



Monitoring large systems and networks

Big Time

If you need to monitor very large networks, you need powerful software. OpenNMS offers serious monitoring capabilities in a truly open source package.

By Kurt Seifried

Despite administrators' best efforts, networks, servers, and services are destined to fail at some point. Sometimes the failure is obvious (a fan goes out or a backhoe meets your network cable), and sometimes the failure is subtle (a DNS change that breaks email to a server).

Troubleshooting these problems can be easy if you know the root cause, but that's not typically the case, and the task is orders of magnitude more difficult when you have 1,000, 10,000, or 100,000 machines to manage.

In such cases, OpenNMS [1] can help. When you need to monitor very large networks, especially geographically distributed and non-homogeneous (Windows, Linux, Solaris, AIX, etc.) networks, you need some pretty serious monitoring software. The big commercial products, such as IBM Tivoli or HP Systems Manager will do the trick, but they aren't open source or cheap. Thus, you might have issues if you need to customize them significantly, and tracking down the appropriate support can be difficult.

What Is OpenNMS?

At first, I thought OpenNMS was a typical network monitoring tool like Nagios. I thought you could just fire it up, tell it what to watch, and wait for your cell phone to ring when something failed. OpenNMS will do that, and more. OpenNMS is designed to address the monitoring of extremely large (e.g., 100,000 machines) networks with many different types of devices. To this end, OpenNMS is a monitoring platform upon which you can build pretty much whatever you want. Also, it ships with a fairly complete prebuilt monitoring solution, so you don't have to do much from scratch unless you want to.

Installation

OpenNMS has several main components: a back-end database (PostgreSQL), a Java-based engine that does all the heavy lifting (monitoring, alerting, etc.), and a Java-based web front end for administering and managing the system, viewing reports, and so on. I won't cover all the details of installation because OpenNMS has an extremely comprehensive installation guide [2] that covers the process in depth.



<http://yum.opennms.org/repofiles/> 
opennms-repo-snapshot-fc12.noarch.rpm

The file name for the Fedora 11 stable version would be `opennms-repo-stable-fc11.noarch.rpm` and so on. Once this step is done, you can simply run `yum install opennms` and go through the configuration process.

A Brief Note on Performance

I spoke with the OpenNMS developers about performance of the system. I asked how many monitoring systems would be needed to monitor large numbers (e.g., tens of thousands) of hosts. The response was that a relatively normal server (basically anything modern with enough RAM to run the Java apps and host the database without thrashing the disks) would be able to handle things. So, for most installations, you can easily get away with one or two servers (probably best to have two in case one fails or needs to be taken offline for maintenance). Also, you can install remote agents; more on this later.

Configuration of OpenNMS

Once you've installed OpenNMS, you can log in to the web interface on port 8980. You'll need to change the admin password, then you can select *Admin* and click on *Configure Discovery*, where you can enter individual services and URLs or network blocks to scan and monitor.

Alternatively, you can edit the `discovery-configuration.xml` file by hand or generate your own using a script if you know your network and IP blocks. OpenNMS also supports exclusion ranges, so you can avoid monitoring things that might not belong to you.

Once you have entered the information in the web interface, just click *Save and Restart Discovery*. OpenNMS will then scan all the hosts you have specified, looking for common services (DNS, HTTP, file sharing, etc.) and adding them to the database. OpenNMS also supports SNMP and will attempt to connect to devices and poll them, which can provide a wealth of information.

Note that, if a node is SNMP capable and read access is given out publicly, OpenNMS will identify that system by the name with which it is configured rather than the DNS hostname or the IP address (assuming reverse DNS lookup fails).

Once you have a list of nodes discovered by OpenNMS, you can start categorizing them: Are they used for production, development, testing? Is it a router, a switch, a server? You can of course add new categories ("payment processing servers" or "database servers"), and a node can belong to more than one category (e.g., "accounting" and "production" or "staffing" and "database").

