The Simple Times™

THE QUARTERLY NEWSLETTER OF SNMP TECHNOLOGY, COMMENT, AND EVENTS℠
VOLUME 3, NUMBER 1 FEBRUARY, 1994

The Simple Times is an openly-available publication devoted to the promotion of the Simple Network Management Protocol (SNMP). In each issue, ***The Simple Times*** presents: a refereed technical article, an industry comment, and several featured columns. In addition, some issues include brief announcements, summaries of recent publications, and an activities calendar. For information on submissions, see page 12.

In this Issue:

Technical Article

Aiko Pras and Jacques Togtema
Twente University of Technology

In this issue: *SNMPv2 at Twente University*

The management group at Twente University in the Netherlands is currently developing SNMPv2 software. The purpose of this article is to provide an overview of this development and give future plans. It is not the intention to go into to much detail — the last section of this article tells how to obtain more detailed information.

Background

The last couple of years the network management group of our university has participated in a number of European RACE and Esprit projects. We worked on management architectures and, as most Europeans, concentrated on OSI and TMN.

After some time however it became clear to us that we needed more experience to judge the merits of the various architectural concepts. This resulted in a decision to start our own implementation project. Given the growing importance of Internet management and the different "atmosphere" of the Internet world, we decided to start implementing the complete SNMPv2 Framework (RFCs 1441–1452). Since we weren't involved in the definition of SNMPv2, our work would reveal whether SNMPv2 is sufficiently defined to allow implementation by non-adepts.

Goals

Our first goal was to learn through experience. To achieve this goal, we decided to discuss and implement all aspects of SNMPv2 ourselves. For example, we implemented the Basic Encoding Rules from scratch, although we could have started from one of the existing SNMPv1 packages.

Our second goal is to share our ideas with the community. We therefore make our software freely available and invite others to comment.

It should be noted that we're not working on a commercial product. If, for example, we have to choose between clarity and performance, we choose clarity.

Features

The two main differences between our implementation and those of others (e.g., CMU and 4BSD/ISODE), is our multi-process structure and our high-level programming interface (API). We will discuss each of them in a separate section.

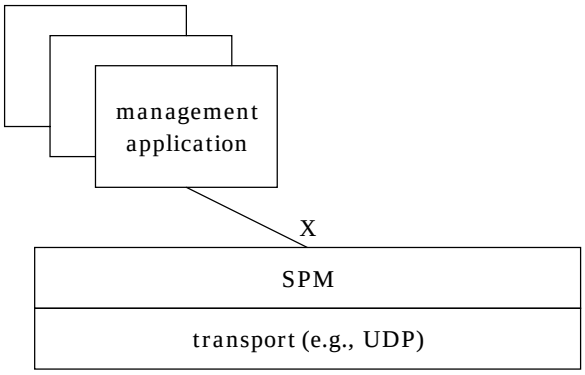
Multi-process Structure

Our software development takes place on SUN Sparcstations running UNIX (SunOS 4.1 and Solaris 2) using the GNU *C* compiler. Although we can't really test our software on other UNIX systems, we try to keep our software portable.

As UNIX allows multiple processes to cooperate, we decided to use this facility and develop a structure in which a single process, called the SNMPv2 Protocol Machine (SPM), serves multiple management applications.

The SPM is responsible for the transfer of SNMPv2 management information between systems and has no knowledge of management application issues. The SPM is therefore not bothered with MIB issues and things like Textual Conventions. The attractive property of our structure is that the SPM can be the same for the manager and agent side: it is the management application that determines whether a system acts in a manager or agent role.

Modification of this role is easy, since other applications (playing different roles) can be connected to the same SPM. Development of dual-role intermediate-level managers (e.g., to support the Manager-to-Manager MIB), is therefore straightforward.



Communication between SPM and management applications uses interprocess communication, such as TLI and sockets. As such, it is even possible to run management applications on different machines. Of course there is a performance penalty in having this multi-process structure with IPC. We believe however that this performance penalty is sufficiently compensated by the improved flexibility and the lower complexity.

