The Simple Times™

THE QUARTERLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS
VOLUME 10, NUMBER 1 DECEMBER, 2002

The Simple Times is an openly-available publication devoted to the promotion of the Simple Network Management Protocol. In each issue, *The Simple Times* presents technical articles and featured columns, along with a standards summary and a list of Internet resources. In addition, some issues contain summaries of recent publications and upcoming events.

In this Issue:

Editorial

Aiko Pras, University of Twente
Jürgen Schönwälder, University of Osnabrück

This issue of *The Simple Times* is published at a historic moment: SNMPv3 has just been published as an Internet Standard (STD 62). As a consequence, the original SNMPv1 protocol has been removed from the official list of Internet standards by declaring it Historic. Despite that status change, we may expect that SNMPv1 will stay with us for many more years. Therefore it remains important to maintain SNMPv1 implementations and correct bugs. As demonstrated earlier this year, more than a decade of SNMP implementation experience could not prevent that there are still many bugs in existing implementations. In February, CERT published an advisory (CA-2002-03) that received a lot of attention within the Internet management community. This issue of *The Simple Times* includes an article from the Oulu University Secure Programming Group; the group that performed the tests that resulted in the CERT advisory mentioned above. In addition to that article, this issue of *The Simple Times* also includes an article discussing the correctness of MIB implementations.

The process of standardizing a successor for SNMPv1 has not always been easy. In fact, work already started in the earlier nineties with two activities to improve SNMP: one activity focused on adding security and the other on enhancing functionality. In 1992, it was decided to join these activities and produce a new standard called SNMPv2. The first set of SNMPv2 RFCs appeared in April 1993; these RFCs were based on the so-called "party based security model." Since SNMPv1 has been very successful, many groups started to experiment with SNMPv2 prototypes. After some time, it turned out that the party based model was difficult to understand and use. One reason for confusion was the fact that the administrative model was general enough to accommodate single (shared) key approaches, as well as two (public and private) key approaches. The actual protocols, however, realised only the second approach. After many heated discussions the climax was in June 1995, when two of the SNMPv2 editors ceased support for their party-based model. Only two months later, there was general agreement that the party based model was too complex; unfortunately, there was no agreement on how the alternative should look like. Consequently, many competing alternatives appeared; the best known ones were SNMPv2u, SNMPv2* and SNMPv2c. At that time, it seemed no longer obvious that discussions should be based on technical arguments; as a result this period turned out to be quite bad for SNMP's reputation. Fortunately, it was decided to form an SNMP Advisory Team, which recommended in 1996 to further progress a simplified version of SNMPv2. In March 1997, a new IETF working group was formed with the task to develop SNMPv3; this group published in January 1998 the first SNMPv3 RFCs as Proposed Standards. In April 1999, the status of SNMPv3 was raised to Draft Standard, and now SNMPv3 has become Internet Standard. This milestone should be important for software developers; an article that discusses the best known open source implementation that supports SNMPv3 is included in this issue of *The Simple Times*.

Despite the fact that SNMPv3 is now an Internet Standard, improvements are still possible. This issue includes an article that discusses possible improvements of the `GetBulkRequest`. Recently, the IETF has chartered a new working group that should work on the Evolution of SNMP (EOS). As the name of this working group already suggests, this group should take an evolutionary approach and propose relatively small improvements. Many possible improvements have already been proposed by the IRTF Network Management Research Group (NMRG). Since SNMP is primarily being used for monitoring purposes and has not been widely accepted for configuration purposes, several groups are currently discussing more revolutionary approaches. In June this year, the Internet Architecture Board (IAB) organised a special workshop to discuss the future of Internet management; an article about this workshop is included in this issue of *The Simple Times*. Various people within the IETF and IRTF-NMRG are now investigating revolutionary approaches for Internet management. Work in this area has just started and we hope to report on these activities in a future issue of *The Simple Times*.

IAB Network Management Workshop

Ran Atkinson, Extreme Networks

At the Spring IETF meeting, the Internet Architecture Board (IAB) announced that it planned to hold an IAB Network Management Architecture Workshop, in coordination with the IESG. The workshop was held in early June at the IETF Secretariat offices in Reston, VA.

IAB Workshops are normally invitation-only because of limited space for attendance and to keep the group small enough that focused architectural discussions are possible. Roughly 30 people were invited and about 25 people actually attended. There was a deliberate effort to get a broad mixture of people with experience in different technologies and different kinds of networks. This was largely successful, though enterprise network operators were probably under-represented. Invitations went out to people in North America, Europe, and Asia/Pacific. Most attendees were from Europe or North America.

Goals for the workshop were to discuss the various alternative approaches to various aspects of network management and to continue the effort to obtain more operator input on significant unresolved problem areas relating to network management.

Several people wrote white papers before the workshop that were distributed to the workshop invitees. This was very helpful in getting a wide range of perspectives out for discussion prior to the meeting. It is expected that several of these white papers will be made available either on the web or as Informational RFCs, as their authors prefer.

The workshop met for two and a half days, discussing a wide range of topics. The discussions were productive, but there were more potential topics than could fit into that time period. On the first morning, the group split into two. One sub-group consisted mostly of network operators and focused on enumerating unresolved network management problems they are facing. The other sub-group consisted mostly of protocol developers and developed a taxonomy of the several network management technologies that exist today. For the remainder of the workshop, the whole group met together. One of the several topics raised by the network operators was the challenge of managing configurations for the many devices inside the network – particularly remote devices in unstaffed locations. Throughout the workshop, there was also extensive discussion about the roles and opportunities for SNMP, COPS, COPS-PR, XML-oriented configuration, and other technologies. On the second day there was a significant discussion about perceived IETF process issues as those relate to network management technologies. A wide range of views emerged throughout the workshop. While unanimity was rare, consensus emerged on several topics after significant discussion.

The workshop report, which will be published as an Informational RFC once it is ready, will discuss the details of these consensus areas in detail. The attendees also agreed not to individually comment in public about what any conclusions of the workshop might have been, in order to prevent misunderstandings from emerging. IAB Workshops do not have any formal standing to force the IETF into undertaking any particular path forward. However, the output from past workshops has always been seriously considered by IETF participants as they go forward with Internet evolution. It is hoped the IETF community will find the workshop report to be a worthwhile input to network management activity going forward.

A brief synopsis of the workshop was presented at the Yokohama IETF in July 2002. An Internet-Draft of the workshop report is already online.

GetBulk Worth Fixing

Marek Malowidzki, Military Communication Institute

The `GetBulk` request is designed for an efficient bulk data transfer from SNMP agents. Its main use involves retrieval of large tables. However, the way it works makes it difficult to use sensibly in practice, mainly due to the "overshoot" problem. Despite new propositions for bulk transfer mechanisms for SNMP, `GetBulk` will remain the only such facility for some time. Thus, the paper proposes a simple fix for `GetBulk` that allows avoidance of reading unwanted data.

Introduction

One of the main flaws of the first version of the SNMP protocol, called SNMPv1, was the lack of an efficient data retrieval facility. The only way to retrieve MIB tables was to perform subsequent `GetNext` requests to get the data in a row-by-row fashion, which caused significant communication and processing overhead as well as long overall latency. The second version of the protocol operations tried to fix this problem through the introduction of a new PDU type, `GetBulk`, which could be used for more efficient table retrieval – a single request could return a larger number of table rows.

Unfortunately, the way `GetBulk` is designed makes it quite difficult to use it efficiently. A user has to decide about the number of rows he would like to read, and this may be impossible to predict in practice. Overestimation could cause an agent to perform unnecessary actions and return a big chunk of unwanted data, which is known as the "overshoot" problem.

There are many propositions of new bulk transfer mechanisms for SNMP. The most promising ones are discussed briefly later in the paper. However, it probably will take a longer time for these propositions to become widely available. Until then, the `GetBulk` operation will remain the only method of bulk transfer in SNMP. Thus, the paper proposes a fix to `GetBulk`, which relieves it of the "overshoot" effect.

GetBulk versus GetNext

Most books and general papers on SNMP management do not delve into `GetBulk` details, describing it as an "efficient bulk data transfer facility." However, the main problem with `GetBulk` is that the user (or the programmer) must decide about the number of repetitions for a single request. For tables (in the paper we concentrate on `GetBulk` used for reading tables), he must estimate the number of rows to retrieve. Sometimes a MIB contains a scalar variable holding the number of rows in a given table, as it is for the `ifTable` table (the `ifNumber` variable contains the number of interfaces). In most cases, however, this may be difficult to guess, especially for generic tools (e.g., MIB browsers, general-purpose communication libraries, etc.), which may deal with many different MIBs. An incorrect (too large) value may cause many unwanted variable bindings to be returned in an agent's response.

The problem arises from the fact that `GetBulk`, although mostly used for retrieving tables, is in fact just a more efficient version of `GetNext` and operates on data organized in a lexicographically ordered list rather than conceptual tables (despite the row-by-row order of returned data). The result is that `GetBulk` returns as much data as requested even if the end of a table has already been crossed – the subsequent variables in lexicographical order are returned.

Let us assume that we need to read a table that contains N rows (and, of course, we do not know this value). With `GetNext`, we must exchange N+1 messages. In the case of `GetBulk`, given that we set `max-repetitions` to Q, we need only (N+Q)/Q messages to be exchanged between a manager and an agent (we do not take message size limitations into consideration). However, the first approach returns one "row" of excess data while the second one may return at most Q unwanted "rows." It is clear that although `GetBulk` usually needs less messages to be exchanged, it also reads more data. Thus, `GetBulk` has the following advantages over `GetNext`:

- communication overhead is usually minimized
- end-to-end latency is also (usually) minimized
- processing overhead in both manager and agent may be smaller, especially when SNMPv3 is in use and every message is to be encrypted and authenticated

Note that the first two points above hold only if one is able to correctly estimate the number of rows in a table or if a table is big enough. This stems from the fact that the number of "overshot" variables returned by `GetBulk` is much bigger. Reading unwanted data (possibly from a completely different part of a MIB and thus from a different context) can result in significant processing or communication overhead and affect the overall latency. Thus, although `GetBulk` should be more efficient in many cases, especially for big tables, it sometimes may perform worse.

