;login:

THE MAGAZINE OF USENIX & SAGE

February 2003 • volume 28 • number 1

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

remote monitoring with SNMP

A Practical Example

by Jozef Skvarcek

Jozef Skvarcek is currently working as a system administrator. He holds a PhD in Physics. Computer technology and science are his long-time hobbies.

jozef@photonfield.net

Introduction

My previous article (*;login:* Vol. 27, No.5, October 2002) introduced the Simple Network Management Protocol (SNMP) as a valuable tool in remote network monitoring. The configuration of the agent was shown using the NET-SNMP open source SNMP software as an example. The functionality of the protocol was demonstrated by retrieving the value of the system.sysUpTime OID and the values of the “host“ branch, which could be exploited for measuring the disk space utilization. In this article we shall provide a detailed procedure for the monitoring of the operating system parameters.

Our objective is to monitor the system availability, disk and swap space utilizations, running processes, and system load. The system availability check will be performed by ping. A system will be considered available if ping succeeds, but if ping fails no further SNMP polls will be done. The monitoring of the parameters will be done by the SNMP polls. It will be assumed that the management station and the monitored systems have the SNMP agents properly installed and configured. The following examples will use the commands that are part of the NET-SNMP agent software. The management information contained in the Host MIB, defined by RFC 1514, will be assumed to be supported by the agent. In a real situation, we would probably have a number of identical machines that are to be monitored. Therefore, the script which will implement the monitoring procedure should provide means for specifying the common thresholds for the groups of identical systems. The script should issue a notification whenever the threshold value for any parameter is exceeded. The syslog will be employed for the notifications.

We would like the syslog messages generated by the script to go to their special log file, say, /var/log/snmp-monitor.log. In order to achieve that, we will use the local syslog facility local1, which should not be used for any other purpose. Therefore, we put into the syslog’s configuration file /etc/syslog.conf the following line:

```
local1.*   /var/log/snmp-monitor.log
```

Of course, this is just an example. We could also direct the notifications to a remote syslog server, emit email or page messages, or send SNMP traps to a full-featured network management station (NMS) such as IBM Tivoli Netview.

Monitoring Disk Space Utilization

We would like to be able to measure the disk space usage on a remote system. How do we start? First, we need to determine the OIDs which hold the suitable management information. The Host MIB can be used for that purpose – it specifies the OIDs under the iso.org.dod.internet.mgmt.mib-2.host branch. By reading RFC 1514 we find hrStorageTable table which includes the entries relevant for achieving our goal. Among the others it contains the names of the file systems and their total and used sizes. Next, we want to see the exact result from the SNMP query of the hrStorageTable on the agent. In the following examples, we will work on the management station called “Jupiter” and will poll the agent named “Europa”. On Jupiter we run

```
$ snmpwalk europa .iso.org.dod.internet.mgmt.mib-2.host.hrStorage.hrStorageTable
```

which produces many lines of output. Here is a truncated example useful for further discussion:

```
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageIndex.3 = 3
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageType.3 = OID: host.hrStorage.hrStorageTypes.hrStorageFixedDisk
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageDescr.3 = /var
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageAllocationUnits.3 = 4096 Bytes
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageSize.3 = 1741346
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageUsed.3 = 22976
```

It describes the /var file system as given by the hrStorageDescr OID. The OID instance corresponding to /var is the number 3 which identifies the other OIDs that belong to the same file system. The information is sufficient for calculating the percentage of the free space in /var using the formula

```
Free space = ( 1 - hrStorageUsed / hrStorageSize ) * 100%.
```

The result can be compared to a threshold value; if it is lower, a notification should be generated.

Monitoring Swap Space Utilization

The approach is very similar to the previous paragraph. Again, we can take advantage of the information from the hrStorageTable from the Host MIB. However, now we are interested in the different lines from the output from the snmpwalk command:

```
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageType.102 = OID: host.hrStorage.hrStorageTypes.hrStorageVirtualMemory
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageDescr.102 = Swap Space
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageAllocationUnits.102 =1024 Bytes
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageSize.102 = 264952
host.hrStorage.hrStorageTable.hrStorageEntry.hrStorageUsed.102 = 0
```

These lines report the values of the OIDs with the instance number 102, which belongs to Swap Space. Similarly as in the previous case we can calculate the free space and compare it to the threshold.

