OpenNMS

Marcin Rybacki
OpenNMS

Plan
What is OpenNMS?
Requirements
Discovery
Data collection
Events

Definitions:

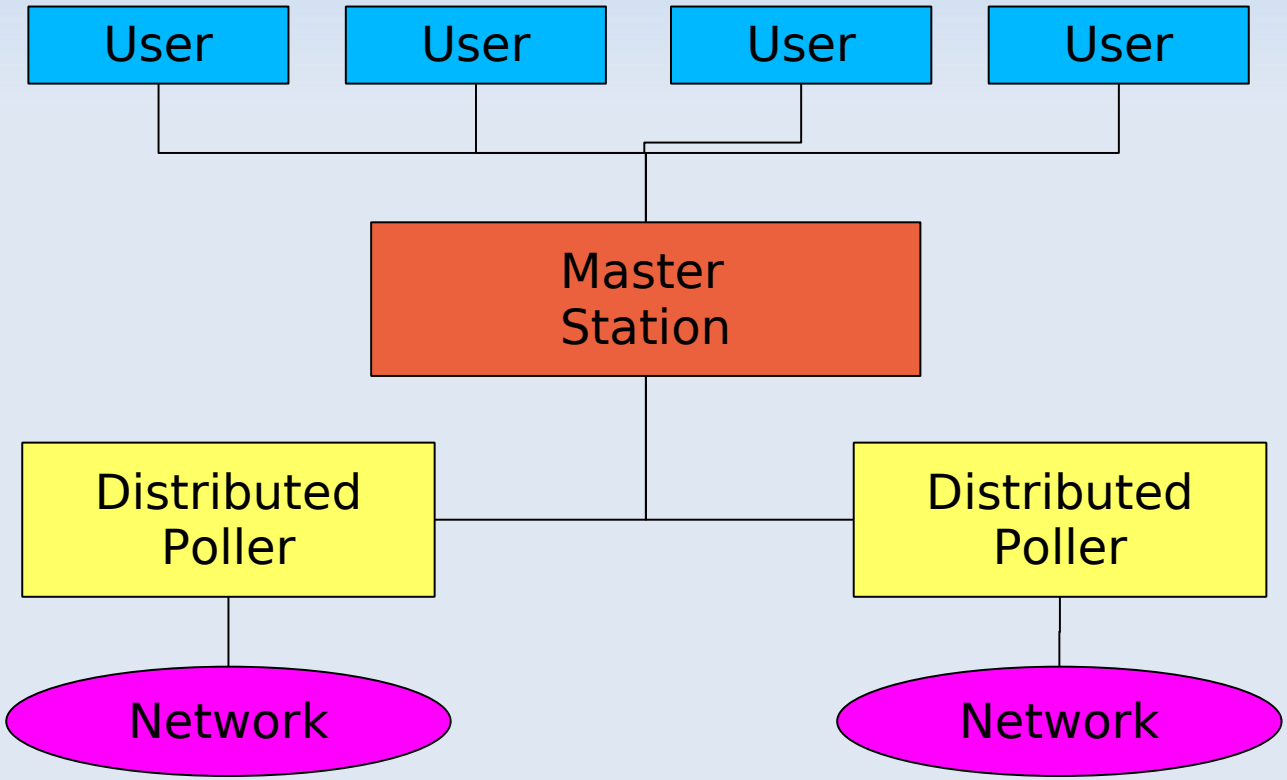
- Network Management
  - Managing TCP/IP addressable devices and the things that make them work
- Systems Management
  - Agent-based solutions, monitoring the health and performance of systems (servers, desktops, et al)
- Applications Management
  - Extensions to agent-based solutions that address application specific monitoring and management

Therefore…

- Enterprise Management combines:
  - Network Management
  - Systems Management
  - Application Management
  - Other disciplines

