expression MIB was planned to move to the agent side part of the management information processing typically performed by managers. In other words, it supports externally defined computation expressions over existing MIB objects. The Expression MIB allows providing the Event MIB with custom-defined objects. The result of an expression can trigger an event, resulting in an SNMP notification. Without the Expression MIB such monitoring is limited to the objects in predefined MIBs[8].

The most recent modules are related to alarm reporting. In fact, SNMP is mainly based on polling instead of alarm reporting (a node sending notifications to the manager when status changes to and from fault conditions). These modules provide the necessary mechanisms to build up management procedures based upon exception handling. It starts by defining a model-neutral method to specify and store alarms[18] thus providing the DM access to SNMP alarm information in a consistent manner across systems.

The DISMAN architecture may be classified somewhere between the weak distribution model and the strong distribution model, according to the user defined distribution policies[19]. In this case, the degree of distribution depends on the set of operations defined in the DM, the complexity of the operations that the DM supports, the communication between them, and the total number of DMs.

It is possible to foresee several examples of scenarios suitable for DISMAN application. For example, the local processing of RMON probes statistics[20] according to specific expressions, the autonomous resource monitoring in workstations through the HOST-RESOURCES-MIB[21], the periodic analysis of data integrity in persistent storage and many more. These tasks can be built around the Expression MIB, Event MIB, Schedule MIB and Script MIB, respectively.

2.2 Delegation of SNMP Operations

SNMP operations may be executed in three different locations according to its nature, complexity and network requirements (Figure 1): on the central manager, on the distributed manager, or on the agent. Each situation is non-exclusive, meaning that any combination of these scenarios is possible and sometimes it is even recommended.

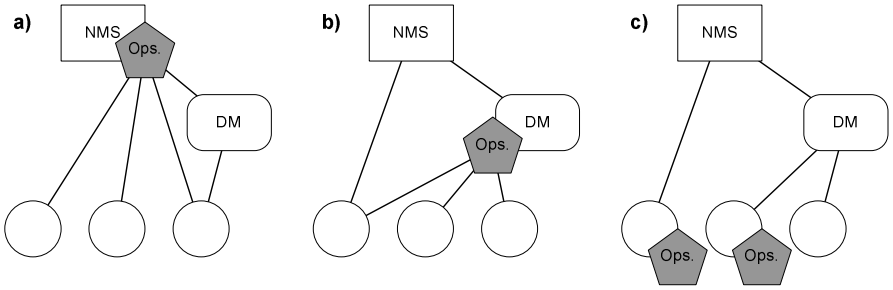

Figure 1. SNMP task execution (Ops.)
a) on the management station (NMS), b) on the distributed manager (DM), c) on the agent

Under DISMAN, each MIB encapsulates a set of operations around a specific concept. This fact allows using almost any combination of modules to achieve the above distribution models. However, some modules are more tightly related than others due to their type, i. e., some modules work better if associated with others which complement their capabilities. For example, the Script MIB cannot autonomously start scripts. This feature is obtained by associating it to the Schedule

MIB. On the other hand some associations are useless, such as the pair <Schedule MIB, Expression MIB>.

Considering local or remote information handling, some modules are able to interact with remote agents (ex. Script, Event) and others are restricted to the local agent (Expression, Schedule). This fact reduces the number of useful associations on real management scenarios:

- a single Schedule MIB cannot start scripts on different locations: it must be multiplied;
- an expression in the Expression MIB cannot obtain values beyond the local agent: a script could help although it is impossible to prevent an increase in complexity.

We have developed and evaluated the Schedule MIB and the Expression MIB, and we are currently performing some work around the Event MIB. The following section will focus mainly on the Expression MIB. It describes the implementation strategy and the evaluation of several aspects such as overhead, functionality and, generically, it points out the general advantages and disadvantages collected from the experience of placing this functionality inside the agent.

3. EXPRESSION MIB

We started working on the Expression MIB implementation when the documentation was still in the Internet draft condition[22]. 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 expression functions according to the clarifications made as the document evolved. We also improved the sampling mechanism to cope with the Event MIB requirements so that it could be used in both modules.