Extensions To Bulk Data Transfer

There have been quite a few recently proposed extensions to bulk data transfer capabilities of the SNMP framework. Some of them require changes to the protocol while others supplement SNMP. For in-depth coverage of the topic, consult [1] and [3].

We will describe shortly two more promising extensions, likely to be accepted. The first approach proposes the `GetCols` request, operating on conceptual columns and free from the "overshoot" problem [1]. An alternate proposal proposes a supplementary mechanism for providing bulk transfer capabilities without altering the protocol [2].

The new `GetCols` request operates on conceptual columns. It is required that the `noSuchInstance` error be returned for "holes" (non-existing values) in a table. We generally do not agree that such "holes" are a serious problem for most manager applications: It is relatively easy to put the response data in the appropriate order. Nonetheless, since this is a new message, it is easy to add other features such as `OID` compression (which could have been also defined for `GetBulk`).

The approach described in [2] contains a new proposal for bulk data transfer extensions, which employ a new `BULK-DATA-MIB`. Summarizing briefly, a manager asks an agent to use some application-layer protocol for data transfer (e.g., FTP), selecting columns to read (possibly from various tables), the data format to use (e.g., plain text or XML) and an optional compression method. Indeed, this capability may be of much value, especially when huge tables are to be retrieved (e.g., billing records or a call history). However, this proposition does not extend SNMP, but introduces a brand new mechanism instead. Because of the complexity it adds and the slow adaptation of new standards by communication equipment vendors, we do not expect this mechanism to be widely available soon (some vendors do have similar proprietary solutions). Besides, it remains outside of the SNMP framework – for example, it is interesting to ask what the relationship between SNMPv3 security and the security mechanisms used by the application-layer protocol is.

Fixing GetBulk

We propose here a simple way to fix `GetBulk` without altering the protocol or breaking existing implementations. A fix would be based on a "bumper" object, providing an end-of-usable-data marker. Such an object could be added to a new MIB of a very modest size. We propose the following solution:

```
SNMP-GET-BULK-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    snmpModules FROM SNMPv2-SMI;

snmpGetBulkMIB MODULE-IDENTITY
    LAST-UPDATED "200208080000Z"
    ORGANIZATION "Marek Malowidzki"
    CONTACT-INFO "malowidz@wil.waw.pl"
    DESCRIPTION  "The GetBulk MIB"
    ::= { snmpModules XXX }     -- not assigned

snmpGetBulkMIBObjects
    OBJECT IDENTIFIER { snmpGetBulkMIB 1 }

snmpGetBulkBumperTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SnmpGetBulkBumperEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "A virtual table of GetBulk bumpers.
                 Contains a row for each bumper index."
    ::= { snmpGetBulkMIBObjects 1 }

snmpGetBulkBumperEntry OBJECT-TYPE
    SYNTAX      SnmpGetBulkBumperEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "A GetBulk bumper entry."
    INDEX { IMPLIED snmpGetBulkBumper }
    ::= { snmpGetBulkBumperTable 1 }

SnmpGetBulkBumperEntry ::= SEQUENCE {
    snmpGetBulkBumper OBJECT IDENTIFIER
}

snmpGetBulkBumper OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The index part of the OBJECT IDENTIFIER for this object
        should be treated as an end-of-usable-data marker for
        the current GetBulk request, supplied as one of its
        non-repeaters and ignored in other scenarios.

        When the current GetBulk request would be to return
        an object with OBJECT IDENTIFIER equal to or older
        (lexicographically) than the marker, it should
        instead behave as if the end of a MIB view has been
        reached, that is, return the endOfMibView error and
        stop retrieving subsequent objects for supplied
        repeater variable binding.

        For successful Get operations, the value of 0.0 is
        returned."
    ::= { snmpGetBulkBumperEntry 1 }

END
```

A manager application would supply the bumper object as one of the `non-repeaters` objects in a `GetBulk` request. The trick is based on treating the index part of the bumper object's `OID` as the end-of-usable-data marker. We are using a `Get`-type operation to implicitly set a value in an agent. This is not as elegant as we would like it to be, but it does the job and allows the request to be self-dependent. As an example, to read the whole `ipNetToMediaTable` (`ip.22`), a manager could set the marker to `ip.23` (`ipRoutingDiscards`), thus telling an agent to refrain from reading the `IP-MIB` any further. Thus, the bumper object's `OID` would be set to

```
    snmpGetBulkBumper.1.3.6.1.2.1.4.23
```

Here we have a flexible variant of a `GetSubtree` operation, especially useful for reading a consistent subtree (for example, a group of subsequent columns in a table). The flexibility stems from the fact that the marker may be set at any place in a MIB. This approach may be a base for more sophisticated solutions at the expense of adding more complexity (and making it even trickier). For example, the index of the bumper object could contain markers for all repeater variable bindings, supplied as

```
  .<n-subids-up>.<n-marker-tail-subids>.<marker-tail-subids>
```

where `n-subids-up` indicates how many tail sub-identifiers in a repeater variable binding should be replaced by the `marker-tail` part to create the marker for this repeater. (To avoid the possibility that `n-subids-up` is 0 and the index starts with `.0`, we define it as the number of sub-identifiers incremented by 1.) Assume that we want to retrieve the `ifType`, `ifInOctects` and `ifName` columns but for some reason we need the statistics for the first ten interfaces only. In such a case, one could use:

```
GetBulk(<request-id>,
    1,
    100,
    snmpGetBulkBumper.2.1.4.1.1.11.2.1.2:NULL,
    ifType:NULL,
    ifInOctets:NULL,
    ifName:NULL)
```

In this example, `.2.1.4` sets the marker for the `ifType` repeater (`ifTable.3`) to `ifMtu` (`ifTable.4`), `.1.1.11` sets the marker for `ifInOctets` to `ifInOctets.11` and `.2.1.2` sets the marker for `ifName` (`ifXTable.1`) to `ifInMulticastPkts` (`ifXTable.2`).

Note that this may result in long bumper `OID`s. However, things may be refined by some compression scheme. For example, a special value of `.1` may be reserved to mean the most common case – iterating till the end of the repeater's subtree. Additionally, `n-subids-up` and `n-marker-tail-subids` could be encoded together. Finally, trailing `.1` markers could be (except for the first one) omitted. In this variant, assuming that we encode `n-subids-up` and (nonzero) `n-marker-tail-subids` as `128 * n-marker-tail-subids + n-subids-up`, the above example modifies itself to

```
GetBulk(<request-id>,
    1,
    100,
    snmpGetBulkBumper.1.129.11:NULL,
    ifType:NULL,
    ifInOctets:NULL,
    ifName:NULL)
```

To shortly compare the proposed fix with the `GetCols` request, one can notice that `GetCols` greatly reduces the need for the fix, although improving `GetBulk` in the proposed manner does not cost much, does not break existing implementations and may enable retrieving a part of a MIB subtree that does not exactly match any conceptual column. It may be used for both reading a part of a conceptual table (e.g., a row slice) and retrieving a bigger MIB subtree, containing both scalar objects and conceptual tables. We also believe that it would be easy to upgrade current manager applications using `GetBulk` to support the improved version. A manager that detects that an agent does not support this version could downgrade to the plain `GetBulk` and stop setting bumpers in subsequent operations.

The shortcoming of the proposition is that this is still the `GetBulk` message, with all other problems (this is obvious, since we must not modify its behavior) – just cured from the "overshoot" problem at the expense of a single variable binding being sent back and forth.

It remains open to discussion whether it is better to fix the existing `GetBulk` operation or design a brand new `GetSubtree` for reading any subtrees of data. While the second approach seems to be cleaner and could allow the addition of some important features (such as `OID` compression), we believe that this proposition is immediately deployable.

Finally, another possibility would be to incorporate the idea of the bumper object into the `GetCols` message, thus making it more flexible.

Summary

The `GetBulk` request indeed enables efficient bulk data transfer, but its design is flawed and undesirable side-effects may occur. The paper proposes a simple and effective improvement to this operation to free it from the "overshoot" problem.

We are not sure what the chances are for this proposition to be accepted. Anyway, we believe that it may be a base for new ideas and further discussions.

References

[1] Chandragiri, S., Efficient Transfer of Bulk SNMP Data, work in progress, April 2001

[2] Battle, D.L., Levin, B., SNMP Bulk Data Transfer Extensions, work in progress, March 2002

[3] Sprenkels, R., Martin-Flatin, J.-P., Bulk Transfers of MIB Data, The Simple Times, March 1999

Net-SNMP: Old Wine in a New Wineskin?

Wes Hardaker, Network Associates
Dave Shield, Liverpool University

It has been an interesting couple of years for the Net-SNMP project. The software suite continues to be used by an ever-increasing number of people, on a wide variety of operating systems and environments – at least judging by the traffic on the mailing lists (best part of 1000 messages a month, and rising), the number of software downloads and the increasing number of distributions that include the software. After many years being based at the University of California at Davis (UCD), whose support we would wish to acknowledge publicly, the project finally acknowledged the increasingly tenuous links with that institution, and renamed itself from UCD-SNMP to Net-SNMP. The project also moved to Source Forge, allowing us to take advantage of the various project features they offer (including bug tracking and web pages), and spread the administrative tasks more widely. For the first time, Wes can point the finger at the other "release managers" for delays in getting new versions of the software packaged up, rather than having to shoulder this burden alone. And most importantly, we have recently released the first version of the software under the new name which includes a number of changes and improvements to the code. The previous code is still being maintained under the old UCD-SNMP name, but this is basically just bug fixes, with no new features being added to that line.

Modularity and Extensibility

The main themes running through much of the new work have been modularity and extensibility. These are not new, of course – neither to software development in general, or to the Net-SNMP suite itself. One of the earliest changes to the UCD-SNMP agent from the original CMU-SNMP code, was splitting the MIB object handling code into a number of self-contained implementation modules. This enabled new modules to be developed and configured into (or omitted from) the agent very easily. Another, even earlier change, was the addition of support for external scripts to extend the agent – either by implementing particular MIB objects (so-called "pass-through" support), or just reporting non SNMP-aware textual output within a simple MIB table ("exec" support). The popularity of the UCD-SNMP package was largely due to the ways in which the package was extensible.

