

An SNMP Filesystem in Userspace

Rui Pedro Lopes^{1,2}, Tiago Pedrosa^{1,2}, and Luis Pires¹

¹ Polytechnic Institute of Braganca, Braganca, Portugal

rlopes@ipb.pt

² IEETA, University of Aveiro, Aveiro, Portugal

Abstract. Modern computer networks are constantly increasing in size and complexity. Despite this, data networks are a critical factor for the success of many organizations. Monitoring their health and operation status is fundamental, and usually performed through specific network management architectures, developed and standardized in the last decades.

On the other hand, file systems have become one of the best well known paradigms of human-computer interaction, and have been around since early days in the personal computer industry.

In this paper we propose a file system interface to network management information, allowing users to open, edit and visualize network and systems operation information.

Key words: Network Management, SNMP, File system

1 INTRODUCTION

A file system is a database typically storing large blocks of information. The information is stored in the form of files, structured as a hierarchy of directories. Each entry in the file system, including directories and files, is characterized by a limited group of attributes (instant of creation, instant of the last access, instant of the last change, permissions, owner, group). This paradigm is, perhaps, one of the best well known mechanism for storing information, available in several operating systems as well as in some embedded devices, such as PDAs, mobile phones and even digital cameras.

Usually, files are stored locally in persistent memory, such as hard-disks or memory cards. It is also common, particularly in enterprises, to use network file systems to store files in a remote server, accessed through a special protocol like NFS [1] or SMB. In this situation, the network server exports part of the local file system to a set of selected clients, allowing them to remotely access files and directories. However, this centralized, single server approach, suffers some scalability issues related to throughput, capacity and fault tolerance.

Distributed file systems are designed to improve scalability and fault tolerance by transparently balancing the access between servers. It provides the same view to every client and is responsible for maintaining coherence of data through distributed locking and caching [2, 3, 4].

Yet another paradigm, cloud storage systems, such as Dropbox¹, transparently synchronize the local copy of data with a remote datacenter, allowing user access to personal and shared files anytime, anywhere.

Based on this paradigm, we considered the possibility of accessing network management information as a set of virtual file systems. Network resources, usually accessed through SNMP [5], COPS [6], or other network management protocol, are seen as remote shares, to be mounted in a regular workstation file system and the instrumentation and configuration information accessed as regular files.

The nature of the information as well as the purpose of this *SNMP File System* (SNMPFS) is radically different from traditional distributed file systems (DFS). This makes the current requirements different of traditional DFS. First, distributed file systems are used to store files belonging to one or more users. In a network management scenario, the information is generated both by network resources and users. The former is used for instrumentation and the later is used for configuration.

Second, the content of the majority of network management backed files is constantly changing, because of the dynamic nature of network parameters. As an example, consider a value representing the number of transmitted packets or a value representing CPU load. In regular file systems, used to store personal and application files, entries seldom change. This allows better cache hit rates than in the former situation.

Third, network management files are very small, resulting from the parameters they represent. Frequently, a single `int` is used and, some times, a small table of values is sufficient.

Fourth, the whole SNMPFS is composed of several network services and resources, such as routers, firewalls, web servers, and so on. Each resource exports the information resulting from the instrumentation of working parameters thus playing a part in a potentially huge cluster of distributed file systems.

Fifth, faults are usually frequent, resulting from connectivity problems, hardware failures, human action and others. The system should cope with this issues, by recovering when possible and replicating when necessary.

2 FILE SYSTEM DESIGN

The goal of the SNMPFS is to unify around a unique name space all of the enterprise network management agents. For concept proving, we are using SNMP agents, since they are common in organizations and widely implemented by network devices. This approach allows integrating the tree of management objects of distinct SNMP agents in a single file system. Resuming, the goals of this approach are the following:

- allow the use of simple files and directories handling tools (`cd`, `cp`, `cat`, ...);
- allow to consult and change the values through simple tools (file redirection, text editors);
- allow the creation of pipes:


```
cat sysUpTime | toxml | mail --s ``sysUpTime`` admin@host.com;
```

¹ <http://www.dropbox.com>

- allow using office tools, such as Calc or Excel, to view and alter tabular information;
- allow reducing the complexity of the management system.