The MIB is divided into three main groups:

- *expResource* – this group is related to resource control, with particular emphasis 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 error 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 it may either be wildcarded or not.
- *expValue* – this group has a single table which instantiates the evaluation objects. It is by querying this table that the result of 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 an SNMP get operation, wildcard objects are retrieved through the 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 maintain coherence in the 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 (Figure 2).

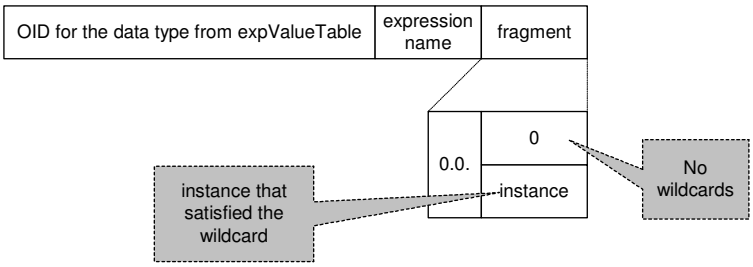

Figure 2. Value identification OID

The *expression name* has the form 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 also converted to a dot separated integers format according to the `SnmpAdminString` textual convention[23].

The *fragment* starts with "0.0." and it ends with a zero, when no wildcard is defined, or with the instance that satisfied the wildcard.

3.1 Implementation issues

The development of management agents is a complex and tedious task. To cope with these difficulties we have developed an open source, extensible API gathering all the agent common procedures – the Agent API[24]. The modular approach of this system allows (already developed) direct access to the agent through SNMP, RMI, CORBA, HTTP or WAP.

The Expression MIB* uses the Agent API services to provide the common SNMP mechanisms and to simplify and accelerate the agent development (Figure 3).

According to the expression properties defined in the tables *expObjectTable* and *expExpressionTable*, the Timer module wakes up at every delta interval and the Sampling module calculates the value according to the sampling type (`absolute`, `delta` or `changed`). This value is then forward to the Expression Parser.

* The implementation of the Expression MIB as well as implementations of the Schedule MIB and preliminary efforts on the Event MIB are available at http://nms.estig.ipb.pt/.

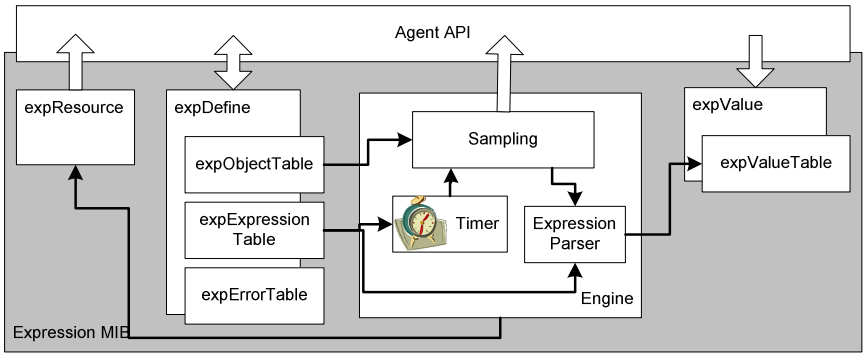

Figure 3. Architecture of the Expression MIB implementation

The Expression Parser module is responsible for evaluating the expression. To achieve this goal it must recognize the expression components (operators, functions, constants and variables), i.e. the lexicon and the grammar (the expression organization). There are, available as public domain software, lexical and grammar analysis tools, which generate code such as C[25] or Java[26, 27]. As this implementation is Java based, we choose Javacc[27]. This tool generates source code based on specification files which are then compiled (into Java .class files) and included in the Expression MIB agent.

The lexical analyser starts by reading the stream of characters and tries to match the sequences by identifying tokens. The tokens' information is forwarded to the grammar, which groups them into meaningful sequences and invokes action routines to act upon them. In this particular case, it must recognize a complete expression and evaluate the result.

3.2 Comments to the Expression MIB

An expression is composed of operators, functions and values. The values may be constants or variables, the latter being associated with OIDs that refer to the corresponding value. A string defines each expression.

