Plan 9 SNMP Filesystem

Rodolfo García Peñas
kix@kix.es

November 11, 2006

ABSTRACT

Abstract. Network management is based on several protocols that allow us to know the state of devices in use. To do so, we need to use a wide set of management tools. These tools are complex, expensive, they facilities often depends on the vendor and usually they arent compatible each other. If we put in practice the new concepts developed in the OS Plan 9 we can access to network devices as if they were files and folders. The OS has been built in a way that it is allowed. This gives the chance to applications to be independent from network protocols and allow the management and maintenance of network devices straight from OS tools. It also allows programming in the language the user desire, included scripting languages, and get information of network devices remotely from a mobile phone or a web navigator.

1. Introduction

To do network management tasks several protocols are used CMIP [RFC-1189] and SNMP [RFC-1157]. The last one is the most commonly used and the file system is going to be concerned about it.

Simple Network Management Protocol (SNMP) allows getting and modifying parameters from the devices plugged into the network, such as commuters, routers, firewalls, servers, etc. This make possible to know and control the state of the different parts of the network.

1.1. Architecture

SNMP protocol is composed of a client-server architecture where the client is any network node (often called managed system) and the server is a manager station (often called SNMP manager).

Managed device has a piece of software (known as agent) that keeps a list of variables. Those variables keep the desired information about the device such as contact name, device name, device load, interfaces information, etc. This information is stored in a Management Information dataBase (MIB). There are two running modes, Manager-Agent and Agent-Manager.

1.1.1. Management-Agent Mode

In this mode the manager station ask information to its managed agents. The agents get that requests and search the value on the MIB and return it to the manager station. This is the most common used mode and has several possible messages, some of the most common are:

1. Get: Ask for a leaf on the MIB.
2. GetNext: Ask next leaf on the MIB.
3. SetRequest: Set the value of a MIB leaf.

This mode allows the manager station to ask and process the necessary information. On this way is possible to get load plots and more general knowledge of the network devices states.

1.1.2. Agent-Manager Mode

Manager-agent mode is very powerful but is unable to locate problems in the very moment they are happening, such as a network interface down. This mode has been designed to solve this problem.

When a concrete situation happens in a node that contains an SNMP agent, this agent can send a Trap message to its manager. In this manner the manager station can know the troubles that its managed devices have in the very moment they are happening.

1.2. MIB

The MIB is the hierarchy database that manages agents information and can be consulted and modified by the manager station.

Its hierarchical structure is depicted as an object tree where each object has a unique ID (OID). For instance, the element sysContact, which has node contact information has the value *1.3.6.1.2.1.1.4* as OID. This is the sum of the *iso[1]*, *org[3]*, *dod[6]*, *internet[1]*, *mgmt.mib-2[1]* and *system.sysContact[4]* branches.

Each branch is subrogated to an organization or organism, and each sub-branch can be subrogated to another organization. It is possible to find into the tree corporate branches which belong to enterprise organizations such as Cisco or HP. Those branches allow managing nodes variables that have a proprietary character.

Each one of MIB objects has several fields such us OID, object type (*integer*, *string*, *element list*, etc.), object size, access type (*read*, *read-write*), etc.

1.3. Versions and Security

With the development of the RFC-1157 [RFC-1157] the first version of SNMP was developed. In it there was a description of the architecture and design of the MIB and the communications protocol. The basic functions were defined and were running using the PDUs GetRequest, GetNextRequest, Trap, GetResponse, and SetRequest.

Regarding security, SNMPv1 use a password to read objects and another to write them in the agent. They are called read community and write community. This password use is due to the managed devices usually belong to the same network and they constitute a community altogether. The SNMP information travel unencrypted over the network, including communities.

With the development of second version with the RFC-1902 [RFC-1902] two new PDU are added, GetBulk and Inform and several new types are added to the protocol semantic. There were no advances regarding security.

With the third version SNMPv3 several security elements were added. Messages could be authenticated by users, encrypted authentication with SHA and MD5 are available and the communication can be encrypted.

While in other protocols the development of a new version block the use of the previous ones in SNMP the versions fist and second still in use and the third version is hardly used.

2. SNMPFS

Given that the aim of this article is apply Plan 9 concept on network management Ill develop a filesystem design. I'll follow the following steps:

1. Filesystem design.
2. Design and implementation of P9-SNMP.
3. Integration.

2.1. File system design

The file system design allows us to know how it will be presented to users and how users will be able to interact with it.