The information generated by each SNMP agent is accessed through a specific file system (SNMPFS), working as a gateway between the regular file operations (open, read, write, append, close, ...) and SNMP commands (Figure 1).

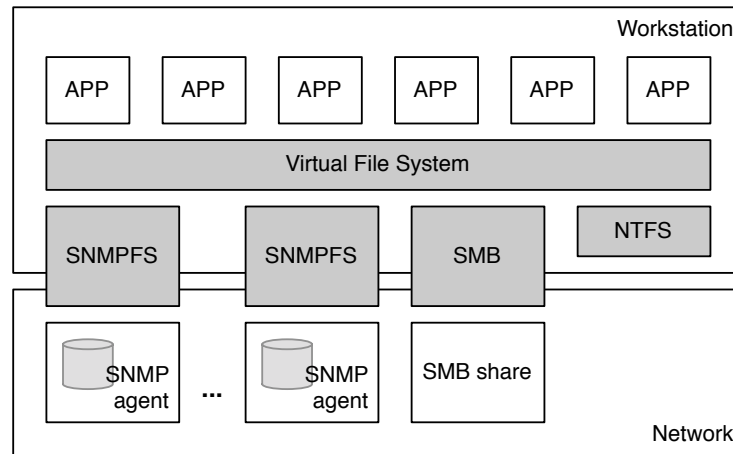


Fig. 1: SNMPFS global architecture.

As other file systems, like Server Message Block (SMB), also known as Common Internet File System (CIFS), to access SMB (Windows) shares across a network or NTFS for local storage, applications access data through a uniform layer (VFS - Virtual File System). Specific file system details are of the responsibility of each file system technology (SNMPFS, SMB, NTFS and so on). As a result, all the information is stored under the same naming tree.

2.1 Topology

The system topology is straightforward: on one side, several SNMP agents, associated with diverse enterprise resources; on the other side, one or more workstations mount the agents' information through the SNMPFS (Figure 2). It is possible that two or more workstations mount the same agents in its local file system. For read operations this does not present any problem. However, for update operations it is possible that potentially many processes try to change the same value.

2.2 Locking

In conventional distributed file systems, file locking is essential for coordinating access to shared information among cooperating processes. If multiple processes are writing to

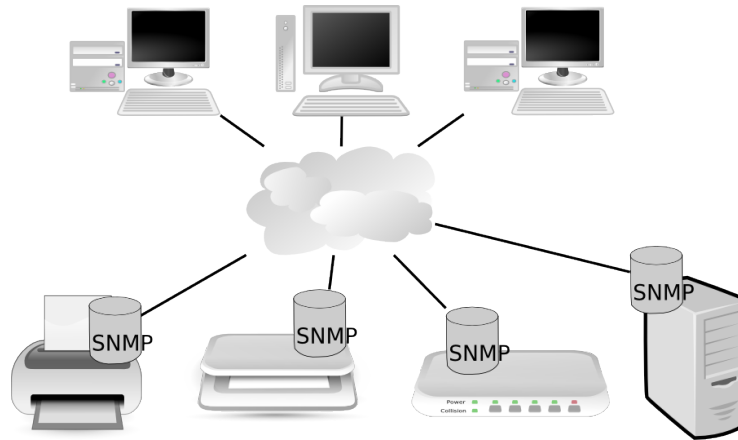


Fig. 2: Simplified topology.

the same file it is necessary to regulate the access through some kind of locking mechanism. In *SNMPFS*, locking is performed by the agent, in accordance with the managed object definition, since SNMP agents may also be updated by network management applications concurrently. Some MIBs provide a mechanism to regulate concurrent access. The Expression MIB [7], for example, has tables with a special column used to instantiate the row – *RowStatus* [8].

2.3 Security

SNMP is inherently insecure. Although true for the versions 1 and 2c, SNMPv3 presents a modular security architecture based on cryptographic protocols and algorithms. It is mandatory that SNMPv3 implementations support the HMAC-MD5-96 protocol for authentication. They can also support the HMAC-SHA-96 for authentication and the CBC-DES for privacy [9]. More recently, a new privacy protocol was added. [10] describes the Advanced Encryption Standard (AES) for SNMPv3 in the SNMP User-based Security Model which can be used as an alternative to the CBC-DES.