The variables are defined in a separate table and indexed by a number of the '$v' form where 'v' is the variable number. For example, the expression '$1+$2+$3' represents the sum of the variables '$1', '$2' and '$3'. Variable indexes (the number prefixed by '$') correspond to entries in a table (*expObjectTable*) that contains an OID and the additional sampling parameters. Any expression can thus be defined according to the following generic format:

$$x = \text{Expression}(oid_1, oid_2, \ldots oid_n) \quad (1)$$

The Expression MIB retrieves the variable values from the local agent and evaluates the result (Figure 4).

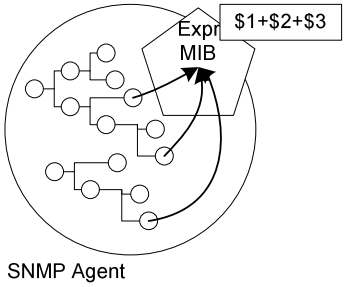

Figure 4. Expression MIB operation on local SNMP agents

The possibility of using variables in expressions is, simultaneously, the strength and the weakness of the Expression MIB. As currently proposed, the MIB does not allow retrieving values from remote agents restricting the expression evaluation to local objects. This limits the possibility of creating some expressions, for example, when they require values from different sources.

This scenario is modelled as weak distribution because the connectivity between DMs is non-existent, the delegation is occasional (restricted to expressions valid only in the local agent) and the number of DMs with the Expression MIB is usually low (it is more meaningful near raw management information, typical on the agent side). These facts limit the usage of the Expression MIB to a single situation – the agent side (Figure 5).

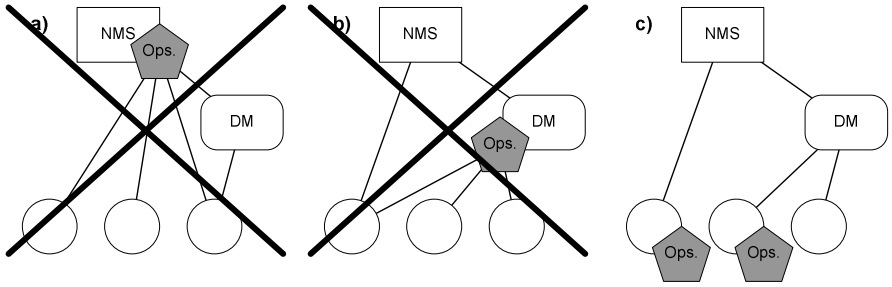

Figure 5. Expression MIB usage situations

Moreover, the integration of the Expression MIB module in existing SNMP agents is not possible unless the agent uses some extensibility feature, such as AgentX[28]. In this situation, the Expression MIB module could be the subagent and the existing SNMP agent would play the master role. Even in this case, some obstacles may persist because "Subagent access (via the master agent) to MIB variables" is a non-goal in AgentX.

The Event MIB, for instance, depends on the SNMP-TARGET-MIB to describe the remote hosts under monitoring. However, the same approach was not adopted for the Expression MIB, which eliminates the possibility to construct expressions over remote attributes. The use of "target-based" approach in the Event MIB caused some debate in the DISMAN workgroup because of the difficulty in storing credentials needed to contact remote hosts. However, it solves the problem (at least

theoretically) by storing the security name and model in the SNMP-TARGET-MIB and by looking for the keys in the *usmUserTable* of the SNMP-USER-BASED-SM-MIB[29].

The same approach could be used in the Expression MIB to provide this kind of feature thus allowing it to gather parameters from remote MIBs. This change would allow defining expressions where variables are mapped to target tags and then resolved with the help of the SNMP-TARGET-MIB:

$$x = \text{Expression}(\text{target}_1, \text{oid}_2, \ldots \text{target}_n) \quad (2)$$

In this case, the expression variables are defined as target tag references. The security credentials and profiles required for contacting remote agents are defined in the appropriate tables on the SNMP-TARGET-MIB.

To update the MIB with this functionality it would be necessary to modify the table responsible for designating the objects where parameters are obtained – *expObjectTable*. Like in the Event MIB, it would be necessary to introduce the following objects:

- *expObjectTargetTag* – specifies the remote system. Works together with the SNMP-TARGET-MIB.
- *expObjectContextName* – specifies the context used to get the parameter.
- *expObjectContextNameWildcard* – points out if the context name should be truncated for wildcards.

The main advantage of this approach is the compatibility with the current SNMPv3 framework. However, it requires an additional MIB module, the SNMP-TARGET-MIB.