Alternatively, you can open a category (*Admin | Category | Show*) and select from a list of all the available hosts. You'll need to fill out the asset information; by default, this supports things like serial number, asset number, operating system, location, rack ID,

The basic installation procedure is to install PostgreSQL and Java and then OpenNMS, then configure the whole thing to run at boot time, and that's it.

Choose your Version

The first choice you'll need to make is which version of OpenNMS to run. You have four options here: stable, unstable, testing, and snapshot. Stable is, of course, the latest stable release, and unstable is the latest official development version of OpenNMS.

The testing version is a nightly snapshot of what will be the stable release, and the snapshot release is a nightly snapshot of the development release (if you want bleeding edge, this is the one to choose).

You also have several options for obtaining OpenNMS. You can download the source and install it from scratch; download an RPM, DPKG (for Debian and Ubuntu), or Solaris package; or install it on Mac OS X via Fink. Or, you can grab the JAR file for a Windows installation.

For an RPM-based system, the easiest method is to get and install the stub RPM that contains the repository information. You can do this with the following command:

Availability Over the Past 24 Hours		
Categories	Outages	Availability
Network Interfaces	3 of 14	81.839%
Web Servers	2 of 9	79.989%
Email Servers	0 of 0	100.000%
DNS and DHCP Servers	0 of 6	100.000%
Database Servers	0 of 2	100.000%
JMX Servers	0 of 0	100.000%
Other Servers	2 of 11	90.205%
Total	Outages	Availability
Overall Service Availability	7 of 43	87.393%

Figure 1: My test network isn't all that healthy.



vendor information, authentication information, and comments. Thankfully, once you've entered this information, you won't need to do it again. You can just save and export the data for later use.

Monitoring

By default, OpenNMS runs on a five-minute polling schedule to determine availability (i.e., it checks nodes, services, etc. every five minutes; see Figure 1). The reason for this schedule is simple: A 99.99% uptime guarantee means you get 4.32 minutes (on average) of outage time per month, so anything over five minutes has blown that away. When an outage is detected, OpenNMS starts polling the service every 30 seconds for a period of five minutes; once this process is complete, the outage is assumed to be significant and OpenNMS reverts to a five-minute polling interval for 12 hours.

If after 12 hours the service hasn't been fixed, OpenNMS reduces polling to every 10 minutes for five days. After this point, OpenNMS marks the service as "forced unmanaged" and stops polling it. If you have different service agreement levels (i.e., only 99% uptime, or 99.999% uptime), you can easily modify the `poller-configuration.xml` file accordingly.

OpenNMS can monitor just about everything: SNMP, of course, ICMP ping, and support for most common services (SSH, DNS, HTTP, etc.). You can request web pages (and check the page contents for specific items like price information or a status message), write custom checks, use Nagios plugins directly, and access Windows Management Instrumentation (WMI). You also get response times (Figures 2 and 3).

Notifications

Tracking outages is all well and good, but being notified of them promptly is even better. OpenNMS uses the concept of "Destination paths," which is essentially a list of one or more alerting methods that are used in turn until the notification event is acknowledged (thus indicating that someone is working on it).

This approach allows you to contact multiple people in a given sequence using a variety of methods (including email, IRC, XMPP, phone, and pager messages, as well as sending an arbitrary HTTP request or executing an external program). Thus, you can start by sending an instant message to your admin and escalate to pages and phone calls at home.

The back end of notifications is pretty straightforward. When OpenNMS notices an event, it

generates an SNMP event that is handled internally; by default, many of these events have already been grouped into generic notification events, like `nodeDown` (a node has gone down) and `nodeLostService` (a node loses a service). You can, of course, create your own events (e.g., monitoring for 3Com- or Cisco-related traps).

Remote Monitoring

If centralized monitoring won't work (e.g., you have branch offices sitting behind firewalls), OpenNMS supports remote monitors. These software agents can run on remote systems and check the health of machines and networks then report back to the central OpenNMS servers. With the use of this approach, you can ensure that services are remotely available and detect more fine-grained failures – also known as the "it works for me, it must be your end that's broken" scenario.