The SNMPv3 security service provides data integrity, data origin authentication, data confidentiality and message timeliness as well as limited replay protection. It is based on the concept of a user, identified by a *userName*, with which security information is associated. In addition to the user name, an authentication key (*authKey*) is shared between the communicating SNMP engines, ensuring authentication and integrity. A privacy key (*privKey*), also symmetric, ensures confidentiality.

Complementing the communication security, the SNMPv3 model also provides access control through a view-based access control model [11]. This model grants or denies access to MIB portions (view subtrees) according to the predefined configuration and the current user permissions.

The security details for SNMPv3, either for authentication, integrity, confidentiality and access control, dictates the security functions for the *SNMPFS*. We have to pass to the file system the authentication and the access control required by the SNMP model.

The authentication problem is performed by the system when mounting the file system. A similar approach is followed for NFS or SMB shares:

```
mount -t smb //server/share /mnt -o username=aUser,password=xxx.
```

If the server recognizes the username and password, the host is allowed to access the file system and a user ID (uid) is associated with it.

Access control is enforced by file permissions. In Unix, each file has a set of permissions (read, write, execute) for the file owner, group and others. For example, the permissions

```
-rwxr-x---
```

gives the owner the possibility to read, write and execute the file, the group to read and execute and no other user can read, write or execute.

SNMPFS translates each file permission to the View-based Access Control mechanism of the SNMPv3.

2.4 Attributes

File system entries, such as files or directories, are characterized by a set of attributes which describes their fundamental aspects, such as size, date, permissions, name and others. The name and number of attributes is typically static, meaning that it is not possible to add or remove further information to each file system entry latter on.

An attribute which is necessary to better describe the data types and the structure of an SNMP agent is the MIB tree it implements. The MIB tree is described in a set of MIB files which contain each node name, data type, restrictions and role. With this information, the SNMPFS can present to the user a more meaningful set of file names as well as file types (a table, a string, an int, etc). In particular, this information is valuable for tables, which the SNMPFS exports as Coma Separated Values (CSV) format and can be opened and edited by a spreadsheet, such as Microsoft Excel or OpenOffice Calc.

To be able to access the meta-information about management data, the SNMPFS has the possibility to load MIB files from a specific directory. This information will allow the files to have a more meaningful name as well as adapting the content to the nature of the information it stores.

3 IMPLEMENTATION DETAILS

The SNMPFS implements a gateway between regular file system access primitives and SNMP commands. Generally, file systems are implemented at kernel space, however, we chose to implement the file system in userspace to facilitate the development and test process. We used FUSE [12], a stable and well known API for file system development.

The SNMP information is somehow austere, mainly because of the Object IDs (OIDs) that it uses to identify each managed object. The OID is a sequence of integer values, separated by a ‘.’ (dot): 1.3.6.1.2.1.1.1.0. This sequence defines a path in the agent’s tree of objects, referring to a specific value (Figure 3). This path, for example, points to a specific object, which refers to a value resulting from device instrumentation.