A second approach to this problem is based on the use of SNMP URLs[30] in variable mapping instead of OIDs. We have previously proposed this concept as a way to globally identify network management information. The main idea is to use a single string to locate and configure network and systems parameters. Examples of this URL can be:

snmp://rlopes@sw1.estig.ipb.pt/sysContact/0?op=set&value=Rui?v3
snmp://guest@nms.estig.ipb.pt:161/sysUpTime?op=getNext?v3?router

Within the Expression MIB these specifications can be used to obtain values from remote agents thus increasing its flexibility and capability. It is then possible to use expressions like:

$$x = \text{Expression}(\text{url}_1, \text{oid}_2, \ldots \text{url}_n) \quad (3)$$

This second solution implies replacing the OID column by a URL column, which allows changing only the column data type. It uses SNMP URLs to designate the target host, the object, context and other communication parameters in a single line of text. Contrarily to the first solution, this one will not require implementing a new MIB module. The only change to the Expression MIB occurs with the *expObjectID* column. Despite this modification is conceptually simple, considering the IETF standardization track, it implies the redesign of the MIB or even the proposal of a new one. The processing mechanism for each expression has to be

further elaborated in order to cope with the URLs, but the increase in complexity is not too significant.

Regardless of the adopted solution, if the Expression MIB were allowed to obtain remote parameters it would be possible to obtain values from remote agents as well as remote DMs. Moreover, it could be associated to the network management station as well as other DMs and agents (Figure 6).

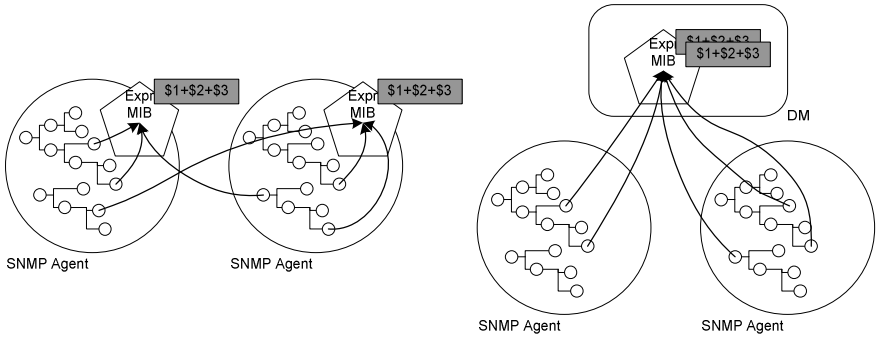

Figure 6. Modified Expression MIB operation

A bigger picture is shown in Figure 7, revealing the increase in connectivity among DMs and agents equipped with the Expression MIB. This scenario makes it possible to use variables from different sources within the same expression thus making the information correlation possible. Moreover, according to the set of expressions defined by the manager, it is now possible to build a strong distribution scenario.

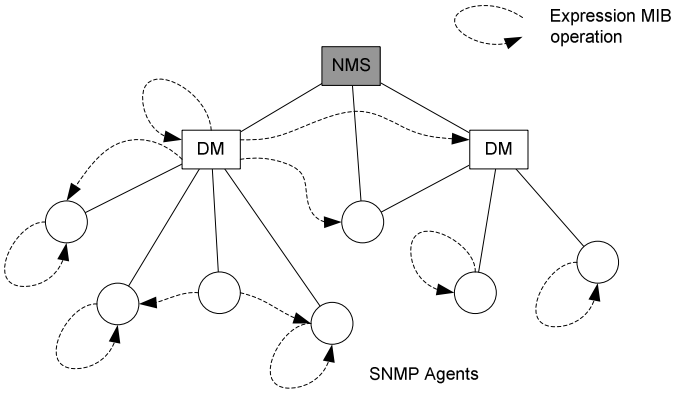

Figure 7. Strong distribution with the Expression MIB

These considerations may also be extended to other modules, such as the Schedule MIB. The current specification does not allow it to perform operations on remote modules, limiting its functionality to the local entity. This limitation may be

eliminated by associating it with the Script MIB although some situations do not require such an elaborate tool. The extension of the Schedule MIB to allow remote operations does not seem to significantly increase the overall complexity and it would enlarge the usage possibilities because of the added flexibility.