The first Net-SNMP release (version 5.0) has taken these modularity concepts and applied them even more widely throughout both the agent and the library. Most people using SNMP probably never think about how encoded PDUs get from one system to another – they are just "sent over the network" somehow. But the library now includes a new modular "transport layer," allowing additional externally-defined transports to be plugged in with no changes to the Net-SNMP code base, and particular transports to be selected (or omitted) when configuring the package. The Net-SNMP 5.0 release includes a number of new transports (including UDP and TCP over IPv6, AAL5PVC, IPX, Unix domain sockets and a new internal "callback" socket mechanism), as well as the traditional UDP and TCP over IPv4 transports, supported by the original UCD-SNMP code.

Similarly, in the mind of many people, SNMPv3 is probably synonymous with the User-Based Security Model. But of course, the SNMPv3 framework was deliberately designed to accommodate other security models as well. The library now also includes a pluggable security model mechanism, allowing new security models to be developed and selected (or omitted) as part of the configuration stage. The software is distributed with a reference implementation of the experimental Kerberos Security Model which is specified in an Internet-Draft.

MIB Handlers

The changes which are probably most visible to the majority of developers using the package are primarily contained within the agent. Most notably the agent core was rewritten and now supports a new MIB handler architecture. Implementing MIB objects has always involved a large amount of SNMP-related "donkey work" – comparing OIDs, handing index and instance sub-identifiers, and the like. Much of this is pretty predictable and can be handled via general utility routines, or automatically generated code templates. But MIB implementors in the past have always been exposed to an unnecessary amount of SNMP detail.

The latest agent is built around the idea of MIB handlers – modular chunks of code that can deal with much of the SNMP detail, and other shared handling tasks. At its most basic, a MIB implementor can simply define the structure of a table, or point to the relevant internal variable for a scalar object, and let "helper" handlers take care of the rest. Of course, these handlers still allow the implementor to chip in where necessary – perhaps where a particular object needs more individual handling. And there is a handler to support the previous UCD-SNMP version 4 module API, so existing MIB implementation modules will still work unchanged. But the general idea is to lift the load of dealing with SNMP syntax-related matters from the implementor wherever possible, and let them concentrate on the more important semantic behavior. Finally, the handlers are more dynamic in nature. At run time, it is now possible to make a section of the tree read-only by inserting the appropriate read-only handler into the run-time stack. A debugging handler also exists that can be inserted at run-time to help by producing debugging output when needed. The flexibility of the new architecture has taken the modularity of the package to new heights.

Template Code Generator

Allied to this, the mib2c template code generator utility has also been re-written – again to make it more modular. It now provides a selection of alternative configuration files to generate code aimed at the various different handlers, and new configurations can be added very easily.

Although the mib2c configuration files distributed with the software are all concerned with generating code to implement MIB modules within the agent, the new mib2c is actually a much more general and flexible tool. Given a suitable configuration file (written in what is essentially a scripting language), it can be used to generate arbitrary output, based on the contents of a portion of the MIB tree. This could include for example C code for an application, an XML data structure, or database SQL commands to create and manipulate data. Below is an example of how it might be used to analyze the tables within the HOST-RESOURCES-MIB and produce some descriptive English text:

```
$ cat > mib2c.test.conf << EOF
@open -@
@foreach $table table@
  @eval $column_count = 0@
  @foreach $c column@
    @eval $column_count = $column_count + 1@
  @end@
  @eval $index_count = 0@
  @foreach $index index@
    @eval $index_count = $index_count + 1@
  @end@
  Table $table at $table.objectID contains:
     $index_count indexes and $column_count columns
@end@

$ mib2c -c mib2c.test.conf host
writing to -
  Table hrStorageTable at .1.3.6.1.2.1.25.2.3 contains:
     1 indexes and 7 columns
  Table hrNetworkTable at .1.3.6.1.2.1.25.3.4 contains:
     1 indexes and 1 columns
  Table hrPartitionTable at .1.3.6.1.2.1.25.3.7 contains:
     2 indexes and 5 columns
...
```

New Agent Features

Turning back to the theme of extensibility within the Net-SNMP package, there have been a number of changes to improve the way that the Net-SNMP agent can co-operate with other agents. We have supported AgentX (as both master and sub-agent) for some time, and this support has continued to be improved and developed – particularly in the area of resilience. Net-SNMP's own trap receiver (snmptrapd) is now an AgentX sub-agent and implements the NOTIFICATION-LOG-MIB for archiving received notifications. The Net-SNMP agent now also supports multiple contexts – not just for AgentX, but for SNMPv3 as well. Community based request mapping to SNMPv3 contexts via the SNMP-COMMUNITY-MIB is planned for the future.

There have also been improvements to the proxy support, which passes requests on to other agents, possibly over different transports and even using different SNMP versions. Again, this has been included in the UCD-SNMP package for some time, but in a relatively limited form. In Net-SNMP 5.0, it is significantly more robust and can now make use of the context information to control where to pass the request on to. The new proxy code also takes advantage of the new easy-to-use, non-blocking API mechanisms within the new agent API suite, and thus will not block the agent while it is proxying a request.

Finally, the software also includes support for embedding Perl code directly within the agent to handle requests. MIB module handlers can now be written in either C or Perl. The Perl handler can even be loaded directly within the agent's configuration itself. For example, `perl print "hello world\n"` is a valid directive in the agent's configuration file, with the obvious result. This embedded Perl support is still extremely new and is expected to be fleshed-out in the future, but the basic architecture is now in place. The Perl API allows MIB handlers to be written either to be run directly within the agent, or as a separate process altogether and linked to the main agent through the AgentX sub-agent protocol.

The new release also includes changes to the Perl modules, providing improved access to the Net-SNMP library routines from within Perl programs. The oldest of the Perl modules ("SNMP"), will now read and parse snmp.conf configuration files allowing Perl scripts to make use of administratively defined configuration settings. The newer Perl modules are also somewhat more modular, and are grouped under the package name "Net-SNMP." This even includes a DBI-style wrapper module, providing access to SNMP management information via SQL queries. This comes with a "network shell" providing a number of useful additional features, such as aliasing, command definition, CSV and XML output etc.

One other addition to the agent, and something that has been missing for some time, is initial support for the DISMAN-EVENT-MIB from the Distributed Management working group. It has long been a source of confusion to people using the software that although the agent can be configured with various thresholds for "normal" behavior, it does not actually do anything when the activity being monitored strays outside these boundaries. Although the DISMAN-EVENT-MIB module is still very new and not all the features of it are supported, the agent can now take a much more active role in monitoring the system and can generate SNMP notifications when appropriate.

Software Organization

There have been other changes, including a reorganization of the header file structure, to avoid the irritating inconsistencies between coding within the main source tree, and using a fully-installed system. This also includes a number of wrapper header files, so application writers need only include two header files (or three if developing code for the agent).

There is a new tool (net-snmp-config) to help with identifying details of the configuration of an installed Net-SNMP suite. This shell script takes care of many previously annoying problems with compiling external applications and sub-agents. Makefiles for other packages can now just include a net-snmp-config invocation in order to get the appropriate list of libraries and header file paths to include in compilation rules. The net-snmp-config script will even compile a C-coded Net-SNMP agent handler module directly into an AgentX sub-agent in one invocation. Finally, it will perform other useful tasks like creating new SNMPv3 users – prompting for any necessary information.

And of course there have been innumerable bug-fixes and the like, as well as minor new features like long-form command line options. The code has also been generally cleaned up (indent has been run on the entire package to provide a vaguely consistent coding style) and more documentation has been written (much of it using the new adopted doxygen style of in-line-coding documentation). But this probably covers the main important changes and developments since the previous UCD-SNMP line releases.

Future Developments

And what about the future? Probably the main task on the horizon is a fuller review and re-design of the library – extending the modularization throughout the rest of the basic library APIs. This should aid those needing to extract individual portions of the library (e.g. to only include SNMPv3 support, or to omit MIB file handling when working in a resource-limited environment), as well as hopefully simplifying things for the general SNMP developer.

The UCD-SNMP-MIB is still supported, but experience has shown that at least some of the tables could benefit from a re-design. Any tables or groups that are rewritten will be moved into the new netSnmp enterprise branch of the MIB tree.

Then there are the moves afoot in the SMIng and EoS working groups looking at possible future developments for SNMP-based management, which are being actively followed by some of the Net-SNMP coders. The package is often a test-bed for new features of the SNMP framework, and we are hoping to be on the forefront of the EoS protocol implementations. If the timing is right, maybe the new Net-SNMP MIBs will be the some of first to make use of the new SMIv3 language.

And we have always had something of a bias toward Unix-based systems, and could desperately do with more input from those with experience of SNMP on Windows boxes. Although most elements of the package compile and run under Windows, with vaguely sensible results, the most active developers do not use Windows on a daily basis, and struggle to provide anything more than the most basic level of assistance.

It should be very clear that the new release is more than just a change of name, and the next few years show signs of being every bit as interesting as the last few have been. Why not come and join in?

Evaluating MIB II (RFC1213) Implementations

Henrik E. Holland, University of Twente
Remco van de Meent, University of Twente
Aiko Pras, University of Twente

Since the introduction of the Simple Network Management Protocol (SNMP) in the late nineteen eighties, SNMP and SNMP related network management has gone through substantial changes. The original version of the protocol issued as RFC 1157, has already engendered two successors: SNMPv2c and SNMPv3. The basic architecture of the SNMP concept has however remained fairly intact as have its basic functions. One of these basic functions is to provide network management systems with the capabilities to monitor their networks.

It goes without saying that the accuracy of measured quantities and retrieved data values within a network is of great importance to the network operator. If a network operator cannot rely on the accuracy of the network statistics provided to him by means of some network management scheme, this scheme will quickly be rejected. In this respect the vast popularity of SNMP based network monitoring might reflect the (overall) strong faith which the networking community has bestowed in the accuracy of SNMP based network statistics. As with any network management scheme however, the use of SNMP involves some risk. This is the risk of perceiving reported statistics as accurate when they are in fact erroneous. In July 1991, three months after the MIB-II was issued as RFC 1213, a group of AT&T researchers at Bell Labs presented the results of a series of SNMP agent tests performed on nine different SNMP supporting routers [1]. The results of the experiments were somewhat disappointing as they showed that the risk of receiving erroneous data values was by no means neglectable.