2.1.1. Mounting

Mount is the action that users do to tell to the OS that a file system should be added to the local file system. On Windows OS a network folder unit or FTP server can be added. On UNIX OSs a folder is added to the local file system. Once the file system has been added (mounted) it is possible to work on it as if it were a local folder structure.

Plan 9 works like UNIX do, mounting remote file systems into the local one. It is also possible add systems which dont support natively the protocol 9P by using servers (such as SNMPFS) that make the data interchange between protocols.

There are two ways to mount file systems on Plan 9. Depending on how the implementation is we can use:

Direct mounting When you run the server command, the mount point is communicated as a parameter. For instance, `ftpfs m /n/myfolder 10.0.0.1` will mount into the folder `/n/myfolder` the FTP service offered by `10.0.0.1` device.

Server mounting It needs two separate stages. The first one runs the server command. For instance, `ftfps`. It adds the service into the `/srv` folder. The second stage allow mounting the remote server. To do so the command `mount /srv/ftpfs /n/myftpfolder 10.0.0.1` will be ran and the same result as previous paragraph is get.

To use snmpfs Ive selected the first option indicating the mounting point and the remote server. For instance: `snmpfs −m /n/snmpnode www.snmp.org`

2.1.2. Users presentation

Users presentation is how users will perceive the file system, including files and directories, once this will have been added to the local file system.

Due to the tree arrangement of SNMP variables the most suitable option is to mount a file system where tree branches will be folders and leaves (where the values are stored) will be files. This system would allow represent as text branches and variables. For instance, reach the branch `1.2` would be possible by running either `cd 1/2` or `cd iso/org`.

After several test and analysis we found some troubles that demonstrate the inefficiency of this system:

- It is necessary to keep a complete SNMP tree on memory for each system managed.
- Normally not every SNMP variable is checked, the common ones are those related with interfaces, load, etc.
- Navigation through a folder tree that extent is complex and makes device management inefficient. Nevertheless it can be interesting from a didactical point of view.

We found a new solution without those troubles. Its characteristics are the following:

- Given the small number of variables to manage a unique folder (the mounting point root) will be use and all files will be stored there.
- The file name can be the OID (f.i `1.3.6.2.1.1.4`), the whole contextual path (f.i `iso.org.dod.internet.mgmt.ietf.mib2.system.sysContact`) or an alias (f.i `sysContact`).
- The file would give permission only to the user who mounted the file system. These permissions will determined if the file is read or write, *400* and *600* respectively. The group and the other users wont be allowed (in this first version).
- Files wont be created or erased from the folder. The reason is that OIDs are not created or destroyed. They are used just to consult or change their values.
- The files that will appear on the folder will be just those which would be indicated on the settings.

2.1.3. Configuration Settings

An SNMP request has a set of parameters, they are commonly:

Type GET, GETNEXT, GETBULK, INFORM, SET, RESPONSE y TRAP

Version Version 1, 2 and 3. The version 2 is usually 2c.

Communities The read and read-write communities.

OID lists The OID list requested or to change.

Host The node to which the request is asked.

Port The remote port, usually 161.

Timeout Max waiting time.

Retries Default number of retries.

All of those variables must be set somewhere; in addition they can be different on different systems. Plan 9 network settings file is `/lib/ndb/local`. On it network devices, DNS configuration and other settings are set up. Values are set in pairs following the structure `variable=value`. For instance, out coming mail server is indicated as `smtp=10.0.0.17`.

Previous variables must be represented in a settings file and can have default values. The IDs and its default values are shown in Table 1.

Element	Identity	Default
Packet type	-	GET / SET
Port	snmpport	snmp (161)
Version	snmpver	1
Read comm.	snmprocom	public
Write comm.	snmprwcom	private
Timeout	snmptimeout	3
Retries	snmpretries	3
OIDs	snmpoid	-

Table 1: SNMP configuration options

Following values are special and arent specified on the settings file with the same syntax:

Type Request type is simple `GET` to read and `SET` to write. It wont be necessary to be specified on settings file because it will be automatically set by the server depending on if it is writing or reading the OID file.

OID list It will be indicated as oid and should be specified in settings file. OIDs will be separated by commas (`,`).

Host The SNMP settings will be indicated under host line on settings file.

