Table of Contents

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Chapter 7. Service Monitoring

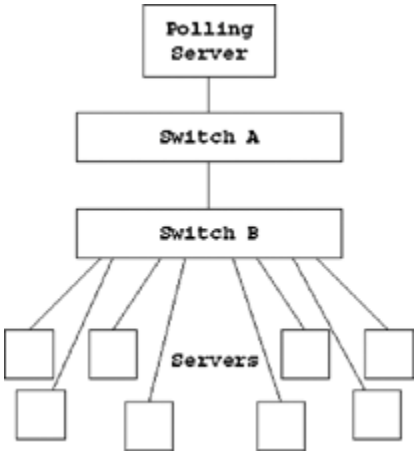
7.1. Overview of Service Monitoring

The most common type of problem that network administrators deal with is something that spontaneously stops working: A network link fails, a Web server refuses connections, or a switch stops passing traffic. Often, failures like these do not result in any kind of notification to administrators. A switch that loses power or suffers a software failure does not have any way of sending a message indicating that it is no longer functioning. It is is vital, however, for administrators to be notified of failures like this when they occur.

The solution is to use some kind of monitoring or polling software. This is a program running on one or more servers that sends out probes at regular intervals in order to test network connectivity and service functionality. This software may ping all of your switches and routers. If a device does not respond, the failure is reported to the appropriate administrators. The software might also attempt to retrieve Web pages, make SNMP requests, or perform other kinds of service level testing. Any failure to respond as expected can similarly be reported to an appropriate administrator.

There are some subtleties to performing this task effectively. Imagine the scenario depicted in Figure 7.1. Switch B is connected to a large number of servers. Each one is important, and if any one of them should fall off the network, an administrator needs to be paged. The polling server pings each one, and if any one server does not respond, a notification is sent. What happens if switch B itself fails? Not only can the polling server not contact switch B, but it can also no longer contact any of the servers behind the switch. If the polling software were not intelligent, it would send notifications about every service behind switch B. If all these messages were sent to a network administrator's pager, they would overwhelm the recipient and also obscure the real problem. For this reason, intelligent polling software includes a mechanism for describing which services depend on which others. If the software knows that switch B has to be operating in order for the servers behind it to respond, it can ignore the failed tests for the servers and simply report that switch B is not responding.

Figure 7.1. Many Servers Behind a Single Failed Device.



Note that system polling can have an impact on the performance of the network or the devices being monitored. If, for example, ping tests are performed at an excessively fast rate, the polling software itself can cause network congestion. Additionally, a network device with a relatively slow CPU can be easily overwhelmed by rapid ping tests or SNMP queries. Even devices with a fast CPU, such as high-end routers, can experience degraded service if asked to participate in an excessively large number of SNMP transactions. By default, system polling software will usually place an appropriate delay between tests. If you choose to change the testing interval or find that your network is experiencing

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

degraded service after you deploy monitoring software, check to make sure the monitoring software itself is not causing an unnecessarily high load on the system.

There are two pieces of free, open source software that make good service monitors. One is called Sysmon, available from http://www.sysmon.org/. This is a relatively simple program that is easy to configure and get running. It does not have many advanced features, but for a small to medium-sized network, it will get the job done. The other tool is called Nagios and is available from http://www.nagios.org/. This is a better tool for large networks. It is a much more complicated program, but it includes advanced functionality that may be required at a larger installation. This chapter focuses almost exclusively on Sysmon, except for a discussion at the end that lists the additional features available in Nagios.