A decade later, while celebrating the tenth anniversary of the MIB-II, we decided to reinvestigate the accuracy and performance of SNMP based network monitoring. Within this perspective, we had grown curious about the results of ten years of SNMP development and implementation. The research was carried out as part of the Dutch Internet Next Generation project and was produced under deliverable D2.16 [2]. During our investigation we evaluated the SNMP capabilities of three devices configured as Internet Protocol (IP) routers. Our research was dominated by three concerns:

1. The accuracy of statistics collected by the SNMP agent and reported to the manager.

2. The speed at which these statistics are updated.

3. The performance of the SNMP entity while the router is handling a heavy traffic stream.

We believe that these three concerns address the most basic issues in relation to SNMP based network monitoring, and will in this respect provide at least a general insight into the current status of MIB-II implementations. Note that we concentrated on counters and that we did not look at set operations, the generation of notifications, correct get-next processing etc.

The rest of this article is divided into three sections. The first section describes which part (i.e. which set of objects) of the MIB-II was tested, how we engineered the test set-up and the tests themselves. A subsequent section presents the test results and a bit of result analysis. The conclusions of the research are presented in the last section of this article.

Approach of the Evaluation

This section consists of three sub-sections, which as a whole describe the approach of the evaluation. The first sub-section describes the tested MIB area, the second presents the test set-up. The third section finally describes specifically the three types of tests conducted on each of the three routers.

The Test Objects

The first question requiring an answer was the selection of the MIB-II test objects. This question is an important one since its outcome determines the exact test configuration (e.g. the required hardware) and the way in which the tests are run. In total, we selected thirty-six objects from five MIB groups:

```
system:                  ip:
  sysDescr                 ipInReceives
  sysObjectID              ipInHdrErrors
  sysUpTime                ipInAddrErrors
  sysContact               ipForwDatagrams
  sysName                  ipInUnknownProtos
  sysLocation              ipInDiscards
  sysServices              ipInReceives
                           ipInDelivers
interfaces:                ipOutRequests
  ifInOctets               ipOutDiscards
  ifInUcastPkts            ipOutNoRoutes
  ifInNUcastPkts           ipRoutingDiscards
  ifInDiscards
  ifInErrors             udp:
  ifInUnknownProtos        udpInDatagrams
  ifOutOctets              udpOutDatagrams
  ifOutUcastPkts
  ifOutNUcastPkts        snmp:
  ifOutDiscards            snmpInPkts
  ifOutErrors              snmpOutPkts
                           snmpInGetRequests
                           snmpOutGetResponses
```

The `interfaces` and `ip` group objects shown in the table above were selected because they belong to the set of MIB-II objects frequently used for SNMP based network monitoring. During the object selection, openly available vendor recommendations (such as [3]) considering the use of specific MIB-II objects were taken into account. Specifically included into the evaluation were objects responsible for delivering error related statistics such as cyclic redundancy check (CRC) error counters and IP header error counters. Finally, to achieve a complete overview of the MIB-II implementations, we included a few objects from the `udp` and `snmp` groups. All but the objects from the `system` group are of the `Counter` data type as defined in RFC 1155.

The Test Set-Up
To address our concerns accordingly, we devised a test configuration which allowed for an isolated evaluation of our devices. This excluded the possibility of network traffic interfering with our tests. The test set-up is shown in Figure 1.

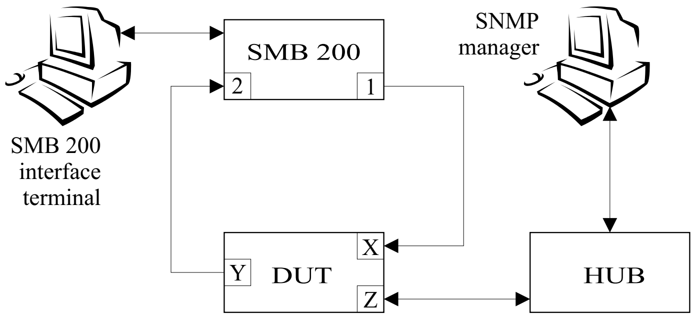

Figure 1: Test Set-Up.

All devices in the test-setup were connected with standard cat-5 cables. The devices unter test (DUTs) were the following:

- A: Cisco Systems AGS+, software: GS3-K Version 9.1(11), ©1986-1994.
- B: Cisco Systems C2600, software: IOS C2600-I-M Version 12.0(3)T3, ©1986-1999.
- C: Cabletron Systems ssr2000, software: Riverstone Networks Version 6.2.0.1, ©2000.

For the measurements, the following additional devices have been used:

- SMB200: Netcom Systems Smartbits 200 with two ML-7710 Ethernet smartcards.
- SNMP manager: Adventnet SNMP Utilities 3.0 on Windows NT.

From now on we shall refer to the routers as A, B and C respectively. Note that routers B and C are 100 Mbps devices while router A is a 10 Mbps device. This older 10 Mbps router was chosen for the evaluation to see if the SNMP capabilities of newer devices have improved.

Figure 1 shows that the device under test is connected to the Smartbits 200 over ports `x` and `y` and to the Adventnet SNMP manager via port `z`. The general idea was to use the Smartbits for generating and transmitting IP traffic, and to poll for MIB objects with the Adventnet manager. The Smartbits 200 is an industry standard transceiver with advanced network data generation and measurement capabilities [4]. Amongst other things, it allows the user to set up layer two and three traffic streams and it provides the user with frame editing capabilities. Furthermore, the Smartbits is capable of accurately counting and analyzing received bytes and frames. Tags can be added to IP packets before transmission to ensure that only packets transmitted by the Smartbits itself are counted at the receiving end.

As stated above, our test set-up was to enable an isolated evaluation of the test devices, so that no network or arbitrary background traffic could pollute our test results by unintentionally triggering the MIB-II counters. However, during the configuration of the test set-up, we found that router B was emitting some kind of spurious background traffic. Regrettably, we were unable to halt the transmission of these frames. Although the frequency of the frames was very low, it did cause some of the MIB object counters to increment. During our analysis, we canceled out the error introduced by the disturbing traffic by measuring its frequency and the size of the frames, and correcting the collected MIB object values accordingly.

Three Types of Tests
Within the scope of the evaluation, we have distinguished three kinds of test sets and henceforth three types of outcome spaces. The test sets and outcome spaces each correspond to one of the earlier defined concerns.

Accuracy of the Statistics
The set of tests devised to determine the accuracy of the statistics involved routing IP datagrams through the DUT and polling the MIB objects before and after the respective transmission.

The tests were executed by transmitting large packet bursts of different frame sizes at various transmission speeds. The exact content of the test streams was configured in accordance to the object(s) tested during that particular test run. In general, there were two kinds of test frames: non-erroneous Ethernet frames containing normal non-erroneous IP packets, and frames containing some kind of error where a distinction is made between an Ethernet frame error and an IP header or IP address error. The erroneous frames were constructed manually using the Smartbits' frame editor. In this respect, the Smartbits did limit our capabilities slightly. We were not able to put together a traffic burst consisting of an arbitrary amount of different Ethernet frames or IP packets. Instead the Smartbits allows one customized frame or packet to be transmitted along with the stream every `x` normal data frames, were `x` is a positive natural number.

As noted in the previous paragraph, the burst sizes of the test transmissions were very large. Depending on the transmission rate, the burst sizes varied between 5,000,000 and 10,000,000 packets. As does the correction for background traffic, transmitting such large amounts of packets will contribute to the accuracy of the test result. The unwanted incrementation of counter values due to the polling of the DUT's SNMP entity before and after the test becomes less of a problem, since the amount of SNMP packets belonging to the polls is relatively small.

The size of the test frames was constant during each individual test run, but varied per test run. Test runs of two different frame sizes were made on router A, and of three different frame sizes on router B and C. The frame sizes used were 64, 512, 1024 and 1518 bytes.

The rates of the test transmissions expressed as a percentage of the utilization rate, varied between 40% and 100% for routers A and C, and between 20% and 100% for router B. The utilization rate, here in respect to Ethernet, refers to the maximum theoretical transmission rate, which depends on the size of the transmitted frames.

Each device was tested at least at three transmission rates, where a distinction is made between two situations. In the first situation, the router is not dropping any of the offered packets while in the second situation it is. The last situation can occur when the device has to handle a large amount of traffic. Our intention was to test the devices in both situations. To do this precisely, we had to know at what rate the routers started dropping packets. The rate pertaining to this qualification is called throughput rate. According to RFC 1242, throughput is defined as follows:

Throughput: The maximum rate at which none of the offered frames are dropped by the device.

Before commencing with the actual evaluation, the throughput rate of the devices was determined using the Smartbits' benchmarking suite. These tests were conducted conform the benchmarking methodology as described in RFC 1944. Table 1 shows the results of these tests.

The results of the throughput tests not only provided a qualification necessary for the tests themselves, but they

router	throughput rate (pps)	theoretical max. throughput rate for Ethernet (pps)
A	10020	14880
B	25227	148810
C	147932	148810

Table 1: IP Throughput Rates (64 bytes per frame).

also revealed the "class" of the devices. B is obviously of lesser cachet than A and C, as far as throughput is concerned. These results are however not so awkward considering the difference in price between the devices.

Update of Counter Values

The second series of tests was conducted to verify the speed with which new MIB object values are made available. These tests were only run on objects of the `Counter` type. We used the same types of data streams as during the first test series, but now the MIB values were polled *during* the transmissions and not only before and after.

The objects were polled at a frequency of one poll per second. The difference between two successive values was plotted in a graph in real time during the test transmission. A continuous counter update within the DUT should therefore reveal a straight horizontal line representing the transmission rate in bytes or packets per second, depending on the polled object.

Of course there will be some fluctuation in the exact moment of the poll. Under normal circumstances however this fluctuation will never be large relative to the poll interval of one second. This test was not performed to measure very exactly certain values at certain moments and thus a small variation in the results is acceptable. Furthermore, by looking at the difference between successive values and not at the absolute values some of the (constant) delay is canceled out.

These tests were performed apart from the first test series described above, since we did not want the continuous polling of the MIB objects to interfere with the packet counts.