Generally speaking, the DISMAN architecture can successfully distribute SNMP management operations because it is completely compatible and has a set of meaningful autonomous management operations. It should, however, allow the possibility for evaluation of expressions with remotely obtained parameters and to start remote periodic or calendar actions, which would increase the overall flexibility.

3.3 Changes to the Implementation

To cope with the previous recommendations it is necessary to perform some changes to some objects in the Expression MIB implementation. The modification of standard MIB structures is not a straightforward process. However and for research purposes, this can be done in a controlled and private environment without disrupting the standard and without affecting other management systems.

Besides the obvious type changes on the managed objects (which could imply the creation of a new object and the deprecation of the current one), it is necessary to change the sampling mechanism to allow retrieving values from remote locations. A pleasant side effect of this design choice is that the new sampling mechanism also works unchanged in the Event MIB.

We have chosen the SNMP URL approach to reduce the complexity and to minimize the table structure modification: only a single column type is modified.

There is however another important detail which cannot be left unnoticed: the expression functions. According to the specification, expressions can use a set of functions which work with a broad choice of value types, such as constants, variables, OIDs and others.

Will the SNMP URL approach dramatically change the implementation of functions?

For example, the function `sum(integerObject*)` may receive an OID or a variable referring to an OID, causing it to sum all the integer values of the wildcarded object. Will this definition suffer some modifications to cope with the above recommendations?

The answer is no. The resolution of the parameters happens before calling the function. So, the function `sum(OID)` will receive an array of integer values. The same happens to `sum($1)` or `sum(snmpURL)`, all resulting in an array of integer values to be summed. The result of the function remains unchanged regardless of the parameter type because the latter is resolved the function is called.

The main changes to the implementation happen at the sampling level. The suggested sampling mechanism is modular and relies on inheritance to provide different access and sampling methods. We should not forget that the sampling can be regular or wildcarded and absolute or delta or changed. Moreover, it can target a local or a remote peer. By chaining together the appropriate classes we will be able to build sampling mechanisms for any of these combinations as well as targeting different location hosts (Figure 8).

The Sampler uses Peer objects to provide the location dependent classes, namely the access to local or to remote agents. The value transformation (absolute, delta, changed) and access method (regular, wildcard) is provided by chaining Sampler

classes. For example, the following code defines the access object to a remote wildcard MIB object with delta sampling:

```
Sampler sampler = new DeltaSampler(new WildcardSampler(remotePeer));
Sampler.sample();
Sample sample = sampler.getSample();
```

Further sampling examples may be built by using different classes and different constructor parameters.

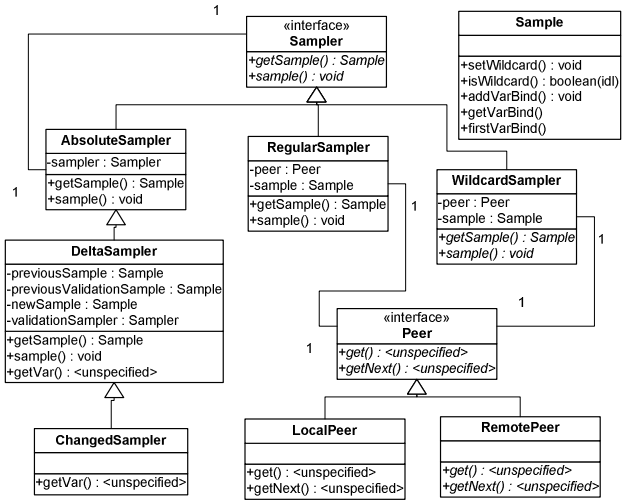

Figure 8. Sampling mechanism class diagram

4. 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. Among them, the Script MIB is the most studied and deployed module thus gathering a reasonable degree of knowledge around its features and applicability.

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 but, unfortunately, it is not allowed to use values from remote agents in the expressions.

In this paper we presented a possible solution to remote data retrieving by the Expression MIB. By performing some simple changes to the specification, this MIB can be extended to retrieve data from remote locations such as the Event MIB. Due to its importance the Expression MIB should be less restricted.

