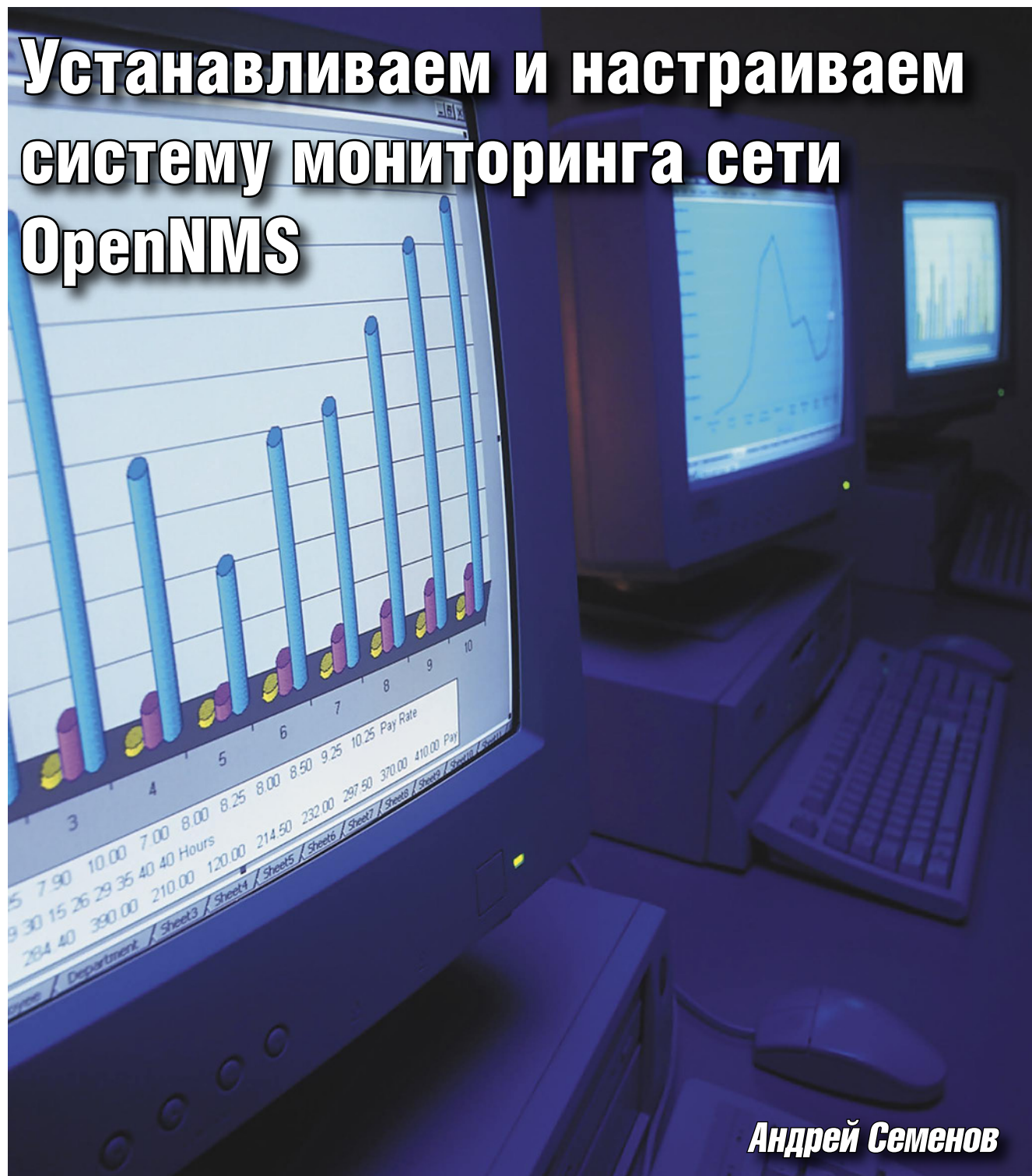


Устанавливаем и настраиваем систему мониторинга сети OpenNMS



Андрей Семенов

Когда количество активного оборудования и серверов в сети неуклонно растет, а контролировать работоспособность и качество предоставляемых сервисов становится все сложнее, на помощь приходит система мониторинга сети OpenNMS.

Кратко о системе мониторинга OpenNMS

OpenNMS [1] – система мониторинга сетевой инфраструктуры уровня предприятия, распространяемая по модели свободного программного обеспе-

чения (Open Source). Кроме обычной для Open Source-проектов поддержки сообществом пользователей, производитель предоставляет многоуровневое коммерческое сопровождение продукта: от внедрения до обеспечения тех-

нической поддержки 24x7 и обучения персонала.