Performance Under Heavy Load

During the third series of tests, the response of the DUT's SNMP entity was tested while the DUT was dropping packets. The test verified if object values could still be delivered while the DUT was under strain and if so how many polls could successfully be made per second. In effect this means that the rate of the test stream was configured to be higher than the DUT's throughput rate. The SNMP entity was polled at various intervals during the tests. This poll interval was constant during a test run but it varied between the test runs. The IP test stream offered to the device consisted of 512 byte Ethernet frames at a transmission rate of up to 100% of the utilization rate.

Results

As described in the previous section, the evaluation consisted of three types of tests resulting in three sets of outcome spaces. Each of these outcome spaces will be treated separately in the following sections.

Accuracy of Counter Values

The accuracy tests in themselves were concluded successfully as we were able to obtain useful and significant test results. With a few exceptions, the accuracy of the octet and packet counts was high for all of the evaluated routers. All of the SNMP entities reported counter values which stayed within a margin of at least 1
% of the actual amount of offered octets or (respectively) packets.

The mentioned exceptions are related to Router C. The tests showed that the router's `ipInReceives` and `ipForwDatagrams` counters are troubled by a structural miscount of 2 packets for every counted packet. This means that for every incoming IP packet, three are counted.

In addition to the "triple count" error encountered at router C's SNMP entity, a number of specific peculiarities were observed, which concern routers A and B as well. The next sections will describe these peculiarities.

Packet Loss Count

During our tests, routers A and B dropped packets when the test stream was set to a higher rate than their respective throughput rate. The dropped packets, however, could not be accounted for at any of the counters belonging to the `interfaces` or `ip` groups.

There are two objects within the `interfaces` group and two within the `ip` group, which are related to the discard of packets not due to erroneous input but (amongst others) to a lack of buffer space:

- `ifInDiscards`
- `ifOutDiscards`
- `ipInDiscards`
- `ipOutDiscards`

When an IP packet is discarded, it should be counted at a counter pertaining to one of the above listed objects depending on where it is discarded. If the packet is discarded at IP level, then it must be counted at `ipInReceives` or `ipForwDatagrams` as well, depending on where it is discarded. To illustrate the above, Figure 2 shows a part of the `ip` group case diagram as depicted in [5]. Only the objects relevant for this example have been included, the dotted lines serve to indicate where objects have been left out.

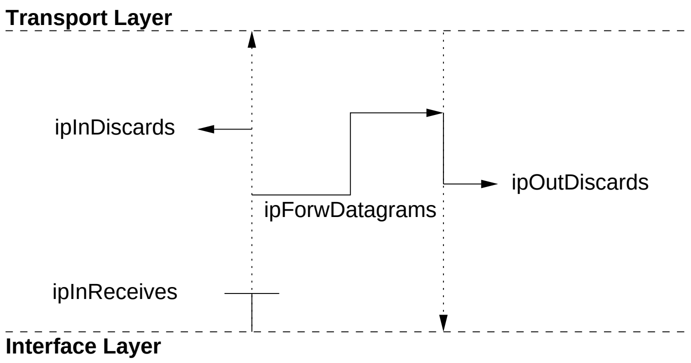

Figure 2: Excerpt from the IP Group Case Diagram.

During our tests, none of the discarded packets were counted at `ipInReceives`. This implies that the packets were not discarded at the IP level, apparently we must focus our attention on the lower interface layer.

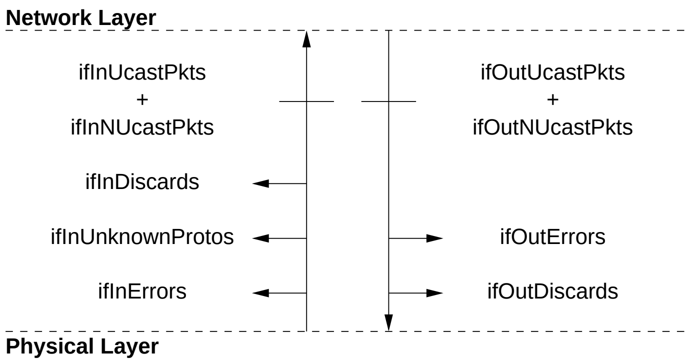

Figure 3: Interfaces Group Case Diagram.

Figure 3 taken from [5] shows the `interfaces` group case diagram. Remarkably enough, `ifInOctets` is not depicted in this case diagram. Curiously we took a look at [6] but found that the `interfaces` group case diagram presented in this book, also lacks the `ifInOctets` object. This seems at least awkward. RFC 1213 defines the `ifInOctets` object as follows:

> `ifInOctets`*: The total number of octets received on the interface, including framing characters.*

The definition implies that `ifInOctets` should count *all* of the octets received on the interface. In the case diagram, this would place the object just above the physical layer, and under the `ifInDiscards` object. This means that just as `ipInReceives` should count packets before they are discarded at the IP layer, `ifInOctets` should count all of the octets at the interface layer before being discarded. During our tests, however, this was not the case.

Apparently, the frames are dropped before reaching any of the counter mechanisms pertaining to the mentioned objects. Although this might be understandable, a network operator will want to see the amount of packets dropped by the devices that he manages. Besides this, the presence of the discard objects, defined as they are, bestows the network operator with the belief that he is able to monitor the amount of dropped packets, which is apparently not the case for routers A and B.

Router C did not drop a significant amount of packets during any of the test runs, since its throughput rate is almost equal to the utilization rate at all of the possible frame sizes. Henceforth, we could not study the administration of dropped frames for this device.

CRC Count

The evaluation revealed that routers A and B count incoming frame octets excluding the four byte Ethernet cyclic redundancy check. Router C however, does include the four byte CRC into its `ifInOctets` count.

The definition of `ifInOctets` suggests that the octet count should include the four byte CRC, since the count must include all of the octets received on the interface, including the framing characters. The implementation of this object in routers A and B therefore seems wrong.

Count of Packets With IP Header and Address Errors

During the evaluation we simulated four kinds of IP address errors. Two of these types (invalid and reserved Class E) should have lead to an increment of `ipInAddrErrors`, while the other two (not routable and multicast) were meant to test the `ipOutNoRoutes` object. We also tried various kinds of IP header errors to test the `ipInHdrErrors` object.

The IP header errors were handled and counted accurately at all of the right objects by all of the devices. Handling and counting IP packets with an invalid IP destination address proved to be somewhat more of a problem.

All of the tested routers accurately count the offered IP address error test packets at `ipInReceives`, which is a correct procedure. Just as well, all of the test packets are found to be erroneous by all of the routers and are henceforth discarded. What differs between the routers' behavior is the use of the `ipInAddrErrors` and `ipOutNoRoutes` objects. Router A discards the test packets, nonetheless neither `ipInAddrErrors` nor `ipOutNoRoutes` is incremented as it should. One type of address error was counted at `ipInUnknownProtos`, which is incorrect as well. Device B does a "better" job: it correctly counts two out of four test packets at `ipOutNoRoutes`. Router C does not count any of the test packets at respectively `ipInAddrErrors` or `ipOutNoRoutes`.

Count of CRC Erroneous Frames

Routers A and B count frames with an erroneous CRC value solely at `ifInErrors`. However, the definition of `ifInOctets` suggests that the erroneous frames should be counted at the `ifInOctets` counter as well. Routers A and B thus function incorrectly in this respect.

Router C also exhibits faulty behavior with respect to the count of the cyclic redundancy check characters. Router C counts the CRC erroneous frames at `ifInOctets`, `ifInErrors` and `ipInReceives` as well. The erroneous packets should however be discarded after being counted at `ifInErrors`, and should thus never reach the IP layer and its pertaining counters.

Wrap Around Errors

The `ifInOctets` and `ifOutOctets` object counters of router C are affected by wrap around errors. Our tests revealed that these errors occur more than occasionally. During six out of eight test runs, one or more of the mentioned interface objects proved to contain an erroneous counter value.

Our test results showed that the counters seem to "stick" for a while before proceeding with their count. The value at which the objects stick is the maximum value for a 32-bit counter. We have not determined exactly how long the counters stay at this value. It is possible that the wrap around errors are due to a counter type conflict. 32-bit counters are polled, while 64-bit counters are most probably in use on this device.

Counter Updates

Our tests showed that all of the routers' SNMP entities could provide actual statistics at a frequency of one Hertz. There was one exception which concerned router A and the `ifInErrors` object. The data value pertaining to this object changed 24 times every four minutes, which corresponds to one change per ten second. No other rarities were observed with any of the routers during any of the test runs. We may therefore conclude that all three routers could easily maintain their counters in a timely fashion.

test run	IP rate (pps) (%)	SNMP rate (pps)	SNMP loss (%)
1	5,000 (21)	10.0	0
2	10,000 (43)	2.5	72 ± 3
3	10,000 (43)	10.0	93 ± 3
4	20,000 (85)	10.0	94 ± 3

Table 2: SNMP Packet Loss under Heavy Load.

Object Retrieval Under Heavy Load

As described in a previous section, routers A and C are capable of realizing high throughput performance, whereas router B is much more limited. It seems that this difference reflects in the performance of the devices' SNMP entities when the device itself is under heavy load. Routers A and C do not have any problems with reporting requested object values while handling a large amount of traffic. However, router B shows a great performance degradation of its SNMP entity. Even when the rate of the IP test stream is below the device's throughput rate, the device is unable to report all of the requested object values.

Table 2 shows a list of test runs performed on router B during this series of tests. In column two, it shows the transmission rate of the IP test stream in packets per second (pps) and as a percentage of the utilization rate. The third column shows the amount of SNMP requests per second in packets per second (every IP packet contains one request). The last column contains the percentage of SNMP requests not replied to for each run. During runs one through three, the rate of the test stream was below the throughput rate of the router. The figures in the last column show that the device was unable to respond to all of the SNMP requests during runs two, three and four.

What we wanted to see is where exactly these SNMP requests are lost, i.e. where they appear in the packet counts and where they become absent. To illustrate this, Figure 4 shows a flow diagram containing relevant sections from the UDP and SNMP case diagrams.

The diagram is not meant to provide a complete overview of the respective case diagrams including all pertaining objects, but to illustrate where the SNMP packets should be counted and what the relation is between the counters. Printed beside the objects in Figure 4 are the results from test run two. The figure shows that somewhere between `udpInDatagrams` and `snmpInPkts` the requests "disappear" from the statistics, this area is encircled.