7.2. What Service Monitoring Can Help You Do

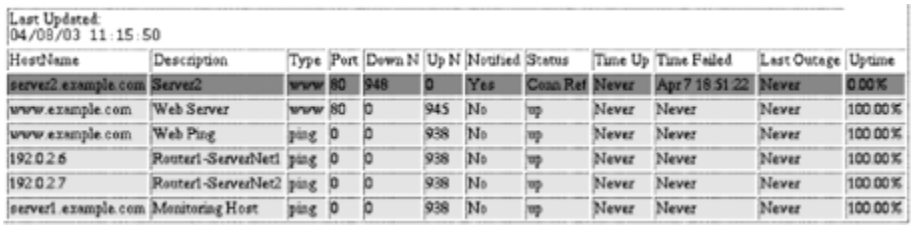
The most obvious benefit to running a service monitor is that it gives you early warning of failures. Practically speaking, this allows you to fix a problem before it becomes an even larger problem. If a piece of hardware fails in the middle of the night, you have the chance to repair or replace it before the start of business the next day. Or say you have a piece of network hardware whose sole purpose is to provide redundancy for another piece of hardware. When the primary device fails, the backup is configured to take over gracefully. If the primary fails and it goes unnoticed for months, what happens if the backup hardware fails as well?

Monitoring software can also help determine where in your network a problem resides. Users typically report problems only from their own point of view. If a connection serving hundreds of users goes down and you rely on user reports to notify administrators when something is wrong, by the time reports come in, there may be an overwhelming number of cases to deal with and it may not be immediately clear where the problem really is.

Additionally, it is always in your best interest to know about problems before either your customers or your superiors do, regardless of how quickly you intend to take action. There is nothing more embarrassing than hearing that a core service is down from one of them first.

In daily use, you will typically ignore the system polling software until it pages you or otherwise notifies you of a problem condition. You may also view the status of monitored services on a Web page such as the one in Figure 7.2. Here, the darker line indicates a service that is not responding, and the lighter lines are servers that are functioning without any problem.

Figure 7.2. Sample Sysmon Status Web Page.



Last Updated:
04/08/03 11:15:50

HostName	Description	Type	Port	Down N	Up N	Notified	Status	Time Up	Time Failed	Last Outage	Uptime
server2.example.com	Server2	www	80	948	0	Yes	Conn Ref	Never	Apr 7 18:51:22	Never	0.00%
www.example.com	Web Server	www	80	0	945	No	up	Never	Never	Never	100.00%
www.example.com	Web Ping	ping	0	0	938	No	up	Never	Never	Never	100.00%
192.0.2.6	Router1-ServerNet1	ping	0	0	938	No	up	Never	Never	Never	100.00%
192.0.2.7	Router1-ServerNet2	ping	0	0	938	No	up	Never	Never	Never	100.00%
server1.example.com	Monitoring Host	ping	0	0	938	No	up	Never	Never	Never	100.00%

7.3. Installing Sysmon

The following sections provide specifics on installing Sysmon.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

7.3.1. Where to Place the Server

Because Sysmon will suppress notifications for any failed devices behind a single failed device, it is important to run the monitoring program somewhere that has a good view of the network. It would be a mistake to place the server four devices removed from the core of the network, especially if one of those devices tends to fail often. When one does fail, Sysmon will lose its view of the rest of the network. You would rather have the Sysmon server located as close to the core as possible, if not directly on the backbone. This way Sysmon will be able to give the most accurate report possible when something fails.

7.3.2. How to Install Sysmon

The installation for Sysmon is mostly straightforward. Download the latest version from http://www.sysmon.org/, then unpackage and build the software. There is one step that is out of the ordinary; you must change to the `src` directory before typing `make`:

```
Solaris# gunzip -c sysmon-0.91.17.tar.gz | tar xvf -
Solaris# cd sysmon-0.91.17
Solaris# ./configure
Solaris# cd src
Solaris# make
```

On Solaris, you may see a list of warnings for every file that is built. This is not a serious problem; it is the compiler complaining about the curses library. Sysmon uses curses to build a program that will display a nice list of unreachable hosts. As long as the files `sysmon` and `sysmond` are present, the build was successful:

```
Solaris# file sysmon
sysmon:         ELF 32-bit MSB executable SPARC Version 1, dyn...
Solaris# file sysmond
sysmond:        ELF 32-bit MSB executable SPARC Version 1, dyn...
```

At this point, you can install the software from a root account. By default, it will place the `sysmon` and `sysmond` programs in `/usr/local/bin/`:

```
Solaris# make install
```

7.4. Using Sysmon

The following sections discuss the details of running Sysmon.