REFERENCES

[1] G. Goldszmidt, Y. Yemini, "Delegated Agents for Network Management", IEEE Communications Magazine, Vol. 36 No. 3, March 1998, pp. 66-71.
[2] A. Bieszczad, B. Pagurek, T. White, Mobile Agents for Network Management, Carleton University, Canada, 1997.
[3] V. Pham, A. Karmouch, "Mobile Software Agents: An Overview", IEEE Communications, Vol. 36, No. 7, July 1998, pp. 26-37.
[4] S. Krause, T. Magedanz, "Mobile Service Agents enabling Intelligence on Demand in Telecommunications", Proc. IEEE GLOBCOM'96, 1996.
[5] R. Lopes, J. Oliveira, "Software Agents in Network Management", proc. of the 1st International Conference on Enterprise Information Systems – ICEIS'99, March 1999, Setúbal, Portugal.
[6] DISMAN Charter (http://www.ietf.org/html.charters/disman-charter.html).
[7] J. Schoenwaelder, "Network Management by Delegation: from Research Prototypes towards Standards", Proc. 8th Joint European Networking Conference - JENC8, Edinburgh, Scotland, UK, May 1997.
[8] R. Kavasseri, B. Stewart, "Distributed Management Expression MIB", Internet Request for Comments 2982, October 2000.
[9] J. Martin-Flatin, S. Znaty, J. Hubaux, "A Survey of Distributed Network and Systems Management Paradigms", Technical Report SSC/1998/024, Swiss Federal Institute of Technology Lausanne, August 1998.
[10] Y. Yemini, G. Goldszmidt, S. Yemini, "Network Management by Delegation", I. Krishnan and W. Zimmer (Eds.), Proc. IFIP 2nd Int. Symposium on Integrated Network Management - ISINM'91, Washington, DC, USA, April 1991.
[11] Sun Microsystems, "Java™ Management Extensions Instrumentation and Agent Specification, v1.0", (http://www.javasoft.com/).
[12] SNMP Research (http://www.snmp.com).
[13] R. Kavasseri, B. Stewart, "Event MIB", Internet Request for Comments 2981, October 2000.
[14] R. Kavasseri, B. Stewart, "Notification Log MIB", Internet Request for Comments 3014, November 2000.
[15] K. White, "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", Internet Request for Comments 2925, September 2000.
[16] D. Levi, J. Schoenwaelder, "Definitions of Managed Objects for Scheduling Management Operations", Internet Request for Comments 3231, January 2002.
[17] D. Levi, J. Schoenwaelder, "Definitions of Managed Objects for the Delegation of Management Scripts", Internet Request for Comments 3165, August 2001.
[18] S. Chisholm, D. Romascanu, "Alarm MIB", draft-ietf-disman-alarm-mib-07.txt, June 2002.

[19] J. Schoenwalder, J. Quittek, C. Kappler, "Building Distributed Management Applications Using the IETF Script MIB", IEEE Journal on Selected Areas in Communications, Vol. 18, Nº 5, pp. 702-714, IEEE Communications Society, May 2000.
[20] S. Waldbusser, "Remote Network Monitoring Management Information Base", Internet Request for Comments 1757, February 1995.
[21] S. Waldbusser, "Host Resources MIB", Internet Request for Comments 2790, March 2000.
[22] R. Lopes, J. Oliveira, "Distributed Management: Implementation issues", 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.
[23] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", Internet Request for Comments 2571, April 1999.
[24] Agent API (http://nms.estig.ipb.pt/).
[25] T. Manson, D. Brown, lex & yacc, O'Reilly & Associates, 1990, ISBN 0-837175-49-8.
[26] A. Appel, A Modern Compiler Implementation in Java, Cambridge University Press, 1998, ISBN 0-521-58388-8.
[27] Javacc (http://www.webgain.com/products/java_cc/).
[28] M. Daniele, B. Wijnen, M. Ellison, Ed., D. Francisco. Ed., "Agent Extensibility (AgentX) Protocol Version 1", Internet Request for Comments 2741, January 2000.
[29] U. Blumenthal, B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", Internet Request for Comments 2574, April 1999.
[30] R. Lopes, J. Oliveira, "A Uniform Resource Identifier Scheme for SNMP", proc. of the 2002 IEEE Workshop on IP Operations & Management – IPOM 2002, Dallas, Texas, USA 2002.