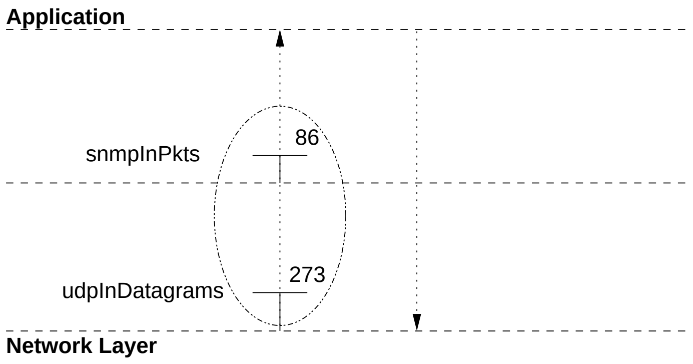

Figure 4: Example of the SNMP Packet Flow.

Conclusion

The evaluation described in the previous sections proved to be an interesting one. Our tests were dominated by three concerns: accuracy of the counters, speed of counter updates, and the performance of the SNMP entity under heavy load. These three concerns provided a satisfactory basis for the tests, and as a whole formed a suitable framework.

Our tests showed that the overall accuracy of the evaluated SNMP entities was good. Where packets were counted, this was done very precisely and consistently. There were however a number of errors concerning some basic objects. Most striking is the two packet miscount noticed at two of router C's `ip` group objects. This device also exhibits wrap around errors at two of its `interface` group objects. Interesting as well is that router C does count Ethernet CRC bytes, while routers A and B (incorrectly) do not. More inconsistency between the routers was observed regarding `ifInOctets` in case of CRC errors. A and B discard frames with CRC errors without counting them at this object while the definition suggests that this is wrong. All of the devices appeared to have problems with handling IP packets with an IP address error. Only device B administrates two out of four address error types correctly.

With the exception of one object at router A, all of the tested routers were able to update all counter values at least once per second.

Router B did not exactly pass the performance test of its SNMP entity with flying colors. Even when the router is not subjected to a high rate IP test stream, it loses the ability to respond to all of the received SNMP requests. The other devices show no problems of this kind. They maintain their ability to respond to a large amount of SNMP requests per second, under all conditions.

If we compare our results to those of the Bell Labs tests from 1991, an overall improvement in accuracy can be observed. Nevertheless, a number of serious errors were found. At the basis of some of these errors could be a simple inconsistency in viewpoints regarding exactly *how* some of the objects must be implemented, e.g. the CRC count discrepancy between the routers, and the miscount of CRC erroneous frames on routers A and B. The MIB-II RFC, however, is clear about these particular issues, and a sound implementation according to the standard seams feasible. What will help vendors with a correct interpretation of the RFCs, is the inclusion of case diagrams within these RFCs. This would unambiguously define how and where each object should be implemented.

Our overall impression is that the basic objects, regarding incoming and outgoing octets and IP packets, are implemented quite well, although this is a cautious and reserved statement, keeping the errors found at router C in mind. The implementation of more specific error objects, such as `ipInAddrErrors`, is done rather poorly on all of the devices. Based on the results, it seems fair to say that operators should always verify the statistics provided by a device before putting them into use.

References

[1] Mier, E., Bell Labs Test Routers' SNMP Agents, Network World 8(26), July 1991

[2] Holland, H.E., Evaluating MIB-2 Implementations, Deliverable D 2.16 of the Internet Next Generation Project

[3] Cisco Systems, Cisco Network Monitoring and Event Correlation Guidelines

[4] Spirent Communications, Smartbits 200, December 2001

[5] Rose, M.T., The Simple Book, An Introduction To Internet Management, 2nd Edition, Prentice-Hall, 1994

[6] Stallings, W., SNMP, SNMPv2, SNMPv3 and RMON 3rd Edition, Addison Wesley, 1999

The Story Behind the SNMP Vulnerabilities

Tiina Havana, OUSPG
Ari Takanen, OUSPG

In the PROTOS project, the Oulu University Secure Programming Group (OUSPG) and VTT Technical Research Center of Finland have developed and used software testing methods in a new application area: in the discovery of security problems in software. Since 1996, the researchers in Oulu have effectively been studying methods for pro-actively discovering security related bugs in software. During the past years, several security and reliability test-suites have been developed for various protocols by PROTOS researchers.

In spring 2002, the PROTOS project released a new test suite that received a lot of publicity. This time, the research group focused on SNMP. It was chosen to be the next test material after several flaws in various LDAP implementations were found using the PROTOS testing methods. Like LDAP, SNMP utilizes complex ASN.1/BER structures. Many of the discovered SNMP and LDAP flaws were caused by errors in decoder functions. Exceptionally formatted BER constructs, such as length fields claiming 2GB data to come, caused havoc amongst BER implementations. Different kind of errors were observed when correctly encoded SNMP packets containing exceptional data passed the decoding layer and proceeded to the application layer of a SNMP implementation, long community names triggering buffer overflows being a perfect example.

SNMP implementations are a good choice for trying new testing approaches in practice. SNMP is an old and mature protocol with numerous vendors providing solutions for it and even more numerous parties providing critical services over it. SNMP is also a complex protocol using an error prone ASN.1 encoding and containing various error prone data types over an unreliable network environment (UDP). Unlike most of the other testing groups who select a vendor and test the vendors products, OUSPG looks for interesting protocols for testing – and SNMP was one. Eventually, with the help of the testing tool developed in the PROTOS-project during the past couple of years it was possible to prove that SNMP implementations were vulnerable and could be exploited if wanted.

There be Bugs

In the SNMP-case about ten software products were tested by OUSPG researchers, but dozens of others were tested by the software developers themselves. The sample size of the products that OUSPG decided to test was intentionally limited. The programs that they chose for testing were selected according to their knowledge of programs that use SNMP and the availability of those programs. Products were not tested if no evaluation copy of the product was available, the evaluation copy had a restrictive license prohibiting evaluation or OUSPG simply was not aware of the product. OUSPG encouraged software developers and integrators to test their products by themselves.

Software bugs can never be fully avoided. Programs are complex entities, and, in the end, all men are fallible. However, a lot can be done to reduce the amount of bugs. Good testing systems are an essential help in this work. According to a research done by NIST, software testing is still more of an art than a science. The earlier bugs are found the cheaper they are to fix. The lack of historic tracking data and inadequate tools and testing methods usually limits the ability to obtain sufficient testing resources and to leverage these resources effectively.

In the SNMP case, by making the testing automated and practical, as opposed to an art form of manual test design, numerous software vendors were provided a chance of getting familiar with errors that traditionally had either been ignored or not been noticed in the first place. By being aware of the security implications and the risks involved with software development, it is possible to build up more precautions and preventive guidelines to assure the quality of the various SNMP implementations out there. As with testing always, the found problems are just samples and even increased testing coverage can never discover all errors. Software security vulnerabilities are a quality issue in the software implementations.

The testing tools to pro-actively probe for security problems in software are sometimes difficult to find, and the creation of such tests, if they are ever made, can be an extremely manual and time consuming task. The free PROTOS test-suites aim at rising the quality in software by providing a minimum set of tests that all protocol implementers have access to.

End of Story?

The discovery of a software vulnerability is followed by the vulnerability handling process. In the case of SNMP some of the software vendors were informed directly by OUSPG, but most of them got the report from CERT/CC.

Altogether over 130 reports were send to various vendors. Although many of them reacted to the report, it also was surprisingly common not to give any response at all. Receiving a bug report may cause an exceptional situation in the vendor company. It might happen that no one knows who is responsible for handling the issue or who one should and is allowed to inform about it. In the worst case the effect of not being prepared may be that the whole handling process is left incomplete and thus the vulnerability will not be repaired.

Vendors are often afraid of the potential negative publicity that may be related to the vulnerability and for this reason leave the reporter without any kind of non-automatic response. This is however often very shortsighted. The reporter may publish the information about vendors who have admitted their program to be vulnerable and promised to fix it, about vendors who have told that their software is not vulnerable – and about vendors who have not reacted. Often the most severe mistrust falls on these vendors.

In an ideal case, the reporting process of a software vulnerability goes as follows: when the vendor gets a vulnerability report the reporter is immediately informed that the report has been received and it will be evaluated. This response should be non-automatic. After the reporter has been informed the actual evaluation process takes part. This may last some time but it should produce information on whether or not the vulnerability actually exists in the vendor's products. After this has been done, the vendor should inform the reporter about the results of the evaluation. This makes it possible that the vendor and the reporter together decide the actions that are needed.

A company that aims to make high-quality products, and to be reliable, also takes reclamations seriously. Bug reports can be seen as one form of reclamation. They are complains about flaws in the products. Many companies have specific communication channels for reclamations, and everyone in the company is aware of the procedure. However, in the case of vulnerability reports things often do not proceed as smoothly.

The ultimate aim of the software vulnerability research and reporting especially in the academic field is to make the quality of the software better so that these kind of flaws could be avoided in the future. This requires, however, that the information management about the vulnerabilities has been arranged efficiently. The established practices of programming, software testing and communication about the quality issues must be analyzed critically from time to time. This is one of the most important lessons that can be learned from the SNMP vulnerability case and this is why the story is not in the end but means a beginning of a new era.

References

[1] Beizer, B., Software Testing Techniques, Thomson Computer Press, 1990

[2] Kaksonen, R., A Functional Method for Assessing Protocol Implementation Security, VTT Publications 447, Espoo. Technical Research Centre of Finland, 2001

[3] Laakso, M., Takanen, A., Röning, J., The Vulnerability Process: A Tiger Team Approach to Resolving Vulnerability Cases, Proc. 11th FIRST Conference on Computer Security Incident Handling and Response, Brisbane, 1999

Book Reviews

The reviews published in this column represent the opinion of the author(s). Please contact the author(s) directly if you want to share your comments. Please contact the editors of *The Simple Times* if you are interested to publish your own book review in this column.

Web-based Management of IP Networks and Systems

Reviewed by Frank Strauß, TU Braunschweig

This book is derived from the author's Ph.D. dissertation. Readers may realize this in two major aspects: The one who expects a primer that gives an introduction and a broad overview on network and systems management in general and on Web-based management in detail will probably have difficulties to understand this book. On the other hand, the reader who has a fundamental knowledge of traditional network management will regard this book as a valuable source of well reasoned pieces of essential information. You get the impression of a well organized and structured book from the very beginning when you catch a glimpse of the table of contents, the detailed index, and the extensive list of references. This impression gets confirmed when you start reading.