Using these options is possible to build a settings file to manage the devices of a network. Figure 1 shows an example for one node.

```
ip=192.168.1.69 sys=pepino  dom=pepino.9grid.es
    snmpver=1
    snmpport=snmp
    snmprocom=public
    snmprwcom=private
    snmptimeout=3
    snmpretries=2
    snmpoid=1.3.6.1.2.1.1.1.0,1.3.6.1.2.1.1.2.0
```

Figure 1: SNMP Settings

2.2. Plan 9 SNMP

One of the main difficulties that We have addressed on this article is that Plan 9 did not have SNMP support. The only try has been written by Russ Cox to manage some variables from some devices of Harvard University. Nevertheless it used just some types of messages for version 1. It extremely limitated their functionality. We decided to write the whole client SNMP code from the beginning.

Next trouble is that SNMP use ASN.1 to codify messages. Russ solves it by using several little function tailored specifically. ASN.1 uses a system of three fields. The first one indicates the data type, integer for instance. Second one indicates first value long, two octets for instance. Lastly the third one indicates the value. Figure 2 is an SNMP package capture and shows the different fields. Data type is depicted in red (characters string in octets). String long is depicted in blue (11 on hexadecimal, 17 on decimal). Finally the value is depicted on green.

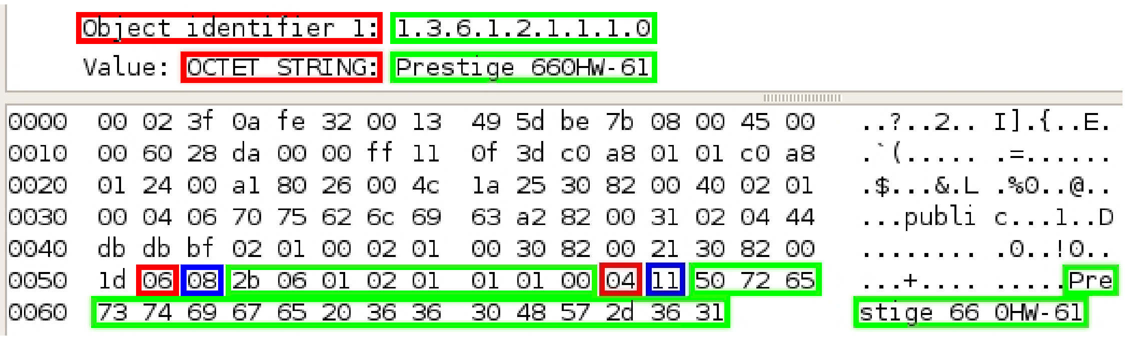

Figure 2: ASN.1 Structure

It wasnt a problem to treat variables independently. However, when we nest them a recursive calculation of their longs was needed. Typical values (integer, string, etc) were simple to manage. Though, those relative to SNMP made the processing complex. On Figure 2 you can see that coloured octets come after two type 30 octets. These octets indicate a string composed of several subtypes. Behind those octets is the value of the total long, `00 21` and `00 1d` respectively which stand for the sum of the two coloured objects with they types and they longs. Therefore, the only way to do the codification and decodification of messages was using ASN.1.

Instead of develop an ASN.1 library, a time and effort consuming task, We find a Plan 9 library which offer ASN.1 treatment. This allows us to focus on getting an SNMP code. The X.509 library offer certified treatment.

The main troubles to face with this library were that not every ASN.1 object used on SNMP was supported and that there was no documentation. This last trouble has consumed lot of time before We have been able to use the library. At the same time We have expand the library to add up the types that were missing.

Once codification problems have been solved We have created the basic data structures on SMTP, several structures of code aid, functions of data transformation to ASN.1 and vice versa (codification and decodification), and a structure to keep SNMP session information.

2.2.1. SNMP Session

SNMP protocol is based on sending and receiving of UDP data. Thus there is no need to use sessions (understanding by session the connection establishment, sending of data and end of connection). Nevertheless, there are several parameters used in the establishment of communication that are required and reused constantly such as, remote host, port, communities, timeouts, retrys, etc. Therefore, in several SNMP implementations the concept of session is used and all data generated by the manager entity and device managed during the data transmission are stored in the same structure.

In this implementation We have created one data structure to store information and two functions. One of them to generate a session, (memory request and data start up). And the other to end up the session, (free memory).