Starting the Sysmon Daemon

While Sysmon does not strictly have to be run from a root account, any account that does not have root privileges will be unable to run ping tests. So practically speaking, you will probably want to run the daemon as root. If you are concerned about security and do not need to use ping tests, you may run Sysmon from a user-level account.

Before starting the Sysmon daemon, you will need to create a configuration file. A short sample config will be sufficient for now. Place the following in `/usr/local/etc/sysmon.conf`, which is the default location for the configuration file:

```
root="server";
config showupalso;
config statusfile text "/var/tmp/status.txt";
object server {
   ip "server.example.com";
   type ping;
   contact "admin@example.com";
};
```

Replace “server” and “server.example.com” with the name of the machine you are running Sysmon on and replace “admin@example.com” with your email address. Now you can start the daemon with:

```
Solaris# /usr/local/bin/sysmond
sysmond: 15:25:36 Starting System Monitor version v0.91.17
System Monitor version v0.91.17
/usr/local/bin/sysmond started on server.example.com
forked process as pid 7467
```

If you are running Sysmon from a user-level account and do not have access to write the configuration file to `/usr/local/etc`, you may place it in another file and start sysmond with the `-f` option:

```
Solaris% /usr/local/bin/sysmond -f /var/tmp/sysmon.conf
```

If the configuration has an error, it will be reported at startup time. Some errors are considered only warnings, and `sysmond` will start with the new config regardless. For example, if no description had been set for the “server” object:

```
Solaris# /usr/local/bin/sysmond
sysmond: 15:30:57 WARNING: object has no descripton near line 10
sysmond: 15:30:57 Starting System Monitor version v0.91.17
System Monitor version v0.91.17
/usr/local/bin/sysmond started on server.example.com
forked process as pid 7476
```

If Sysmon detects an error in an object definition, it may choose to exclude that object from the configuration and start without it.

Once Sysmon is up and running, you can check the status of monitored devices by looking at `/var/tmp/status.txt`, whose path was specified in the configuration file:

```
Solaris# cat /var/tmp/status.txt
Network Summary     System Monitor version v0.91.14
Hostname     Type  Port DownN UpN   NotifiedStat     Time Failed
server       ping  0    0     220 No     up       Never
```

This file is periodically written by Sysmon, and it will contain the latest status information as long as the Sysmon daemon is running.

Stopping the Sysmon Daemon

Though you can kill Sysmon by sending it a signal, it is more convenient to use the `sysmond` program instead:

```
Solaris# /usr/local/bin/sysmond stop
sysmond: 15:35:33 sending signal 15 to sysmond process 7467

sysmond: 15:35:33 Please remain seated as your ride comes to a...
```

As the output indicates, this is equivalent to sending a TERM signal to the sysmond process.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Pausing Sysmon

You can also instruct Sysmon to temporarily stop functioning and then resume later with the `pause` and `resume` commands:

```
Solaris# /usr/local/bin/sysmond pause
sysmond: 15:37:15 sending signal 17 to sysmond process 7486

Solaris# /usr/local/bin/sysmond resume
sysmond: 15:37:19 sending signal 17 to sysmond process 7486
```

Both commands send a USR2 signal to the `sysmond` process, which toggles between the paused and running states.

Reloading the Configuration

After changes are made to the configuration file, Sysmon has to either be restarted or be instructed to reload the configuration. Otherwise, the changes will not take effect. The `reload` command is executed as:

```
Solaris# /usr/local/bin/sysmond reload
sysmond: 19:07:23 sending signal 1 to sysmond process 7486

sysmond: 19:07:23 Done reloading new config file
```

The reload command sends a HUP signal to `sysmond`. If the configuration is not valid, the process will continue to run with the old configuration. Otherwise, the new configuration will take effect.