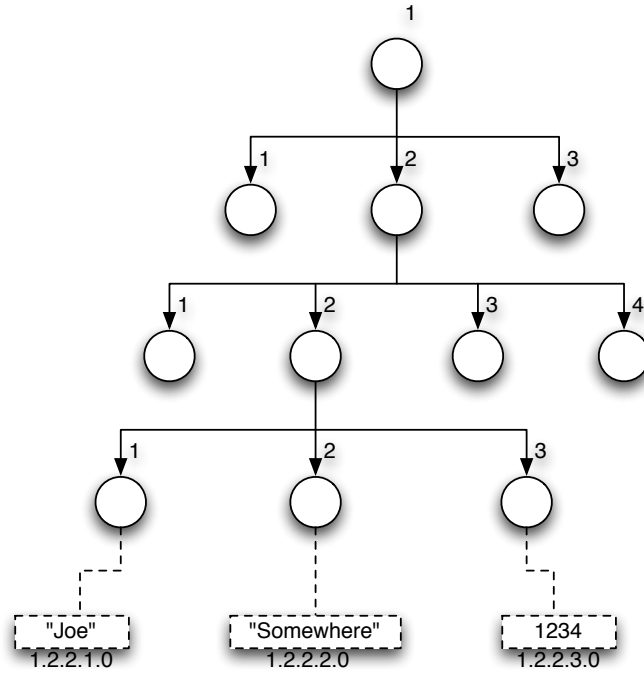


Fig. 3: Scalar values organization in the agent.

In Figure 3, for example, the OID 1.2.2.1 refers to the lower left node in the tree. This node is associated with a specific value, a scalar, in this case, which contains the string “Joe”. The scalar is viewed as an additional node, a leaf, and is referred by adding a ‘.0’ to the OID.

Each object, the circles in the figure, has a set of attributes or meta-information, which allows the user to get the semantics of the value (what does “Joe” stands for). The attributes, as well as the overall structure, is described in a file, called a Management Information Base (MIB), which associates a descriptive, meaningful, name to each object and further describes the data type, access restrictions, OID structure and others. From the user perspective, this also allows mapping the sequence of integers to a short name: it is easier to refer to each object by the short name, instead of the OID (get sysDescr, instead of get 1.3.6.1.2.1.1.1.0).

After parsing the MIB files, the *SNMPFS* performs this mapping, storing the meta-information to improve the information provided to the user by the file system. The OID in the sequence of integers format will only be used when the MIB is not available.

3.1 The MIB parser

As mentioned above, the MIB structure is important to *SNMPFS* to provide a more useful and meaningful view of the file system. MIB files are written in a subset of the Abstract Syntax Notation One (ASN.1), called “Structure of Management Information” [13].

The MIB must be parser so that the tree, nodes, types and objects extracted. We are using Marser [14], an API to parse SMI (v1 and v2).

Moreover, the information from the MIB allows to identify tabular information, which further helps the file system to present the information to the user in a more manageable way. In this case, tables will be available is CSV format, allowing the user to read and modify it using a spreadsheet application.

Tables are represented as a further extension to the OID tree (Figure 4). As in the previous case, where each scalar is retrieved from a leaf, tabular values are retrieved from several leaves, hanging on the OIDs that represent the columns (in the figure, the table is referred with the OID 1.2.3, which has the columns 1.2.3.1 and 1.2.3.2).

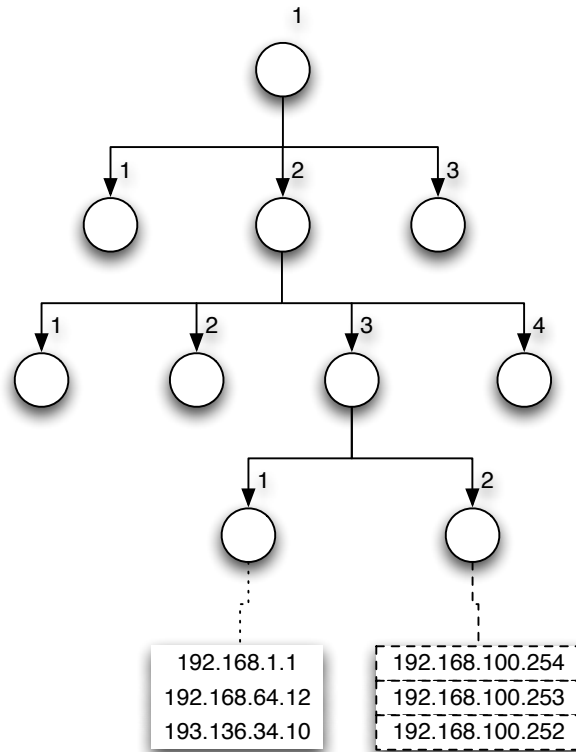


Fig. 4: Tabular values organization in the agent.

One or more of the columns are the index, which allows to retrieve the rows in the table. So, to get the first value of the table, it is necessary to issue: get 1.2.3.2.192.168.1.1, which yields 192.168.100.254.