One of the arguments for implementing SNMP for the remote monitoring is the fact that it is an open standard which allows for monitoring different operating systems or hardware platforms. The procedure described here is exactly like that. Well, almost exactly . . . What is our point? There are small nuances in the management information provided, even by the same agent software, depending on the particular platform. For example, the NET-SNMP agent on Solaris 8 does not show Swap Space in the hrStorageTable. On the other hand, it shows the /tmp file system with the storage type of a fixed disk. We know that the size of /tmp on Solaris is correlated with the amount of the available swap; therefore, as a workaround we could monitor the free space in /tmp. The conclusion is that one "has to see" the content of the management information before writing a monitoring application. In most cases, there are multiple ways to meet the objectives.

Monitoring CPU or System Load

The CPU load can be observed by checking the values of the OIDs in the hrProcessorTable table defined in the Host MIB. The relevant OID is named hrProcessorLoad. However, according to our experience this information is not provided by the NET-SNMP agent (v4.2.3) on RedHat Linux 7.2 or Solaris 8, an example of the kind of peculiarity mentioned in the previous paragraph. What can be done if we have GNU/Linux or Solaris agents? We can monitor the system load instead. The CPU and the system loads are not the same thing but they are normally correlated. That fact makes them equivalent for our purpose. The system load can be measured by polling the OIDs in the NET-SNMP MIB. The MIB file is called UCD-SNMP-MIB.txt and it is included with the NET-SNMP sources. The MIB defines the objects under the .iso.org.dod.internet.private.enterprises.ucdavis branch. By reading the MIB file, we locate the relevant information in the laTable table so that we can run, on Jupiter,

```
$ snmpwalk europa .iso.org.dod.internet.private.enterprises.ucdavis.laTable
```

which returns

```
enterprises.ucdavis.laTable.laEntry.laLoad.1 = 0.06
enterprises.ucdavis.laTable.laEntry.laLoad.2 = 0.03
enterprises.ucdavis.laTable.laEntry.laLoad.3 = 0.01
```

From the MIB, we know that the three instances hold load averages over one, five, and ten minutes. The five-minute average (instance 2) might be a good candidate for our purpose since it smoothes out occasional spikes. The value can simply be compared to the threshold, and if it is higher, then a notification should be created.

Monitoring Running Processes

Each production system runs specific processes that are important for the functionality of the environment it is a part of. For example, a Web server may run Apache. If Apache is not running, then we would like to be notified. Once again, the Host MIB comes in handy since it includes the hrSWRunTable, which contains useful objects that will allow us to meet the objective. Let's see exactly what information we can get by running the following on Jupiter:

```
$ snmpwalk europa .iso.org.dod.internet.mgmt.mib-2.host.hrSWRun.hrSWRunTable
```

Since the output is long, here is a truncated version for the purpose of illustration:

```
host.hrSWRun.hrSWRunTable.hrSWRunEntry.hrSWRunIndex.535 = 535
host.hrSWRun.hrSWRunTable.hrSWRunEntry.hrSWRunName.535 = "syslogd"
host.hrSWRun.hrSWRunTable.hrSWRunEntry.hrSWRunPath.535 = "syslogd"
host.hrSWRun.hrSWRunTable.hrSWRunEntry.hrSWRunParameters.535 = "-m 20 -r"
host.hrSWRun.hrSWRunTable.hrSWRunEntry.hrSWRunType.535 = application(4)
host.hrSWRun.hrSWRunTable.hrSWRunEntry.hrSWRunStatus.535 =  runnable(2)
```

This tells us that the syslogd daemon is running with the given parameters. If we specify the name of the process that should be present on the system, the script will try to find a match among the running processes. If the match is not found, the script will create the notification.