2.2.2. SNMP Data

SNMP protocol sends and receives using UDP packages. These packages have the origin and destiny addresses (sockets). And they have embedded as data the SNMP information which includes a field to store the version. Another field used to store the community. And one more field used to store the PDU.

There are eight types of PDUs previously related, (Get, Set, GetNext, Inform, Trap, etc.) Each one has its own structure although all of them are very similar. Each one includes a field used as PDU type indicator, a field use as request indicator, two fields use as error records and finally a list of pair name/value where name is the OID and value is the value that the OID has. In case of request the field will be send empty and the answer package will have the value. In case of writing the value send will be written on the remote device.

We used this scheme to do the implementation. The data contained in UDP header will be stored on the session structure with the host and the destination port. We have included the community and version values into the session structure albeit they belong to the package because they are common in every package.

The SNMP header will be stored into the package structure. This structure contains three fields containing the version, the community and the PDU. Even when version and community fields are also into the session structure are needed here to be used on the codification process. Send packages dont use them.

The PDU field is formed by the values pdu type, pdu id, error registry, error id, number of retries, max of retries and long. The last three values are used on snmpBulkRequest packages. The UDP will have a list of pares name/value that will be encapsulated in a structure called Tuple.

Figure 3 shows how data structures are linked. This shows how the tupla list that a PDU can contain is formed. The package list is used solely to link packages of the same request. For instance the packaged requested by a SNMP GetNext. In that way they can be managed as a unique element (packages list.)

Ive created the previously referred structures and functions to create and erase each one of them. The relations (in red and blue on the picture) have been implemented as pointers.

2.2.3. SNMP to ASN.1 conversions

The last link is data process. While package related information is stored on the previously mentioned data structures is necessary send and receive data using ASN.1. To do so We have created two separate conversion processes. The first one convert data from SNMP to ASN.1 and the second one does the inverted process.

2.2.4. SNMP Applications

After the development of those functions Ive develop several SNMP applications common on most OSs. These applications are the commands snmpget; snmpgetnext; snmp-walk; snmpbulk

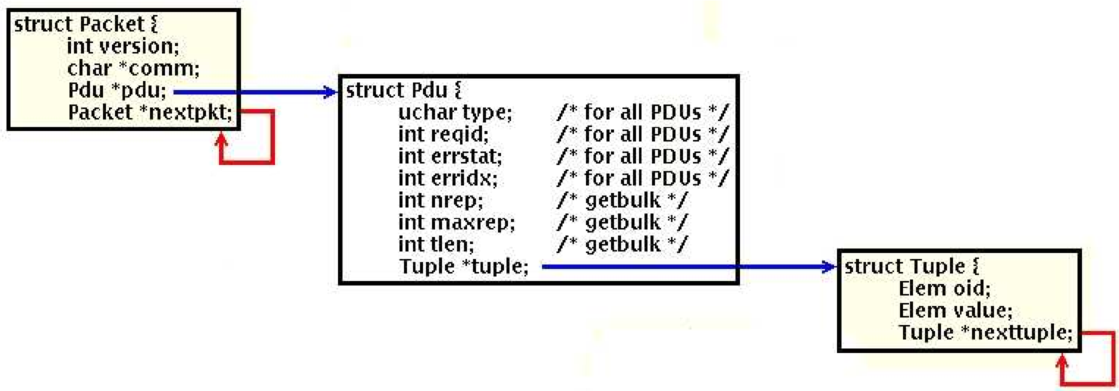

Figure 3: SNMP Structures

and snmpset. In addition Ive created a little took called snmptool that allow users to create packages with the fields that each user desire.

Every of that constitute an SNMP client implementation over Plan 9.

2.3. P9 SNMP and SNMPFS Integration

Once the file system design and a common set of applications are available is the moment to integrate them together.

SNMP consultation and modification applications do the next steps:

1. A session object is created. Every variable needed to do the query will be stored there.
2. Creation of as many packages and PDUs as the query need.
3. Sending of query request including data conversion from SNMP to ASN.1.
4. Answer reception including inverted conversion from ASN.1 to SNMP.
5. Erasing of packages and PDUs structures.
6. Screen printing of data.
7. Erasing of session structure.

The file system needs to do as well the next steps:

1. Mount the file system. In that process the root folder and OID files will be create.
2. Wait for file read and write requests.
3. Unmount the file system and erase files and folder.

These three phases will be used to integrate SNMP applications and file system.