3.2 Files

When dealing with agents with an unknown structure, the user usually has to explore the management information tree, retrieving all the information that the agent stores. This operation is called ‘walk’, because it allows to visit all the places “hidden” in the agent. The `SNMPFS` has a special file which allows doing precisely this. The file, called `_walk.txt`, lists all the managed objects retrieved as the result of a ‘walk’ operation. By simply opening this file in a text editor, the user will be able to immediately see the objects the `SNMP` agent implements:

```
sysDescr;1.3.6.1.2.1.1.1.0
sysObjectID;1.3.6.1.2.1.1.2.0
sysUpTime;1.3.6.1.2.1.1.3.0
sysContact;1.3.6.1.2.1.1.4.0
sysName;1.3.6.1.2.1.1.5.0
sysLocation;1.3.6.1.2.1.1.6.0
sysServices;1.3.6.1.2.1.1.7.0
sysORLastChange;1.3.6.1.2.1.1.8.0
sysORID;1.3.6.1.2.1.1.9.1.2.1
sysORID;1.3.6.1.2.1.1.9.1.2.2
sysORID;1.3.6.1.2.1.1.9.1.2.3
sysORID;1.3.6.1.2.1.1.9.1.2.4
sysORID;1.3.6.1.2.1.1.9.1.2.5
sysORID;1.3.6.1.2.1.1.9.1.2.6
...
```

With the information obtained in the file, the user can configure the file system, describing which files should be available and what is the name they should have. The configuration is written in XML and defines all the aspects of the file-system: the agent’s address, access credentials, MIBs to load, which nodes to show and where to mount:

```
<device name="device">
  <mount dir="tmp" />

  <mibs dir="../mibs/">
    <mib file="SNMPv2-MIB"/>
    <mib file="RFC1213-MIB"/>
    <mib file="IF-MIB"/>
  </mibs>

  <snmp address="192.168.1.1" port="161"
    version="v2c" community="public" />

  <entries>
    <scalar label="sysUpTime" />
    <scalar label="sysDescr" />
    <table label="ifTable" />
```



```

<table oid=".1.3.6.1.4.1.63.501.3.2.2"
  file="myTable">
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.1" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.2" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.3" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.4" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.5" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.6" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.7" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.8" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.9" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.10" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.11" />
  <col oid=".1.3.6.1.4.1.63.501.3.2.2.1.12" />
</table>
</entries>
</device>

```

The previous configuration file will result in the appearance of 5 files in the 'tmp' directory – two tables, two scalars and the `_walk.txt` (Figure 5).

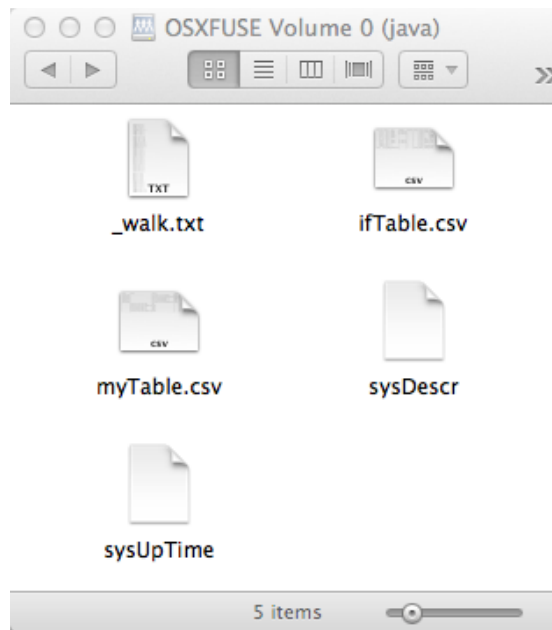


Fig. 5: Screenshot of an SNMPFS directory.

3.3 Values and Tables

Each file representing a scalar simply has to get the value from the agent each time it is read. In the previous configuration file, the scalar 'sysUpTime' and 'sysDescr' show as files, containing the information from the agent located in 192.168.1.1.