Connecting with a Remote Client

Sysmon runs a TCP service on port 1345 where it provides data about monitored services. You can connect to it using the curses client that was built and installed as `/usr/local/bin/sysmon`:

```
Solaris% /usr/local/bin/sysmon server.example.com
```

The screen will clear and a display like the following will come up:

```
Server: server               Current Time:  Apr  7 18:22:57 2003
Hostname         Type Port Count Notif Stat       Time Failed
-------------------------------------------------------------------
www.example.com  www  80   66    Yes   Conn Ref   Never


-------------------------------------------------------------------
 q = quit  space = refresh  h = help
```

Only services that are down or have failed tests will be present. You can exit this application by pressing press the “q” key.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Other Runtime Options

Running the `sysmond` program with the `-help` argument will produce a listing of all the options that can be used on the command line:

```
Solaris# /usr/local/bin/sysmond -help
Usage: /usr/local/bin/sysmond [ -f config-file ] [ -n ] [ -d ]
   [ -v ] [ -t ] [ -p port ] [ reload ]
  -b             : IP Address to listen on
  -f config-file : Alternate config file location
          DEFAULT: /usr/local/etc/sysmon.conf
  -n             : Don't do notifies
  -d             : Don't fork
  -i             : Disable ICMP
  -v             : Print version then exit
  -w             : Toggle warning messages
  -D             : Toggle debug messages
  -M             : Toggle memory debugging
  -t             : Test/check config file then exit
  -p #           : Change port number listening on (0 to disable)
  -q             : Quiet
  -l             : do not syslog
  reload         : Test/check config file, and if it passes ...
  pause          : Suspend/resume monitoring (SIGUSR2)
  resume         : Suspend/resume monitoring (SIGUSR2)
  stop           : End monitoring and quit (SIGTERM)
```

7.5. Configuring Sysmon

Once you have verified that `sysmond` is up and running, you can begin to fill out the configuration with the devices you wish to monitor. The Sysmon configuration is made up of a list of objects to be monitored, along with a section of global configuration options. Within each object definition, you must specify the IP address of the device to be tested, the kind of test that should be performed, and any objects the device depends on. When object A is configured to depend on object B, notifications will not be sent about the status of object A if object B is also unreachable.

Note that each line of the configuration ends with a semicolon, except for lines that end with an open curly brace. Additionally, lines beginning with a pound sign (#) are ignored and can be used for comments.

7.5.1. The Root Node

Sysmon requires one object to be defined as the root of the device hierarchy. This is the object that all others depend upon to be up. A good choice for this is the server that Sysmon runs on itself, as was configured in the earlier example. Note that in defining the root node, the name of the object must be in quotation marks:

```
root="server";
```

7.5.2. Objects and Dependencies

Now you can add to the configuration other objects to be monitored. Start by adding a simple ping test for the router that server.example.com is connected to:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
object router1 {
  ip "192.0.2.5";
  type ping;
  desc "Router1";
  dep "server";
  contact "admin@example.com";
};
```

The Object Name

First note the name of the object, here "router1." The name does not need to correspond to the hostname of the device, though having it correspond may keep the configuration easy to maintain. The object name is used solely for referencing the object later in the configuration. It can be any text string you like, as long each object has a unique name.

Setting the IP Address

The first object option listed above is the `ip` option, which specifies the IP address of the device to be tested. This field can really contain either the hostname or the IP address of the device. There are advantages and disadvantages to both. If an IP address is used, Sysmon will have no dependence on DNS's working properly, and the test will be performed appropriately even if DNS has failed. However, if you have many pieces of equipment to keep track of and the hardware being monitored may be replaced by a device with a different IP address that later takes over the older hostname, it would be highly preferable to use hostnames instead of IP addresses. This way, your Sysmon configuration does not become a second place where DNS information must be maintained. Sysmon does keep its own DNS cache, which gives you some ability to control the interaction between Sysmon and the DNS independent of the software running on the server. The options that control this behavior are described in the section on Global Options.

Setting the Test Type

The next line in the object added above directs Sysmon to perform a ping test on the device in question. There are 10 other possible values for the the test type, all of which are listed in Figure 7.3. Most of these tests are configured just like the ping test: Simply declare the test type, and you're done. For a few of them, there are extra options you must use. For example, in configuring an object to test Web service, using the `www` test type, you must include the `url` and `urltext` options:

Figure 7.3. Sysmon Test Types.

Test	Function	Options
`ping`	standard ping test	
`pop3`	working POP3 server	`username`, `password`
`tcp`	generic listening TCP port	`port`
`udp`	generic listening UDP port	`port`

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Test	Function	Options
radius	working radius server	username, password, secret
nntp	listening news server	
smtp	listening mail server	
imap	listening IMAP server	
x500	listening x500 directory server	
www	listening web server	url, urltext
sysmon	running remote sysmon server	