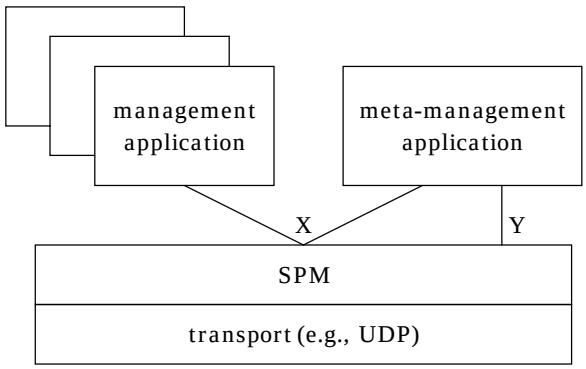
As the first management application becomes active, the SPM is automatically started. The SPM initializes by reading a configuration file and is then ready to serve the application. If other management applications become active, they will be connected to the same SPM. The SPM remains active until all management applications have terminated. Upon termination, the SPM saves a copy of the recent configuration information to disk.

The SPM is implemented in a single-threaded fashion, although we originally considered a multi-threaded approach. The advantage of a multi-threaded approach is that messages can be processed in different threads independent from each other. It guarantees that large messages that require authentication and encryption will not delay the processing of small messages that do not have security requirements. However, after an investigation of DES and MD5 processing times and after we understood the complexity associated with multi-threaded design, we decided to use a single-threaded approach.

Implications

The decision to have a multi-process structure and to separate management transfer functions from management application functions, has important implications. Let's discuss two of them.

The first implication is that management of SNMPv2 itself (meta-management) should not be performed by the SPM, but by special (meta-management) applications. As a consequence, the SPM is not bothered with issues such as maintaining the Party, SNMPv2 and Manager-to-Manager MIBs. This is equivalent to the approach followed with other protocols, such as IP and TCP. Implementors of IP and TCP are not bothered with MIB issues either; it is sufficient if they provide a local interface that allows reading and writing of IP and TCP variables by a special management process. It is the task of this special process to transform local information into the form required by the MIB-II and check the validity of the management operations.



To illustrate the functioning of our meta-management application (MMA), consider the example of changing a party's authentication clock. To change this clock, a `set` PDU must be received by the MMA via the same IPC port (e.g., X) as used for other application exchanges. The MMA (and not the SPM!) implements the rules (as specified by the Party-MIB) for changing clocks: the MMA therefore checks whether the received PDU also changes the authentication key. If this is the case, the MMA uses the local interface port (e.g., Y) to perform the actual clock change.

A second implication is that proxy relationships will not be performed by the SPM, but by special proxy applications. This makes the design of proxy agents straightforward: it is sufficient to understand the management API, it is not necessary to know the details of SNMPv2.

Application Programming Interface

The main difference between our implementation and other implementations (e.g., CMU and 4BSD/ISODE) is the programming interface. As opposed to other implementations, our API hides most of the complexity of SNMPv2 from the management applications. Writers of management applications need little knowledge of SNMPv2, which makes development of applications easier.

Our API is actually a library of C-functions, which must be included in every management application. Details of the IPC mechanism are handled within the API and are therefore not visible to the writer of management applications.

The API functions can be divided into two categories: functions necessary for initialization and termination purposes and functions necessary for sending and receiving SNMPv2 PDUs. To initialize, the application calls the `snmpOpen` function. This function starts, if necessary, the SPM and connects the application to it. To terminate, the application calls the `snmpClose` function.

This function removes the connection with the SPM — if no other applications are connected to the SPM, the SPM will terminate too.

Most of the API functions are used for sending and receiving SNMPv2 PDUs. Most "service elements" (e.g., `get`) require four function calls: two for sending (Request and Response), and two for receiving (Indication and Confirm). The Response and Confirm function calls are necessary for sending and receiving Response PDUs. Although it is possible to use the same Response and Confirm functions for all 'service elements', we decided (primarily for clarity reasons) to introduce separate function calls for each individual "service element".

Since the application can not know in advance to which "service element" a received PDU will belong, the application precedes each Indication and Confirm call with a `snmpLook` call. This call tells the application the type of service element that has been put into the IPC queue by the SPM and must therefore be handled first.

The table below shows all API calls that can be used to send and receive SNMPv2 PDUs. Note that because the SPM translates, at the agent's side, a `get-bulk` into a number of `get-next` calls, there are no Indication and Response calls for `get-bulk`. The table also shows that the SPM, upon receipt of an `inform` PDU from another SPM, automatically generates the `response` PDU.