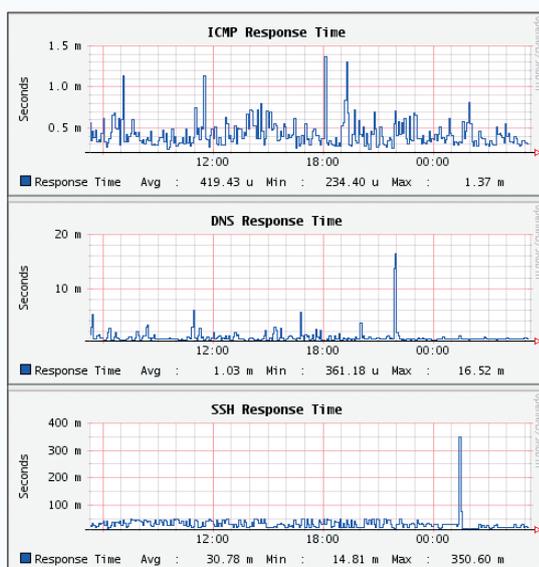


Figure 2: Traffic to and from a printer (not much printing going on).

Critical Paths and Failures

So, what happens when a router dies, disappearing an entire network behind it? You get hit with 15,000 outage notifications, flooding your alerting system (I hope you don't get a page for each one) and generally burying the important event. OpenNMS has a solution for this, too.

By defining a critical path (an IP address and a service) that will be affected by this failure, when the router providing access to that subnet goes down, OpenNMS will not freak out. Unfortunately, ICMP currently is the only critical path service supported directly, service-level critical paths are not yet available (Figure 4).

A potential solution can be found in the form of support for the Drools [3] "Business Logic Integration Platform," which allows you to express rules in XML form and take action on the basis of them. Unfortunately, to configure this, you need to know how to write Drools rules and understand how Drools behaves (the documentation is lacking). Also, to use Drools properly, you would need to figure out all the failure scenarios for your equipment and express them as a rule set; if this were possible, admins wouldn't be spending so much time monitoring and fixing things in the first place.

Monitoring Windows

Windows Management Instrumentation (WMI) [4] is a Microsoft technologies that you've probably never heard about but that is invaluable if you have to administer Windows systems. Basically, WMI allows you to write scripts or applications that

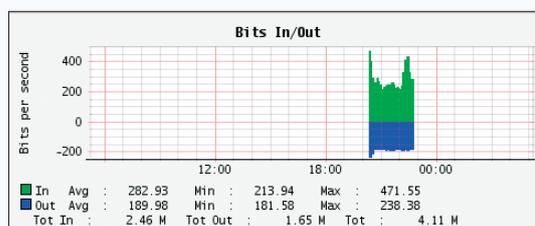


Figure 3: This report shows a critical path with a single server failure.

can automate tasks on remote computers (sort of like Cfengine or Puppet in the Unix world) and can be used to supply management data (i.e., system and application health). Using WMI, you can peer



pretty deeply into the state of a Windows machine (you can even query the BIOS) and, assuming your applications are written with WMI support, you can also check on them. This method allows you to ensure that the application is running smoothly and hasn't stalled or slowed down for some reason. Detailed information on WMI configuration is covered in the OpenNMS wiki [5].

Path Outage Node List	
Critical Path	Status
192.168.17.1	ICMP
Node	Status
192.168.2.1	All Services Up
192.168.2.2	All Services Up
192.168.2.20	All Services Up
192.168.2.101	All Services Down

Figure 4: This shows a critical path with a single server failure.

Custom Monitoring Plugins

Between the default set of plugins and the ability to monitor SNMP and WMI, OpenNMS can provide a fairly in-depth view of your systems. However, if you have legacy applications or a closed source application that doesn't report its health via SNMP or WMI, all is not lost. If you look at the existing plugins in the `netmgmt/pollers/monitors/` directory, you'll see that writing a custom plugin is relatively simple.

These plugins basically test a service in a variety of ways; return a `serviceStatus` code that indicates whether the service is unavailable, unresponsive, or available; and then log the message (e.g., "no route to host" or "TCP connection timed out"). Most of the plugins consist of 200-300 lines of Java.