The Perl script snmp-monitor.pl which is available at *http://www.photonfield.net/snmp-monitor.html,* implements the algorithm for measuring the disk space discussed above. The script is executed on the management station. Our management station runs RedHat Linux 7.2 and our agents are a mix of Solaris, GNU/Linux. Very few modifications would be necessary for porting the script to another platform.

The script works with SNMPv3 using the set up described in my October *;login:* article. If SNMPv3 is not supported by the agents — for example, if there are Windows 2000 clients — then the SNMPv2 can be utilized instead. In that case we should modify the $SNMPWALK and $SNMPGET variables. The script parses the configuration file specified by $CONF_FILE. The configuration file has its own specific format:

```
disk          <disk_fs>      <disk_free>
host          <agents>
endprofile
```

The disk line is for specifying disk usage thresholds, <disk_fs> (string) is the name of the file system, and <disk_free> (integer) is the minimum amount of the free space expressed in percentages. There can be multiple disk lines. The purpose of the host line is to specify the hostname of the agent we want to monitor. There can be multiple host lines so that the identical parameter's thresholds can be used for monitoring a number of agents. The endprofile line specifies the end of a "profile," which is a set of related disk and host entries. We can have many profiles in the configuration file. The comments are allowed and should be on the lines that begin with #. For example, the following configuration file will monitor the file systems / and /var on agent Europa. The minimum amount of free disk space in both file systems is 20 percent.

```
disk    /         20
disk    /var      20
host    europa
endprofile
```

The script populates the arrays @disk_fs, @disk_free, and @agents according to the values found in its configuration file. The elements of the arrays are passed as the arguments when calling the subroutine &check_disk. The SNMP polls are performed by calling the binaries that are part of the NET-SNMP agent. The logger utility is employed for logging the notifications.

The management information available through SNMP is wide and is defined by a large number of standard or vendor-specific MIBs.

The algorithms for measuring the swap space and system load and for monitoring the running processes are very similar to the disk space monitoring. Their practical implementation is left to the reader.

Conclusions

The approach to utilization of SNMP for remote monitoring has been discussed and an illustrative Perl script has been shown. The development requires a certain degree of flexibility because the amount of management information can vary slightly on different platforms or for different SNMP software. The presented script has simple functionality. Nonetheless, it provides us with the single point of administration of the monitoring operations. All the monitored parameters are defined in the single configuration file.

The script could be enhanced in many ways. For example, it has no "memory," which means that it issues a notification every time it finds an exception. We may want to receive only one notification when the exception is detected for the first time and then another one when the things get back to normal. In addition, the script is serial, executing the pings and the SNMP polls one after another. If the number of agents is large it may take many minutes to poll all of them, especially when there are some timeouts. In order to make the execution time shorter we could make the polls run in parallel for multiple hosts, or perhaps we could develop a multi-threaded version.

The scope of the possible monitored targets is not limited to the presented parameters in any way. The management information available through SNMP is wide and is defined by a large number of standard or vendor-specific MIBs. Our ambition was to give readers a helpful practical example which would provide them with a head start for solving their particular problems.

References

Script

http://www.photonfield.net/snmp-monitor.html

Article

J. Skvarcek, "Remote Monitoring with SNMP," *;login:*, October 2002.

Books

D.R. Mauro and K.J. Schmidt, *Essential SNMP* (Sebastopol, CA: O'Reilly & Associates, 2001).

D. Perkins and E. McGinnis, *Understanding SNMP MIBs* (Upper Saddle River, NJ: Prentice Hall, 1997).

P. Simoneau, *SNMP Network Management* (New York: McGraw-Hill, 1999).

D. Zeltserman, *Practical Guide to SNMP v3 and Network Management* (Upper Saddle River, NJ: Prentice Hall, 1999).

Sites

NET-SNMP (UCD-SNMP): *http://net-snmp.sourceforge.net/*

RFC: *http://www.ietf.org/rfc.html*.