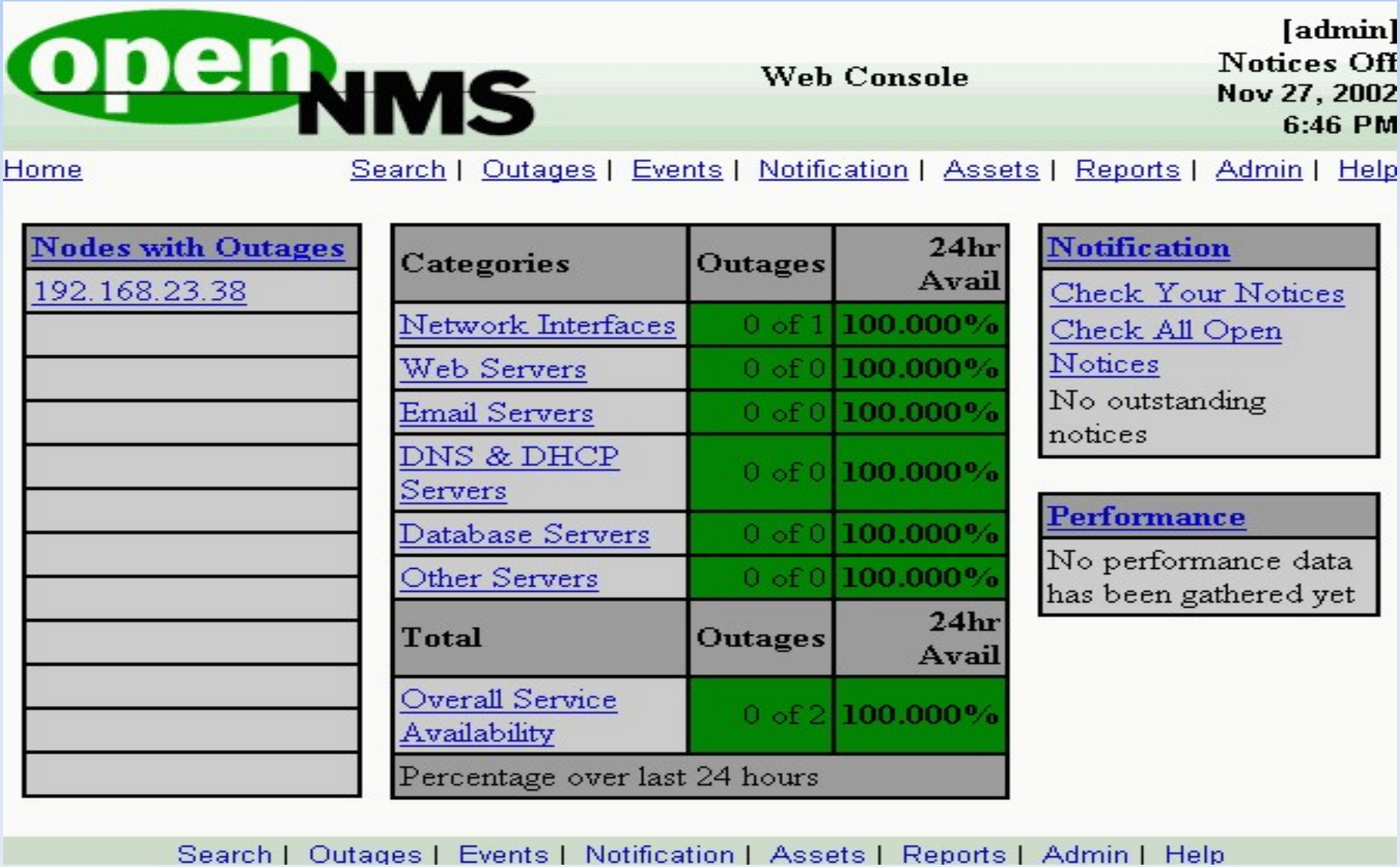
- A network/systems management platform
  - Network node and service discovery
- Ongoing polling for availability at the IP level, the application level, or both.
- Open source; Licensed under the GPL/LGPL
  - Source code available today
  - 7x24x365 support available 2Q01 ($)

Architecture



There are actually two main applications in the OpenNMS product: the application itself and the web-based User Interface (webUI). The webUI is implemented via Tomcat, and it is possible for Tomcat to be running and the OpenNMS application to be stopped and vice versa.