```
object web-server {
  ip "www.example.com";
  type www;
  desc "Main Web Server";
  dep "router1";
  url "http://www.example.com/";
  urltext "<TITLE>";
  contact "admin@example.com";
};
```

This tests the URL `http://www.example.com/`. If the page is not loadable or does not contain the text <TITLE>, the test will fail.

The `tcp` and `udp` tests must have a port number specified so that Sysmon knows which port to test. The `pop3` test requires a valid username and password, and the `radius` test requires the same, along with a radius secret string.

Setting the Object Description

The object description, set with `desc`, is simply text you add to help identify the object in reports. Sysmon will run if you do not include this option, but because of the way the configuration file is parsed, if the `desc` option is not present, the description from the previous object will be used. This is probably not the desired behavior. If you explicitly want to set no description for a device, you can do it with:

```
desc "";
```

Specifying Dependencies

An object's dependencies are set with the `dep` command. Every object must depend on at least one other object, with the exception of the root node, which has no dependencies. An object may depend on more than one other object by including multiple `dep` lines:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
object server1 {
  ip "server1.example.com";
  type ping;
  desc "Server 1";
  dep "router1-servernet";
  dep "router2-servernet";
  contact "admin@example.com";
};
```

In this example, server1.example.com has a connection to two different routers, which are objects defined elsewhere in the config. If either router is functioning, there will be connectivity to the server. By listing two dependencies, you can direct Sysmon to ignore the server's status only if both routers are down.

Also notice that the address listed for each router is the address of the relevant interface. It is good practice to use separate dependencies for each interface instead of using just one of the router's addresses for every dependency. This way, if a particular interface goes down, you will ignore only services behind that interface instead of the services behind the entire router.

Setting the Contact

The `contact` command specifies the email address that should be notified if an object fails its tests. You can list multiple email addresses by separating them with commas:

```
contact "admin@example.com,joe-pager@example.com";
```

Using the Spawn Option

Email is not always the best way to notify an administrator of a critical problem. To begin with, email is not guaranteed to be a timely service. Though abnormal, it is possible for an email message to be queued for hours, or even days. Additionally, email notifications will be useless if the mail system itself is unavailable. It is preferable to send critical notifications via some other mechanism, such as a direct message to a pager or cell phone.

Sysmon does not yet have support for sending pages by itself, but there is a hook that will let you do it. The `spawn` command will execute a program of your choosing when notification needs to be sent for an object. The argument to `spawn` is the name of the program to execute and the arguments that should be passed to that program. In those arguments, you must specify the format of the message to be sent. Sysmon has a number of "replacement" variables that will translate to different pieces of Sysmon information. For example, `%H` is replaced with the DNS name of the host being monitored, `%s` is the name of the service, and `%U` is the state of the service, either "up" or "down." So the spawn line might look like:

```
spawn "/var/tmp/notify.sh %H %s %U";
```