2.3.1. File System Mounting

On the phase of file system creation (mounting), the settings file will be read and the session structure will be created. This session will last until the file system will be unmounted. This is possible because session information doesnt change over a reasonable period of time. For instance community or SNMP version doesnt need to change.

When the settings file will be read the information related to the OIDs to be used will be extracted. In that moment the OID files will be created on the folder. I consider that creation of OID files once the system is mounted is no needed.

The permission will be those specified during file system design phase.

2.3.2. File System Read and Writes

At this point a directory that represents an SNMP node with several files representing each one an OID of that node is available.

When a user runs a file content read command, for instance a cat-like command, the file system will run an smmpget typed request. Then it will show the received information on the screen as an answer. To do that request it will use the file name as OID and it will convert if that name is an alias. It will use the information stored into the session structure as well. (It wont use the settings file information.)

The information returned by the destiny node will be shown on the screen. It will work similarly to typical SNMP commands that print the requested OID an equal sign (=) and the OID value.

The writing of values into the files, for instance "`echo value > file`", will generate a request snmpset-like. This request will use file name as OID and values stored on session structure.

In this phase the packages, PDUs and all the information needed to consult a remote node will be created, the data will be sent and the returns will be processed. The information will be printed on the screen and data structures will be erased. The file system will end as it begins (same used memory and same data structures.)

SNMP request are not stores on the file system. Therefore no additional main memory is consumed.

2.3.3. File System Unmounting

At this point the user will unmount the file system. (Normally using umount in UNIX, or unmount in Plan 9 commands.)

When the unmounting is running the files and folder will be erased and the memory used to store the session data structure will be free.

3. Applications

To show up the advantages of the new file system Weve done an implementation that shows each different case that will be studied on this section.

The easiest case is direct access to the file. In that case users can access to file contents using OS commands. This prevents the need to know how SNMP protocol and management applications work. Figure 4 example show how using a cat command users can get contact people (`sysContact`) from several different systems. The command output can be used, for instance to search users on a database.

```
cpu% lc
kix.es          router
oliva.9grid.es  snmp-dot-org
cpu% cat */sysContact
"Rodolfo Garcia"
"Gabriel Diaz"
"Gabriel Diaz Lopez de la Llave"
"SNMP.org Management Group"
cpu%
```

Figure 4: SNMP SysContacts

Next case is the automation of the previous command. For instance, using a loop to store in a record log file variables from one or several files. That record can be used to analyse the evolution of any desired variable on a certain period of time.

If you need to modify same variable on great number of managed devices, you will be able to do it with a little script to fill setting files contents. For instance you can change a contact name on all devices over the network using a little script just three or four lines long. Taking the file system of Figure 4 it would be as easy as run the following command "`echo System Contact People > */sysContact`."

From automation we can extract new capabilities, such as statistic generation. By using returns of numeric objects it would be possible to know the evolution of several values (CPU load, processed traffic, etc.) It can be get throughout time or when some events happen (picks hours, etc.) Those kind of scripts could evaluate file contents and change the appropriate parameters when some levels are reached. It is also possible use the stats utility to automatically generate real time statistics and read them on the system panel. An example is shown on Figure 5.

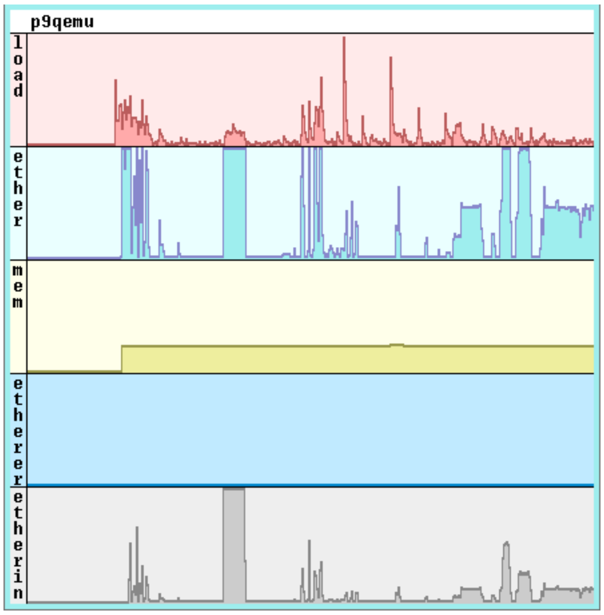

Figure 5: Stats

Working with files is possible to do a back up copy of the settings stored on a given time. The tar command can be straightfully used to that end. The contents of a file can be consulted and changed anytime.