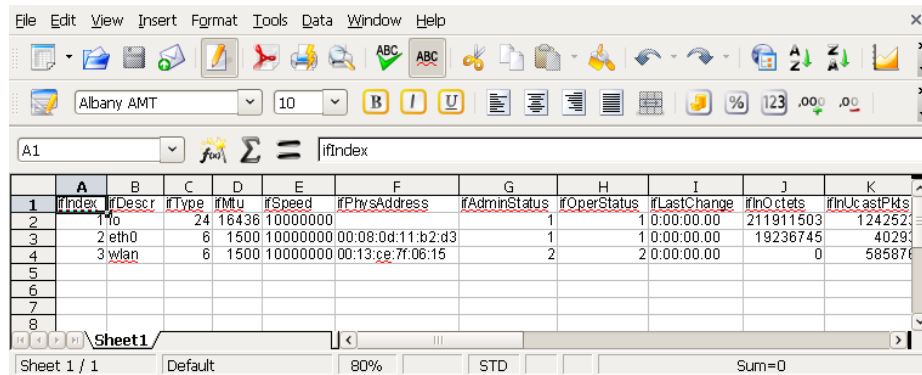
Tables, because of the tree like structure, require more processing. The algorithm we follow is:

```

read configuration file;
for each entry
  if is table
    read columns;
    if columns empty
      read columns from MIB;
  for each column
    while has more leafs
      get leaf;
      store in row,column;
  end;
end;

```

The fetch of values and store in row and column format allows to build the content around the CSV format, that can be manipulated by a spreadsheet application (Figure 6).



	A	B	C	D	E	F	G	H	I	J	K
	ifIndex	ifDescr	ifType	ifMTU	ifSpeed	ifPhysAddress	ifAdminStatus	ifOperStatus	ifLastChange	ifInOctets	ifInUcastPkts
1	1	lo	24	16436	100000000		1	1	0:00:00:00	211911503	124252
2	2	eth0	6	1500	100000000	00:08:0d:11:b2:d3	1	1	0:00:00:00	19236745	4029
3	3	wlan	6	1500	100000000	00:13:ce:7f:06:15	2	2	0:00:00:00	0	58587
4											
5											
6											
7											
8											

Fig. 6: Editing the ifTable with a spreadsheet.

4 USE CASES

In system and network management, the administrators are used to create scripts to automate some of the typically repetitive and/or boring maintenance activities. Many of this scripts work by reading, writing and updating configuration files, altering the way daemons and services work (such as e-mail, HTTP, SSH, ...). The SNMPFS enables

mapping SNMP information to a file system structure, thus contributing to the integration of monitoring, configuration and accounting processes. In other words, the `SNMPFS` will foster administrators to develop new tools easier and to use the same tools in different scenarios because of the transparency and integration of the file system paradigm.

The extension of the file system with files related to instrumentation and configuration information from network devices will further alleviate the burden. By mounting several devices in the same file system, where each directory represents a different agent, enables the administrators to be able to make queries or change values on all the equipment, just by browsing the file system.

4.1 Monitoring

Monitoring operations require the user to retrieve, process, analyse and visualize instrumentation information. For example, several parameters can be queried to get the status of remote hosts (Table 1).

Table 1: Monitoring examples.

Query	Object	Type
the number of users on a system	hrSystemNumUsers	scalar
the bytes transmitted	ifInOctets;ifOutOctets	table
the storage areas	hrStorageTable	table
the processor load	hrProcessorLoad	table

Other common operation is to build the topology map of the network, representing the connections and hosts structure. This is done with the help of the IP forwarding table, maintained in the switches and routers. Each table gathers the MAC addresses in each port, allowing the correlation of addresses into building a visual representation of the network topology. By replicating this information (a simple copy will do), will enable to create a view of the network a specific times.

4.2 Configuration

One challenge on integrated management of networks is how to apply policies that are transversal to more that one equipment. The integration of several SNMP agents in the same file system can enable the administrator to create sound scripts to apply the policy.