Then you can create a simple program `/var/tmp/notify.sh`:

```
#!/bin/sh
echo "$*" | /usr/lib/sendmail admin@example.com 2> /dev/null
```

Of course, it would be silly for your notification script to send email like this since you could accomplish your goal just as well with the `contact` command. It is used here simply as an example. In your own environment, you would instead send the text to a program that would send a page, such as QuickPage.[1]

[1] The QuickPage program is available from http://www.qpage.org/.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

The result of the above `spawn` command, if a service goes down, would be text that looks like:

```
WWW.EXAMPLE.COM www down
```

There are many more replacement variables available, and you can use them to create as detailed a message as you would like. They are all listed in Figure 7.4, which is taken from the Sysmon documentation.

Figure 7.4. Sysmon Replacement Variables. From Sysmon online documentation at `www.sysmon.org/config.html`.

Var	Replacement
%m	local host name
%H	DNS name of host being monitored
%s	service
%p	port number (numeric)
%T	Current Time hh:mm:ss
%t	Current Time mm dd hh:mm:ss
%d	Downtime dd:hh:mm
%D	Downtime with seconds dd:hh:mm:ss
%i	Unique ID for outage
%I	IP of host down
%w	warning/what
%u	error-type converted into string describing it
%h	hostname with failure
%r	reliability percentage
%V	Verbose History (not implemented)
%c	Failure iteration count (since last success)

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Var	Replacement
%C	Success iteration count (since last failure)
%U	Service state (as 'up' or 'down')

Other Object Options

A couple of other options can be used in an object definition. In particular, the `contact_on` option can direct Sysmon to send notifications only when a service goes up or when it goes down, instead of on both occasions, as is the default behavior. There is also an option called `reverse` that swaps the meaning of "up" and "down" for service status. This is useful if you use Sysmon to monitor another Sysmon server.

The use of these and other object options is listed in the documentation that comes with the Sysmon package and on the Sysmon Web page.

7.5.3. Global Options

Most of the global options in the Sysmon configuration start with the word `config`, followed by the option name. While global options can be listed anywhere in the configuration, they are usually placed at the beginning of the file, before any objects are defined.

The Status File

When the Sysmon daemon is running, it will periodically write a file with the status of services that it is monitoring. This can either be in HTML to produce a Web page or in a raw text format suitable for viewing from a terminal. By default, only services that are down are listed, but you can change this by using the `showupalso` option described below. Use the `config statusfile` option to direct Sysmon to write the file:

```
config statusfile html "/usr/local/apache/htdocs/sysmon.html";
```

Or if you want a text file:

```
config statusfile text "/var/tmp/status.txt";
```

The time interval that Sysmon waits between refreshing these files is 60 seconds by default, but you can change it with the `html refresh` option. For example:

```
config html refresh 30;
```

would cause the file to be rewritten every 30 seconds.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Viewing Both Up and Down Services

The status file will print a list of services that are not responding, but if you would like it to also include those services that are responding, use the `showupalso` option:

```
config showupalso;
```

In the HTML version of the status file, hosts that are up are printed in green, hosts that are down are in red, and hosts that are failing tests and may go down are yellow. These colors can be changed with the `upcolor`, `downcolor`, and `recentcolor` options, respectively.

Mail Header Options

Ordinarily, Sysmon sends mail messages with a from address of "root" at the server the software is running on. You can change this and other mail headers in the global configuration:

```
config from "admin@example.com";
config replyto "admin@example.com";
config errorsto "errors@example.com";
```

You can also change the format of the subject line, using the same replacement variables as before:

```
config subject "Sysmon: %H %s %U";
```

Or you can disable subject lines entirely with:

```
config nosubject;
```

This can be advantageous if you use email to send messages to a phone or pager that does not handle subject lines gracefully.