Alternatively, you can use the "General Purpose Poller," which allows you to call an external script or program to check on a service. This technique allows you to write and use check scripts written in any language supported by your OS and to use command-line utilities that ship with your applications to check on their health (search the wiki for "GeneralPurposePoller").

The Output

With all this back-end monitoring, how can you quickly and concisely view the overall health of your system? Reports and charts, of course. OpenNMS includes a number of default reports (service availability overall, email servers, etc.) and allows you to create new reports as well. Reports can also be scheduled and emailed automatically or created on demand (your manager will love this).

The OpenNMS package also includes some useful eye candy – specifically, the mapping feature. You can generate maps internally or with the use of external services such as Google maps. One of the best features is the ability to display maps with Adobe SVG in full screen with an auto-update feature. If you then put that map on a central monitor, you can easily keep an eye on your entire network.

Support and Documentation

OpenNMS is a full-featured monitoring framework that can do pretty much anything, which is also one of its major challenges. OpenNMS has a very active support community, however. The documentation wiki [5] is comprehensive, the mailing list is helpful, and the IRC channel on FreeNode is populated and polite (and some of the OpenNMS authors spend time there).

All of these features lead directly to the OpenNMS Group, the commercial entity that is largely responsible for OpenNMS. The OpenNMS Group makes money from customization, implementation, and training. And, because they are writing the

software, they in effect provide a final backstop for support and service.

What's Wrong with OpenNMS?

OpenNMS is not perfect, so you might run into some issues. The IPv6 support, for example, is currently ad hoc. Some plug-ins, such as the HTTP service check, do support IPv6 because the underlying Java library supports IPv6; however, this isn't the case for all service plug-ins. More importantly, you can't yet do IPv6-based discovery, so you'll have to add your hosts manually (or import the data; creating the import file isn't too hard). However, widespread IPv6 deployment is a ways off, so the OpenNMS group has some time.

Also, because OpenNMS relies on PostgreSQL and is not yet database portable, it can be a little annoying if you aren't familiar with it. The good news, though, is that virtually all OS vendors either ship PostgreSQL or have an easily installable package for it. And, of course, tuning information can be found in the documentation.

Also, because OpenNMS relies on PostgreSQL and is not yet database portable, it can be a little annoying if you aren't familiar with it. The good news, though, is that virtually all OS vendors either ship PostgreSQL or have an easily installable package for it. And, of course, tuning information can be found in the documentation.

Conclusion

Would I buy OpenNMS? Or, because it is freely available, would I be willing to spend the time to get this software installed and configured correctly? Yes. I previously used Nagios [6] to monitor networks, but I ran into problems when monitoring large numbers of hosts because Nagios fired up the `ping` command when it needed to send an ICMP packet to a host to see whether it was up; this is a non-issue with OpenNMS.

I also like the ability of OpenNMS to escalate alerts and use different contact methods. I like the ease of configuration, too. For a simple setup, you can basically add your network block to the discovery page, put your email address to the notification chain, and you're done. Also, OpenNMS has been under active development for 10 years, so it can compete readily against the major commercial offerings.

Because OpenNMS truly is open source (GPL licensed), you won't get stuck in an expensive bait and switch situation, where the vendor gives out a slightly crippled open source version, but you need to buy a commercial version to get the features or scalability that you need. If you have a network to monitor, especially one made up of Unix, Windows, and SNMP-enabled network devices, OpenNMS is definitely worth checking out. ■■■

INFO

- [1] OpenNMS: <http://www.opennms.org/>
- [2] OpenNMS installation guide: <http://www.opennms.org/documentation/installguide.html>
- [3] Drools: <http://www.jboss.org/drools>
- [4] Windows Management Instrumentation: <http://msdn.microsoft.com/en-us/library/aa394582%28VS.85%29.aspx>
- [5] OpenNMS documentation wiki: <http://www.opennms.org/wiki/>
- [6] Rootdev: OpenNMS vs. Nagios: <http://www.rootdev.com/tech/opennms-vs-nagios>