	Req	Ind	Res	Con
get	x	x	x	x
get-next	x	x	x	x
get-bulk	x			x
set	x	x	x	x
inform	x	x		x
trap	x	x		

Of course, associated with each function call are a number of parameters, including:

- the IP address of remote system;
- the list of variable bindings;
- the request ID;
- the requested security, i.e., "none", "auth" (MD5 is used), or "priv" (MD5 and DES is used); and,
- a string that helps to determine the context, which may be empty.

Status

The first version of our software was released in November 1993. The major part of this software was written by a single student as part of his M.Sc. thesis. This first release should be considered as a "statement of direction" — although it can be used to manage SNMPv2 systems, it is not yet complete, e.g., there are no meta-management applications, so it is not yet possible to modify the Party, SNMPv2 and Manager-to-Manager MIBs.

Experience

At the beginning of our project we assumed SNMPv2 would be simple and easy to implement. This assumption proved to be wrong. SNMPv2 is complex for the following reasons:

The demands put on SNMPv2 are much stronger than those put on SNMPv1. This inherent complexity appears for instance from the fact that 12 RFCs were needed to specify SNMPv2.

SNMPv2 has been specified as a monolithic whole — little attempt has been made to decompose SNMPv2 into a number of building blocks with clearly defined interfaces. We believe that a decomposition of SNMPv2 into transfer, application, meta-management and proxy-oriented parts (along the lines of our approach) would have made SNMPv2 easier to understand.

The RFCs describe various mechanisms into great depth, but they hardly explain their purpose and the way they should be used (this is particularly true for SNMPv2's administrative model). It would for instance be helpful to see a description of how to search through the party, context, access control, and view table to determine the party and context identities that must be included with the PDU.

Still no real problems were encountered. Interoperability testing against the CMU package went smoothly. We may therefore conclude that it is possible for non-adepts to implement SNMPv2.

Future

After our first release demonstrated that it was possible to develop SNMPv2 software, we formed a project team to continue this development for at least another year. Our new team is much bigger than the one we had last year, so we expect to complete all aspects of the SNMPv2 framework, including meta-management, this year. Our plan is to make new versions available on a regular basis (e.g., every three months) Our sources can be obtained via anonymous FTP from `ftp.cs.utwente.nl` in the directory `pub/src/snmp`. Our email address is: `snmp@cs.utwente.nl`. Further information, including all project documentation, is made available via our WWW server:

`http://snmp.cs.utwente.nl:8001/snmp/html/homepage.html`

Industry Comment

Marshall T. Rose

Welcome to the third year of ***The Simple Times***.

With this issue, we're moving to a quarterly distribution cycle. The reason is simple: the coordinating editor simply doesn't have enough time to put together six issues each year. So, we're going to try four issues a year.

As a consequence of this, the *Working Group Synopsis* column is discontinued. Although Fred Baker, Deidre Kostick, and Kaj Tesink have done a wonderful job with each issue, the column will lose too much of its value with a longer distribution cycle. Fortunately, each of these contributors has promised to write a technical article later on for ***The Simple Times***!

Applications and Directions

Steven L. Waldbusser

In this issue: *Deploying SNMPv2*

As people have been tracking the progress of SNMPv2 deployment they have wondered about where the bottlenecks in the deployment process are, what is driving the process, and where products will materialize most quickly. Of course, this is a chicken and egg problem, so the answer to the question "Will the deployment be led by managers or agents?" is largely "Yes!". In an attempt to be a bit more accurate than this, we will compare the current scenario with the deployment of SNMPv1, and then make some observations about the current situation and predictions of the future.

SNMP (version 1) Deployment

In 1988, as the first SNMP specifications were finished, there were two interesting things about the industry: there were very few network management applications or protocols available, and much of the networking infrastructure (gateways and servers) was built on UNIX platforms. The first point created an incredible vacuum that was filled by SNMP. The second meant that a few free and commercial UNIX products easily built a critical mass of deployed products. This was followed fairly quickly by many embedded (i.e., non-UNIX) agents as vendors realized it was quite inexpensive to add an agent to a product and provide another check-off item for customers who had been loudly demanding network management capabilities.

Applications and platforms of varying sophistication came later, and when vendors realized the complexity of building a platform, many scaled back to only providing