After a short introduction, a 15 pages chapter gives some clarifications on the relevant terminology, since there is some confusion in the network management community, for example with terms like gateway vs. proxy. The next chapter manifests the problem statement. It carefully describes the various technical and non-technical problems of todays real world of SNMP based network and systems management. The next two chapters contain an overview and analysis of the solution space to these problems, which are based on distributed management paradigms and the deployment of middleware concepts.

After 100 pages, the sixth chapter then gives a substantial overview of the state of the art in Web-based management: a wide range of concepts and research projects on browser-based management, N-tier management, HTTP-based approaches, XML-based management, and distributed Java-based management are presented.

Chapters 7 through 9 represent the core of the book. Here, Martin-Flatin proposes a new management architecture named WIMA (Web-based Integrated Management Architecture). This is motivated by the previously discussed problems of traditional network management and influenced by former approaches. It is well discussed how the organizational model and the communication model of WIMA deploy pull and push models for ad-hoc, regular and notification-driven management. It is described how Web technologies, MIME, and HTTP are applied and how XML is used to integrate SNMP MIB and CIM schema models instead of trying to establish yet another data model from scratch.

Chapter 10 gives a rough description of a WIMA prototype implementation named JAMAP. Since the code is not made available, this is not as helpful to the reader as it could be. However, it documents that WIMA has been practically evaluated and is more than just theory. The book concludes with a comparison of WIMA with other approaches like WBEM and JMX and with an outlook on future work on Web-based management and WIMA.

What I really like a lot about this book is its meticulous academic quality: The step-by-step documentation from a problem statement, a solution space analysis to a well reasoned proposed solution is very detailed. It mentions and explains a lot of related work that could be very helpful to the reader. Consequently this 338 pages book contains 300(!) references. The systematic structure of the book also helps to make it useful as a reference book, e.g. for citing benefits and drawbacks of technologies in network management like XML or SNMP. What I am missing in this book is a broader practical insight to the proposed WIMA architecture. Although there is the JAMAP implementation, the reader does not "get in touch" with it, which could help to get a better feeling for the potential of this Web-based integrated management approach. However, this book is a great source of carefully compiled information for each network management professional who is faced with taking the step from traditional network and systems management to Web-based management.

SNMP at the Edge

Reviewed by Jürgen Schönwälder, University of Osnabrück

This book is not the typical introductionary book about SNMP. This book is primarily about service management. SNMP only plays a role as an implementation tool in some of the chapters. The author, Jonathan Saperia, reasons that profitable service offerings require well designed, scalable and cost-effective service management systems, which are typically non-trivial pieces of software.

The 400 pages book is divided into five parts with a total of 14 chapters. The first part explains the importance of profitable value-added services such as virtual private network (VPN) services, the need for service management systems supporting such value-added services in a cost-effective manner, and the important role network edges play in service management. The network edges considered in this book are usually the boundaries between interfacing service providers or between providers and customers and may be realized by a complex aggregation of hardware and software systems that must work together in order to provide the value-added services to the customers.

The second part of the book starts by reviewing technologies for policy-based management that have been developed in recent years (CIM and PCIM, COPS and COPS-PR, SNMPCONF). Since the author has been one of the driving forces behind SNMPCONF, it comes as no surprise that the book somehow favors a technology that is based on the SNMP framework. The second part of the book also reviews some fundamental object-oriented principles and develops a very high-level generic model for network services.

Part three of the book discusses how SNMP technology in particular can be used to build effective service management systems. The first chapter in this part reviews basic SNMP principles and design decisions. It discusses relationships between MIB objects by looking at concrete object definitions from the OSPF-MIB, the IP-MIB and the IF-MIB. This part also provides an introduction to the technology developed by the SNMPCONF working group. Much of the material in this chapter is taken directly from the SNMPCONF specifications.

The design of management software on managed devices is the subject of part four. It is assumed that managed devices usually have multiple ways to access them. In this context, the approach with a common convergence layer with integrated access methods is compared to completely independent access layers and an argument is made that SNMP access methods can serve as the foundation for other access methods, such as command line interfaces. This part of the book also discusses how SNMPCONF can be used for service level monitoring and reporting.

Part five finally focuses on the construction of management software. A relatively traditional architecture is presented and some components such as the database component are discussed in some greater detail. This part presents some core classes for management systems in the form of UML diagrams and ends with a discussion of user interface issues and upstream interfaces.

The author of this book has been one of the driving forces behind the SNMPCONF working group. As such, one expects that this book is a solid explanation of the SNMPCONF technology and highlights its application with a series of convincing real-world service management examples. While reading the introduction of the SNMPCONF technology, I was somewhat disappointed since the content and presentation are almost the same as in the SNMPCONF specification. Using a different approach to present the conceptual models behind this technology could in my view have made this part of the book more accessible to the reader. (Having to go through almost 50 pages of MIB definitions with some small annotations is likely to bore most readers.)

Since the heart of the SNMPCONF technology is an embedded scripting language, I also expected to see some nifty scripts which demonstrate how this embedded language can be used to solve service management problems. But the text is almost silent about this aspect of SNMPCONF.

To summarize, the book is a great source of information for everyone involved in the development of service management software. The book also is valuable for those who are in general interested in learning about service management approaches and issues. Readers who want to learn more about the SNMPCONF technology and some of the motivations behind it will also find this book a valuable source of information. But readers should be reminded that it is useful to already have a solid understanding of the SNMP technology and object oriented concepts in order to fully understand the many details in the book.

Standards Summary

This section lists the SNMP related IETF specifications at the time of publication. Please consult the latest version of the Internet Official Protocol Standards `http://www.rfc-editor.org/rfcxx00.html` for more current information. The latest published version is RFC 3300.

SMIv1 Data Definition Language

Full Standards:

- RFC 1155 - Structure of Management Information
- RFC 1212 - Concise MIB Definitions

Informational:

- RFC 1215 - A Convention for Defining Traps

SMIv2 Data Definition Language

Full Standards:

- RFC 2578 - Structure of Management Information
- RFC 2579 - Textual Conventions
- RFC 2580 - Conformance Statements

SNMPv3 Protocol

Full Standards:

- RFC 3411 - Architecture for SNMP Frameworks
- RFC 3412 - Message Processing and Dispatching
- RFC 3413 - SNMP Applications
- RFC 3414 - User-based Security Model
- RFC 3415 - View-based Access Control Model
- RFC 3416 - Protocol Operations Version 2
- RFC 3417 - Transport Mappings for SNMP
- RFC 3418 - SNMP MIB

Proposed Standards:

- RFC 2576 - Coexistence between SNMP Versions

Informational:

- RFC 3410 - Internet Management Framework

Experimental:

- RFC 2786 - Diffie-Helman USM Key Management
- RFC 3430 - SNMP over TCP

SNMP Agent Extensibility

Draft Standards:

- RFC 2741 - AgentX Protocol Version 1
- RFC 2742 - AgentX MIB

SMIv1 MIB Modules

Full Standards:

- RFC 1213 - Management Information Base II
- RFC 1643 - Ethernet-Like Interface Types MIB

Draft Standards:

- RFC 1493 - Bridge MIB
- RFC 1559 - DECnet phase IV MIB

Proposed Standards:

- RFC 1285 - FDDI Interface Type (SMT 6.2) MIB
- RFC 1381 - X.25 LAPB MIB
- RFC 1382 - X.25 Packet Layer MIB
- RFC 1414 - Identification MIB
- RFC 1461 - X.25 Multiprotocol Interconnect MIB
- RFC 1471 - PPP Link Control Protocol MIB
- RFC 1472 - PPP Security Protocols MIB
- RFC 1473 - PPP IP NCP MIB
- RFC 1474 - PPP Bridge NCP MIB
- RFC 1512 - FDDI Interface Type (SMT 7.3) MIB
- RFC 1513 - RMON Token Ring Extensions MIB
- RFC 1525 - Source Routing Bridge MIB
- RFC 1742 - AppleTalk MIB

SMIv2 MIB Modules

Full Standards:

- RFC 2819 - Remote Network Monitoring MIB
- RFC 3411 - SNMP Framework MIB
- RFC 3412 - SNMPv3 MPD MIB
- RFC 3413 - SNMP Applications MIBs
- RFC 3414 - SNMPv3 USM MIB

- RFC 3415 - SNMP VACM MIB
- RFC 3418 - SNMP MIB

Draft Standards:

- RFC 1657 - BGP version 4 MIB
- RFC 1658 - Character Device MIB
- RFC 1659 - RS-232 Interface Type MIB
- RFC 1660 - Parallel Printer Interface Type MIB
- RFC 1694 - SMDS Interface Type MIB
- RFC 1724 - RIP version 2 MIB
- RFC 1748 - IEEE 802.5 Interface Type MIB
- RFC 1850 - OSPF version 2 MIB
- RFC 2115 - Frame Relay DTE Interface Type MIB
- RFC 2742 - AgentX MIB
- RFC 2790 - Host Resources MIB
- RFC 2863 - Interfaces Group MIB

Proposed Standards:

- RFC 1666 - SNA NAU MIB
- RFC 1696 - Modem MIB
- RFC 1697 - RDBMS MIB
- RFC 1747 - SNA Data Link Control MIB
- RFC 1749 - 802.5 Station Source Routing MIB
- RFC 1759 - Printer MIB
- RFC 2006 - Internet Protocol Mobility MIB
- RFC 2011 - Internet Protocol MIB
- RFC 2012 - Transmission Control Protocol MIB
- RFC 2013 - User Datagram Protocol MIB
- RFC 2020 - IEEE 802.12 Interfaces MIB
- RFC 2021 - RMON Version 2 MIB
- RFC 2024 - Data Link Switching MIB
- RFC 2051 - APPC MIB
- RFC 2096 - IP Forwarding Table MIB
- RFC 2108 - IEEE 802.3 Repeater MIB
- RFC 2127 - ISDN MIB
- RFC 2128 - Dial Control MIB
- RFC 2206 - Resource Reservation Protocol MIB
- RFC 2213 - Integrated Services MIB
- RFC 2214 - Guaranteed Service MIB
- RFC 2232 - Dependent LU Requester MIB
- RFC 2238 - High Performance Routing MIB
- RFC 2266 - IEEE 802.12 Repeater MIB
- RFC 2287 - System-Level Application Mgmt MIB
- RFC 2320 - Classical IP and ARP over ATM MIB
- RFC 2417 - Multicast over UNI 3.0/3.1 / ATM MIB
- RFC 2452 - IPv6 UDP MIB
- RFC 2454 - IPv6 TCP MIB
- RFC 2455 - APPN MIB
- RFC 2456 - APPN Trap MIB
- RFC 2457 - APPN Extended Border Node MIB
- RFC 2465 - IPv6 Textual Conventions and MIB
- RFC 2466 - ICMPv6 MIB
- RFC 2493 - 15 Minute Performance History TCs
- RFC 2494 - DS0, DS0 Bundle Interface Type MIB
- RFC 2495 - DS1, E1, DS2, E2 Interface Type MIB
- RFC 2496 - DS3/E3 Interface Type MIB
- RFC 2512 - Accounting MIB for ATM Networks
- RFC 2513 - Accounting Control MIB
- RFC 2514 - ATM Textual Conventions and OIDs
- RFC 2515 - ATM MIB
- RFC 2558 - SONET/SDH Interface Type MIB
- RFC 2561 - TN3270E MIB
- RFC 2562 - TN3270E Response Time MIB
- RFC 2564 - Application Management MIB
- RFC 2576 - SNMP Community MIB
- RFC 2584 - APPN/HPR in IP Networks
- RFC 2594 - WWW Services MIB

- RFC 2605 - Directory Server MIB
- RFC 2613 - RMON for Switched Networks MIB
- RFC 2618 - RADIUS Authentication Client MIB
- RFC 2619 - RADIUS Authentication Server MIB
- RFC 2667 - IP Tunnel MIB
- RFC 2662 - ADSL MIB
- RFC 2665 - Ethernet-Like Interface Types MIB
- RFC 2668 - IEEE 802.3 MAU MIB
- RFC 2669 - DOCSIS Cable Device MIB
- RFC 2670 - DOCSIS RF Interface MIB
- RFC 2677 - Next Hop Resolution Protocol MIB
- RFC 2720 - Traffic Flow Measurement Meter MIB
- RFC 2737 - Entity MIB
- RFC 2787 - Virtual Router Redundancy Proto. MIB
- RFC 2788 - Network Services Monitoring MIB
- RFC 2789 - Mail Monitoring MIB
- RFC 2873 - Fibre Channel Fabric Element MIB
- RFC 2856 - High Capacity Data Type TCs
- RFC 2864 - Interfaces Group Inverted Stack MIB
- RFC 2895 - RMON Protocol Identifier
- RFC 2925 - Ping, Traceroute, Lookup MIBs
- RFC 2932 - IPv4 Multicast Routing MIB
- RFC 2933 - IGMP MIB
- RFC 2940 - COPS Client MIB
- RFC 2954 - Frame Relay Service MIB
- RFC 2955 - Frame Relay / ATM PVC MIB
- RFC 2959 - Real-Time Transport Protocol MIB
- RFC 2981 - Event MIB
- RFC 2982 - Expression MIB
- RFC 3014 - Notification Log MIB
- RFC 3019 - Multicast Listener Discovery MIB
- RFC 3020 - Frame Relay UNI/NNI Multilink MIB
- RFC 3055 - PSTN/Internet Interworking MIB
- RFC 3083 - DOCSIS Baseline Privacy Interface MIB
- RFC 3144 - RMON Interface Monitoring MIB
- RFC 3165 - Scripting MIB
- RFC 3201 - Circuit Interface MIB
- RFC 3202 - Frame Relay Service Level MIB
- RFC 3231 - Scheduling MIB
- RFC 3273 - RMON High Capacity MIB
- RFC 3276 - HDSL2 / SHDSL Line MIB
- RFC 3291 - Internet Network Address TCs
- RFC 3287 - RMON Differentiated Services MIB
- RFC 3289 - DiffServ MIB
- RFC 3295 - General Switch Mgmt Protocol MIB
- RFC 3371 - Layer Two Tunneling Protocol MIB
- RFC 3395 - RMON Protocol Identifier Extensions
- RFC 3419 - Transport Address TCs
- RFC 3433 - Entity Sensor MIB
- RFC 3434 - RMON High Capacity Alarms MIB
- RFC 3440 - ADSL Extension MIB

Informational:

- RFC 1628 - Uninterruptible Power Supply MIB
- RFC 2620 - RADIUS Accounting Client MIB
- RFC 2621 - RADIUS Accounting Server MIB
- RFC 2666 - Ethernet Chip Set Identifiers
- RFC 2707 - Print Job Monitoring MIB
- RFC 2896 - RMON Protocol Identifier Macros
- RFC 2922 - Physical Topology MIB

Experimental:

- RFC 2758 - SLA Performance Monitoring MIB
- RFC 2786 - Diffie-Helman USM Key MIB
- RFC 2934 - IPv4 PIM MIB

IANA Maintained MIB Modules

The Internet Assigned Numbers Authority (IANA) maintains several MIB modules. The IANA MIB repository is located at ftp://ftp.iana.org/assignments/.

- Interface Type Textual Convention (ianaiftype.mib)
- Address Family Numbers Textual Convention (ianaaddressfamilynumbers.mib)
- TN3270E Textual Conventions (ianatn3270etc.mib)
- Language Identifiers (ianalanguage.mib)
- IP Routing Protocol Textual Conventions (ianaiprouteprotocol.mib)

Related Documents

Informational:

- RFC 1270 - SNMP Communication Services
- RFC 1321 - MD5 Message-Digest Algorithm
- RFC 1470 - Network Management Tool Catalog
- RFC 2039 - Applicability of Standard MIBs to WWW Server Management
- RFC 2962 - SNMP Application Level Gateway for Payload Address Translation
- RFC 2975 - Introduction to Accounting Management
- RFC 3052 - Service Management Architectures Issues and Review
- RFC 3198 - Terminology for Policy-Based Management
- RFC 3216 - SMIng Objectives
- RFC 3387 - Considerations on IP Quality of Service

Experimental:

- RFC 1187 - Bulk Table Retrieval with the SNMP
- RFC 1224 - Techniques for Managing Asynchronously Generated Alerts
- RFC 1238 - CLNS MIB
- RFC 1592 - SNMP Distributed Program Interface
- RFC 1792 - TCP/IPX Connection MIB Specification
- RFC 3139 - Requirements for Configuration Management of IP-based Networks
- RFC 3179 - Script MIB Extensibility Protocol 1.1

Recent Publications

SNMP at the Edge

- Author: Jonathan Saperia `<saperia@jdscons.com>`
- Publisher: McGraw-Hill Publishing `http://books.mcgraw-hill.com/`
- ISBN: 0-07-139689-6
- Available: June, 2002

This book focuses on offering and managing value added profitable services at the edge of a network. Integrated service management systems must include support for service creation, provisioning, management and billing. Although the author favors SNMP to explain the problem, the book is much more general in the way it discusses service management issues. (For more information, see the book review elsewhere in this issue.)

Web-based Management of IP Networks and Systems

- Authors: Jean-Philippe Martin-Flatin `<jp.martin-flatin@ieee.org>`
- Publisher: Wiley `http://www.wiley.com/`
- ISBN: 0-471-48702-3
- Available: September, 2002

Starting from a detailed analysis of the problems with existing network management architectures, the book develops a new distributed management architecture called WIMA (Web-based Integrated Management Architecture), which is based on Web technologies such as HTTP and XML. The book also describes a research prototype implementation of this new architecture which is called JAMAP. (For more information, see the book review elsewhere in this issue.)

Calendar and Announcements

IETF Meetings:

- 56th Meeting of the IETF
  March 16-21, 2003, San Francisco, CA, USA
- 57th Meeting of the IETF
  July 13-18, 2003, Vienna, Austria
- 58th Meeting of the IETF
  November 9-14, 2003, Minneapolis, MN, USA

Conferences and Workshops:

- International Symposium on Integrated Network Management 2003 (IM 2003)
  March 24-28, 2003, Colorado Springs, CO, USA
- Workshop on IP Operations and Management 2003 (IPOM 2003)
  October 1-3, 2003, Kansas City, MO, USA
- Workshop on Distributed Systems Operations and Management 2003 (DSOM 2003)
  October 20-22, 2003, Heidelberg, Germany
- Large Installation System Administration Conference 2003 (LISA 2003)
  October 26-31, 2003, San Diego, CA, USA
- Network Operations and Management Symposium 2004 (NOMS 2004)
  April 19-23, 2004, Seoul, Korea

Exhibitions and Trade Shows:

- NetWorld + Interop Las Vegas
  April 27- May 2, 2003, Las Vegas, USA

Publication Information

Editors	
Aiko Pras	University of Twente
Jürgen Schönwälder	University of Osnabrück
Editorial Board	
David Harrington	Enterasys Networks
Keith McCloghrie	Cisco Systems Inc.
Bob Natale	Lucent Technologies
David Perkins	SNMPinfo
Randy Presuhn	
Steve Waldbusser	
Bert Wijnen	Lucent Technologies
Contact Information	
E-mail	`st-editorial@simple-times.org`
ISSN	1060–6068

Submissions

The Simple Times solicits high-quality articles of technology and comment. Technical articles are refereed to ensure that the content is marketing-free. By definition, commentaries reflect opinion and, as such, are reviewed only to the extent required to ensure commonly-accepted publication norms.

The Simple Times also solicits terse announcements of products and services, publications, and events. These contributions are reviewed only to the extent required to ensure commonly-accepted publication norms.

Submissions are accepted only via electronic mail, and must be formatted in HTML version 1.0. Each submission must include the author's full name, title, affiliation, postal and electronic mail addresses, telephone, and fax numbers. Note that by initiating this process, the submitting party agrees to place the contribution into the public domain.

Subscriptions

The Simple Times is available in HTML, PDF and PostScript. New issues are announced via an electronic mailing list. Send electronic mail to

```
st-request@simple-times.org
```

with

```
subscribe simple-times
```

in the body if you want to subscribe to this list. Back issues are available via *The Simple Times* Web server:

```
http://www.simple-times.org/
```