A more powerful generation of utilities can be developed if variables are exported used 9P, NFS or SMB. This would allow to several systems and users to work with same variables and each one can use the functions it needs.

This can be empowering vividly by using a web server to export the necessary variables. The server can offer the values as a list of files. And even better, it can format the values making real time plots and sending that information remotely.

The scalability is warranted and any change on protocols wont affect the applications. For instance a chance on the IP version will affect the file system; however it will be completely transparent to applications. In addition several versions of SNMP protocol can coexist at the same time and application wont know it. It is possible even change the SNMP protocol version on one device without modify applications.

Another powerful tool is that applications can do the same requests to all devices. For instance, a device with SNMPv1 which receives a GetBulk-typed request (PDU) will automatically refuse it because it is unsupported on this version. However, SNMPFS could include (as it is into the development) an automatic substitution with several Get-typed requests and return same information to the application.

This model, based on layers (file system and applications,) rises the cohesion to the top and eliminate overlapping.

4. Conclusions

After develop this article and the file system design and implementation Ive been able to get several useful conclusions:

- Implementing file systems to do a great amount of functions allow standard accesses to applications. This make the information interchange easier, diminish the need of user formation and therefore save costs.

- The applications development is simplified shifting a great amount of complexity to file tem. The same is applicable to language programming that become independent from the OS. It allows using standard languages instead of those proprietary.
- Unsupported functions can be allowed by the file system. Following snmpfs example functions from v2 and v3 are available on v1 through the file system like GetBulk. This makes the application easier.
- The use of file systems allows using several incoming protocols to be use with an application that doesnt has a specific support. For instance, if snmpfs would support CMIP and SNMP it could send CMIP information as SNMP to applications. The conversion would be done by the file system and there would be no need to applications to know a new protocol.
- The file system allows common functions to be done by the applications. This eliminates the need to have lots of libraries to do the same functions..
- Security is raised due to the only one that manage communities is the file system.
- Granularity is achieved. Each user can reach only certain branches of the SNMP tree..
- As you can see, protocols abstraction can be used on saving code lines, unify efforts, add up transparency and save cost.

References

[P9OR-01] Dave Presotto, Phil Winterbottom, "The Organization of Networks in Plan 9," Computing Science Research Center, Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey.

[P9OR-01] Plan 9 Programmer's Manual Volume 2. Third Edition, 2000. ISBN 0-9538702-8-9

[ELPC-01] Brian W. Kerninghan, Dennis M. Ritchie, "El lenguaje de programación C," Pearson Educación. Segunda Edición. ISBN: 968-880-205-0

[PIDI-01] Ángel López, Alejandro Novo, "Protocolos de Internet, diseño e implementación en sistemas UNIX," RA-MA Editorial. Primera edición. ISBN 84-7897-382-6 (1999)

[BELL-01] Bell Labs. `http://www.bell-labs.com/`

[P9W-01] Plan 9 from Bell-labs. `http://plan9.bell-labs.com/`

[SUNM-01] Sun Microsystems. `http://www.sun.com/`

[RFC-1155] M. Rose, Performance Systems International, K. McCloghrie, Hughes LAN Systems, "Structure and Identification of Management Information for TCP/IP-based Internets," May 1990

[RFC-1156] K. McCloghrie, Hughes LAN Systems., M. Rose, Performance Systems International, "Management Information Base for Network Management of TCP/IP-based Internets," May 1990

[RFC-1157] Case, SNMP Research, M. Fedor, M. SchoffStall, Performance Systems International, J. Davin, MIT Laboratory of Computer Science, "A Simple Network Management Protocol (SNMP)," May 1990

[RFC-1189] U. Warrier, Netlabs, L. Besaw, Hewlett-Packard, L. LaBarre, The Mitre Corporation, B. Handspicker, Digital Equipment Corporation, "The Common Management Information Services and Protocols for the Internet (CMOT and CMIP)," October 1990.

[RFC-1902] J. Case, SNMP Research, Inc., K. McCloghrie, Cisco Systems, Inc. M. Rose, Dover Beach Consulting, Inc., S. Waldbusser, International Network Services "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)" January 1996.

[KILL-84] Tom M. Killian "Processes as files," USENIX Summer Conference 84. Pages 203-207. 1984.