Test Queuing Options

There are a few options available that control how Sysmon processes service tests and notifications. First, the `numfailures` option controls how many tests a service must fail before a notification message is sent. By default, a service must fail four tests in a row, but if you wanted notification after only three failures:

```
config numfailures 3;
```

Normally, Sysmon schedules a test to be run 60 seconds after the last time the same test was completed. If you are monitoring a very large number of services, you can reduce the load on the server by increasing this interval. If you need tests run faster, thereby making Sysmon more sensitive, you can decrease it. The following would set the interval to 90 seconds:

```
config queuetime 90;
```

Though it is tempting to lower the queue time in order to make Sysmon detect problems quickly, anything lower than 60 seconds is probably excessive.

Sysmon will happily run more than one test at a time; by default, it will run up to 100 tests simultaneously. You can raise or lower this value with the `maxqueued` option:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
config maxqueued 50;
```

A server that has limited resources and many tests to run may need to run fewer tests at once, while a fast server can benefit by taking advantage of the ability to run more tests at a time.

Usually Sysmon sends only one notification when a service changes status. If you would like it to send repeated messages when a service is down, you can use the `pageinterval` option. Unlike with the other options, the units are in minutes:

```
config pageinterval 20;
```

This would send a reminder notification every 20 minutes when a service is down.

DNS Options

As mentioned earlier, Sysmon keeps its own DNS cache separate from the one the server's operating system uses. By default, entries from the cache are expired every 15 minutes, but you can change this with the `dnsexpire` option, which takes an argument in seconds:

```
config dnsexpire 300;
```

This would expire entries from the cache every 5 minutes.

Every 10 minutes, Sysmon also sends information about the cache to syslog. This interval can be changed with the `dnslog` option, in seconds:

```
config dnslog 900;
```

Message Formatting Options

If you do not like the message format that Sysmon uses by default, you can change it by using the replacement variables described earlier, in conjunction with the `pmesg` global configuration option. For example:

```
config pmesg "%H %s %U";
```

would produce the simple text just as before, but this time, it would be the body of all mail notifications. The default message format is:

```
%H (%I) %w is %u %d
```

Using Variables

If you have an even moderately sized installation, you may have many objects configured with the same information again and again, and worse yet, some of that information may need to change over time. Say you have a couple of different groups of people that should receive notifications when different services go down. One group is responsible for a certain set of hardware, another is responsible for a different set. You can list all the email addresses with every object, but then when an address needs to be added or removed, you will need to reconfigure every object. Instead, you can use a global variable to store information once, and that information will be used as is throughout the rest of the file. When you wish to make a change, you will have to do it in only one place. The following example demonstrates the use of variables to store lists of contacts:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
set network-group = "netops@example.com, joe-pager@example.com";
set network-group-nopage = "netops@example.com";
set web-group = "frank@example.com, jill@example.com;

object router5 {
  ip "router5-backbone.example.com";
  type ping;
  desc "Router 5 Backbone";
  dep "server";
  contact "$network-group";
};

object web-ping {
  ip "www.example.com";
  type ping;
  desc "Web Server Ping";
  dep "server";
  contact "$network-group-nopage";
};

object web-server {
  ip "www.example.com";
  type www;
  desc "Main Web Server";
  dep "web-ping";
  url "http://www.example.com";
  urltext "<TITLE>";
  contact "$web-group";
};
```

The variables defined at the beginning are referenced later with a dollar sign in front of the variable name. Notice that quotation marks still need to be used, even when a variable is referenced.

Using Includes

Another problem for large installations is that the configuration file can quickly become large and unwieldy. It will be easier to maintain if you break it down into smaller files by whatever grouping makes the most sense for you. Sysmon will let you do this with the `include` option:

```
include "/usr/local/etc/sysmon.webservers.conf";
```

The named file will have its contents included wherever the statement is placed in the configuration.

There are a number of different ways to organize groups of configuration files. You may wish to have different files for different services: one for Web servers, one for ping tests, and so on. Or perhaps it would work better in your environment to organize into different files for different physical parts of the network.