Przykładowy screen



Currently, OpenNMS focuses on three main areas:

Service Polling - determining service availability and reporting on same.
Data Collection - collecting, storing and reporting on network information as well as generating thresholds.
Event and Notification Management - receiving events, both internal and external, and using those events to feed a robust notification system, including escalation.

The OpenNMS Group is the commercial entity that funds the OpenNMS application development.

Wymagania

Requirements;
- A 1 GHZ Pentium III. . OpenNMS can also take advantage of multiple processors.
- 256 RAM (512MB strongly recommended)
- 25 MB of disk space for the program files, data for one interface takes 2 MB of disk space, so for 200 interfaces you are looking at 400 MB (conservatively).

A: As OpenNMS is written mainly in Java, it can theoretically run on any system that supports a 1.4 SDK.

Currently, the following Operating Systems are Supported with up-to-date builds:

* Linux
    - o RHEL/CentOS (3 and 4)
    - o Debian Sarge
    - o Debian Etch
    - o Fedora Core (2, 3, 4 and 5)
    - o Mandrake 9.2 and 10
    - o SuSE (9 and 10)
    - o Red Hat Linux (7, 8 and 9)
* Solaris 8 and Solaris 9 (SPARC)
* Solaris 8 and Solaris 9 (x86)
* Mac OS X (Panther)

No Windows support, as of yet, but with OpenNMS 2.0 it should be available.

Dependencies:
- Java (SDK not the JRE, as Tomcat will need to compile Java code (which requires "javac" in the SDK).
- Tomcat 4 (doesn't work under Tomcat 5)
- RRDtool provides a "round robin" database that stores time-series data quickly and in a small amount of space. OpenNMS stores its performance-related data in RRD files created using RRDtool
- Version 7.2 or later of PostgreSQL. If you are using a version of PostgreSQL prior to 7.4, the server error messages are required to be in English (the 'C' locale).
- The startup script uses curl to connect to the OpenNMS daemon to check that the various components are up and running ("opennms status").

Discovery

$OPENNMS_HOME/etc/discovery-configuration.xml.

```
<discovery-configuration threads="1" packets-per-second="1"
    initial-sleep-time="300000" restart-sleep-time="86400000"
                retries="3" timeout="800">

        <include-range retries="2" timeout="3000">
                <begin>192.168.0.1</begin>
                <end>192.168.0.254</end>
                </include-range>

    <include-url>file:/opt/OpenNMS/etc/include</include-url>

                </discovery-configuration>
```

OpenNMS focuses on the services network resources provide: web pages, database access, DNS, DHCP, etc. (although information on network elements is also available).

The basic monitored "element" is called an "**interface**" (identified by an IP address). nd if If a number of interfaces are discovered to be on the same device (either via SNMP or SMB) then they may be grouped together as a "**node**".

Discovery in OpenNMS consists of two parts:

- **discovering an IP address** to monitor and then
- **discovering the services** supported by that IP address.

Now, all this file controls is a process that will send an **ICMP** "**ping**" to a particular set of IP addresses. If there is a response within the timeout, a "**new suspect**" event is generated. Otherwise, the IP address is **ignored**. (options: threads, packets-per-second, initial-sllep-time, restart-sleep-time, -timeout, retries, ranges, include-range, exclude-range, specific)

Any changes to discovery configuration file, like most of the configuration files within OpenNMS, requires that OpenNMS be restarted. Second, what if you want to discover a service, such as a web server, on a device you cannot ping?

The format of the send-event.pl is as follows:

```
/opt/OpenNMS/bin/send-event.pl --interface ip-address \
    uei.opennms.org/internal/discovery/newSuspect
```

Capabilities

Capabilities daemon, **capsd**. is responsible for discovering all the services to be monitored, such as httpd, DNS, etc., as well as if any collectors are present (at the time this is only SNMP).