In complement to these use cases, one that is being currently addressed is the use of version control systems for maintaining snapshots of SNMP agents' configuration. Each type and version of equipment has different ways for manipulating their configurations, because of the MIBs they implement. With the `SNMPFS`, administrators can make use of already accepted solutions for version control. A Distributed Version Control, such as `GIT`² enables to have a master repository for each agent as well as a local repository in the management stations.

² <http://git-scm.com/>

The first mount of the agent, a new repository is created and the files are added, tagged as the initial commit. This local repository is pushed to the master repository, which will work as another, more general, versioning peer. The master is configured for post commit routines, that will update the directory where the agent is mounted. The master will apply the changes in the file system, configuring the equipment accordingly and changing the properties that enables the updated of the configurations. This enables that different administrators can work on their stations using the repositories for changing configurations and the push the configurations to the master that will change the agents' status. Moreover, using tags to identify specific configuration versions allows to easily change from one configuration to another.

4.3 Scheduling operations

Modern operating systems have tools that enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates. One such tool, popular on Unix-like operating systems, is `cron`, used to automate system maintenance or administration. Proper configured cron jobs allows to activate some options at some time on agents, for example, to shutdown several devices at a specific time. Moreover, it also allows to watch files for specific values for, for example, triggering some configuration change or event (send emails, executing commands).

5 CONCLUSIONS

Accessing and updating information is a frequent operation in virtually any activity. Because of the evolution of computing platforms, the electronic information is associated with the concept of files, residing in a generic storage mechanism. Usually, the files are updated by general use applications, such as office suites or drawing editors. Often, the content is in plain text, allowing standard editors to retrieve and update the information.

Because of the ubiquity of files, modern operating system have an extensive set of tools to deal with the maintenance of files, such as renaming, creating, copying, backing up and restore, and so on. In enterprises, work files are typically stored in network storage, made available through a virtual file system in local desktop computers.

Network and system management (NSM) is a major concern for maintaining the system in good working conditions. Many of the tasks involved in NSM require monitoring and updating information resulting from instrumentation procedures in applications, services and equipment. The paradigm in traditional NSM models rely on client-server protocols, through special purpose applications.

The `SNMPFS`, proposed in this paper, integrates network devices, applications and service management in a common platform and paradigm – the file system. In this way, network management operations can benefit from the existing powerful operating system tools to monitor, update instrumentation, configuration and monitoring information.

References

1. Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., Noveck, D.: NFS version 4 Protocol. RFC 3010 (Proposed Standard) (December 2000) Obsoleted by RFC 3530.
2. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. Technical report, Google (2003)
3. Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J.: Scale and performance in a distributed file system. *ACM Transactions on Computer Systems* **6**(1) (feb 1988) 51–81
4. Anderson, T.E., Dahlin, M.D., Neeffe, J.M., Patterson, D.A., Roselli, D.S., Wang, R.Y.: Serv-less network file systems. *ACM Transactions on Computer Systems* **14**(1) (feb 1996) 41–79
5. Harrington, D., Presuhn, R., Wijnen, B.: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard) (December 2002)
6. Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., Sastry, A.: The COPS (Common Open Policy Service) Protocol. RFC 2748 (Proposed Standard) (January 2000) Updated by RFC 4261.
7. Kavasseri, R.: Distributed Management Expression MIB. RFC 2982 (Proposed Standard) (October 2000)
8. McCloghrie, K., Perkins, D., Schoenwaelder, J.: Textual Conventions for SMIV2. RFC 2579 (Standard) (April 1999)
9. Blumenthal, U., Wijnen, B.: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3). RFC 3414 (Standard) (December 2002)
10. Blumenthal, U., Maino, F., McCloghrie, K.: The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model. RFC 3826 (Proposed Standard) (June 2004)
11. Wijnen, B., Presuhn, R., McCloghrie, K.: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). RFC 3415 (Standard) (December 2002)
12. FUSE: Filesystem in userspace. <http://fuse.sourceforge.net> (2011)
13. McCloghrie, K., Perkins, D., Schoenwaelder, J.: Structure of Management Information Version 2 (SMIV2). RFC 2578 (Standard) (April 1999)
14. Marser: Marser - a mib parser. <http://marsers.sourceforge.net> (2007)