Other Global Options

Other global configuration options available include an option to change the file to which the process ID is written, an option to turn off registration messages sent to the Sysmon registration server, and an option to change the facility to which the program sends syslog messages. These are all described in the documentation that comes with Sysmon.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

7.6. Maintaining Sysmon

The toughest part of maintaining Sysmon is keeping your configuration in check with reality, especially if you have a large installation or if the people who are deploying equipment are not the same ones who will be updating the config. However, if you wish to have your notification system consistently alert you to trouble, you must keep the configuration accurate. This requires some discipline. Because failures do not occur often, it is easy to forget about the importance of updating the files. If the data becomes stale, with many devices marked as failed that really have been taken out of service, it will become hard to trust Sysmon when it reports that a device has failed. It becomes the server that "cried wolf."

7.7. Nagios

While Sysmon is a good package that takes care of the needs of the average network administrator, it does have a few limitations that can be serious drawbacks for very large networks. Nagios,[2] available from http://www.nagios.org/, is better suited to these installations. Some of the major advantages of Nagios are:

[2] According to the Nagios Web site, the name is pronounced *nah-ghee-ose* and has nothing to do with the fact that the service nags you when it detects system failures.

- **Escalation.** Nagios can be configured to take one action when a service first fails and then different actions if it continues to fail. This way, you can receive an email or instant message when a device is first encountering a problem but a message to your pager only if the problem persists for long enough. The actions and delays are all configurable.
- **Configuration templates.** Because Nagios has many more configurable options, the config file can grow even larger than Sysmon's. However, Nagios allows you to use templates to reduce the size and complexity of the configuration. You specify information that will remain the same for a large number of objects, give that information a name, and then reference it from any objects where it applies. Thus, the only information configured for a particular object is the information that makes it different from others.
- **Monitoring time periods.** Nagios has the ability to send notifications for objects during only certain time periods that you specify. For example, you can direct it to send pages for a particular object only during business hours. You can also inform Nagios about scheduled down time. This way, you can avoid paging the entire operations staff in the middle of the night while you are upgrading a piece of equipment.
- **Modular test plugins.** The tests that Nagios performs are all executed from a set of plugin modules. Each plugin is simply an external program that tests a service, but it follows a specific contract with Nagios so that the results can be processed appropriately. This means that it is easy to write your own tests to complement the suite of tests that comes with the Nagios plug-in package.
- **Passive tests.** Some information that you wish to monitor cannot be sent to the monitoring server by means of the server requesting the data. SNMP traps, for example, can be sent from a device at any time. Nagios can receive and monitor such data and report on it just as it would for an ordinary service test.
- **Host and contact groups.** Both hosts and people can be categorized into generic groups. These groups make it easier to change the configuration for a large, similar set of devices all at once. For example, you could change the escalation procedure for all core routers by changing only one line in the config.
- **Flap detection.** Occasionally, you will find that a test repeatedly fails and succeeds, causing a large number of up and down notifications. In this state, the service is said to be flapping. Nagios has the ability to detect flapping and automatically disable notifications for the service until the flapping has stopped.
- **Optional dependencies.** Dependencies are considered optional in Nagios but are required in Sysmon. Having optional dependencies is the equivalent of allowing as many root notes as you would like. While it is the case that every device ought to have at least one parent, it is sometimes practical and convenient to add a few tests that have no dependencies.

The downside to Nagios is that it is a much more complicated program than Sysmon. As a result, it will take a significant amount of time to install and configure. Whereas Sysmon can be set up in an afternoon, setting up Nagios may take several days or longer. If you need the functionality, however, it is well worth the time spent.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

7.8. References and Further Study

Both Sysmon and Nagios have documentation within their respective packages, and both have documentation online as well. Sysmon is at http://www.sysmon.org/ and Nagios is at http://www.nagios.org/.

Both packages also currently require you to use external software for paging. A popular program for this is QuickPage, available from http://www.qpage.org/. QuickPage uses the TAP/IXO protocol and the Simple Network Paging Protocol (SNPP), which is described in RFC 1861.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003