The capsd process is controlled by the **capsd-configuration.xml** file. This file consists of some basic parameters and a collection of "protocols" to be tested. If the protocol is not in the file, then OpenNMS will not discover it.

(sample options: rescan-frequency, initial-sleept-time, managed-policy, max-suspect-thread-pool-size, abort-protocol-scans-if-no-route)

OpenNMS tests the existence of a particular network service through the use of "protocols". At the most basic, this could be a connection to a TCP port to test for a particular **banner**, but there are also special classes for a variety of other protocols. The current list is:

* Citrix
* DHCP
* DNS
* Domino IIOP
* FTP
* HTTP
* HTTPS
* ICMP
* IMAP
* LDAP
* Microsoft Exchange
* Notes HTTP
* POP3
* SMB
* SMTP
* SNMP
* TCP

Example - HTTP service:

```
<protocol-plugin protocol="HTTP"
class-name="org.opennms.netmgt.capsd.HttpPlugin"
scan="on" user-defined="false">
<property key="ports" value="80"/>
<property key="timeout" value="3000"/>
<property key="retry" value="2"/>
</protocol-plugin>
```

The snmp-config.xml File

The parameters used to connect with SNMP agents are defined in the snmp-config.xml file. Here is an example:

```
<snmp-config retry="3" timeout="800"
  read-community="public" write-community="private">
  <definition version="v2c">
    <specific>192.168.0.5</specific>
  </definition>

  <definition retry="4" timeout="2000">
    <range begin="192.168.1.1" end="192.168.1.254"/>
    <range begin="192.168.3.1" end="192.168.3.254"/>
  </definition>

  <definition read-community="bubba" write-community="zeke">
    <range begin="192.168.2.1" end="192.168.2.254"/>
  </definition>

  <definition port="1161">
    <specific>192.168.5.50</specific>
  </definition>
</snmp-config>
```

POLLING

How does polling work?
Processes called monitors connect to a network resource and perform a simple test to see if the resource is responding correctly.

The basic idea behind the poller starts with grouping network devices into **packages**. Each package will consist of:

- various **services** and how they are to be polled (i.e. frequency)
- **outage** be detected, each package can have its own downtime model which controls how the poller will dynamically adjust its polling on services that are down.
  - **outage calendar** that schedules times when the poller is not to poll (i.e. scheduled downtime).

What is adaptive polling?

You might as well guarantee "100%" availability since any outage will break your service level agreement.

To help combat this, OpenNMS uses adaptive polling. Once an outage is detected, polling is temporarily increased to try and detect, as soon as possible, when the service is restored.

Data Collection and Configuration

Data collection is done by collectors. Currently, the only collector is for SNMP data.

user-defined

In the future, users may be able to define new collection sources (like from a log file) through a GUI, but at the moment this is set to "false".

status

Also in the future, there will be an admin GUI for collectors just as there is for pollers, and users will be able to turn SNMP data collection on or off from a web page. At the moment, this can only be done by editing this file and setting status to either "off" or "on" (default).

collectd-configuration.xml

Reference to the class that is to be used for this collection. Since there is currently only one collector, there is only one collector service statement setting the class to org.opennms.netmgt.collectd.SnmpCollector

Event-Configuration

OpenNMS manages events through a process called **eventd**. There are two main types of events: those **generated internally by the OpenNMS** software and **those generated via external SNMP traps.** Processes can generate events, such as when the discovery process generates a **newSuspect** event when an interface responds to a ping, and processes can "subscribe" to events, as when the capsd process asks to be notified whenever a newSuspect event occurs so it can begin its capabilities scan.

OpenNMS also comes with a feature rich **Notification** system. Particular events can be chosen to cause a notification to be sent, such as a page or e-mail.

The **eventd** process listens on port 5817, so other processes, even those external to **OpenNMS**, can send events to the system. The <security> tag is there so that these events cannot override the actions defined in the **eventconf.xml** file.

Events – parameters

- The UEI
- The event-label
- descr
- logmsg
- severity

Dziękuję :-)