Данная система реализована на Java, поэтому появляется такое положительное качество, как кроссплатформенность.

Поддерживаются ОС:

- Linux:
 - ☑ Debian Etch и Lenny (x86 и x86_64);
 - ☑ Red Hat Enterprise Linux/CentOS (3, 4 и 5; x86 и x86_64);
 - ☑ Fedora Core (версии с 2 по 8, x86 и x86_64);
 - ☑ Mandriva 2007 и 2008;
 - ☑ SuSE (9 и 10).
- Solaris 8, 9, и 10 (SPARC и x86);
- Mac OS X (10.4+, PowerPC и x86);
- Windows 2000, XP, 2003 (Vista и Server 2008 не тестировались).

Теоретически OpenNMS может запускаться на любой платформе, поддерживающей Java SDK 1.4 и выше. Также к положительным качествам можно отнести модульность системы и возможность развертывания частей системы на отдельных серверах (СУБД, демоны сбора статистики и веб-интерфейс могут быть разнесены). Конечно, можно добавить и ложку дегтя: за Java-реализацию, обеспечившую кроссплатформенность, пришлось заплатить увеличением потребления ресурсов.

Система OpenNMS отвечает за мониторинг функционирующих в сетевой инфраструктуре сервисов, таких как Web, DNS, DHCP, сервисы СУБД (Oracle, MSSQL, PostgreSQL и др.), однако информация о состоянии сетевых устройств также доступна.

В системе упрощены способы добавления новых сетевых устройств для мониторинга, и общий принцип работы основан на автоматическом обнаружении (discovery) сетевых устройств. Обнаружение состоит из двух частей – определение интерфейсов (IP-адресов) и определение функционирующих на этих интерфейсах сервисов. Определение интерфейсов осуществляется на основе протокола ICMP (Ping), а определение сервисов – с помощью сборщиков (collectors). На данный момент существует несколько сборщиков, но далее мы подробно рассмотрим сбор статистики, основанный на протоколе SNMP.

Основной единицей мониторинга системы является интерфейс (interface), который уникально определяется на основе IP-адреса. Сервисы (services) привязаны к интерфейсам, а интерфейсы, расположенные на одном устройстве, группируются в узел (node).

Что ж, давайте не будем упускать шанс оценить качество системы и удобство работы с ней самостоятельно, не полагаясь на рекламные строчки и чужие мнения. Приступим к установке.

Установка в дистрибутиве Fedora 8 Linux

Перед началом установки хотелось бы предупредить вас о некоторых тонкостях. Во-первых, для текущей версии OpenNMS 1.5.90 сервер PostgreSQL [4] ветки 8.3.X не поддерживается, так что устанавливаем сервер ветки 8.2. Во-вторых, начиная с версии OpenNMS 1.3.6 в составе дистрибутива отсутствует библиотека `libmpc`, которую необходимо будет загрузить отдельно [2].

Также для работы системы нам понадобится JDK [3]. Уточню, что необходима именно версия JDK (Java Developer Kit), а не JRE (Java Runtime Environment), так как веб-интер-

фейс написан на Java/JSP, а для компиляции JSP в сервлет необходим компилятор Java, который отсутствует в JRE.

Установка JDK

Java Development Kit (JDK) можно скачать с сайта компании Sun [3] либо установить с помощью `yum` из репозитория Tigro [5].

Если данный репозиторий у вас не подключен, то можно установить его через RPM-менеджер:

```
rpm -ihv http://mirror.yandex.ru/fedora/tigro/8/i386/ \
  tigro-release-8-1.i386.rpm
```

После установки репозитория можно воспользоваться `yum`:

```
yum install jdk
```

Хочу упомянуть об одном неприятном моменте: после установки JDK в Fedora 8 необходимо выполнить ряд дополнительных шагов, чтобы заставить работать виртуальную машину. Дело в том, что при попытке запуска любого Java-приложения выдается ошибка следующего вида:

```
java: xcb_xlib.c:50: xcb_xlib_unlock: Assertion `c->xlib.lock' failed
```

Проблема всем давно известная (попробуйте поискать в Google строку с ошибкой) [6] и решается в Fedora 8 следующим образом:

```
sed -i 's/XINERAMA/FAKEEXTN/g' /usr/java/jdk1.6.0_05/ \
  jre/lib/i386/xawt/libmawt.so
```

Возможно, ваш путь к файлу `libmawt.so` будет отличаться от приведенного выше.

Установка PostgreSQL

Если у вас еще не установлен PostgreSQL [4], то установим с помощью `yum`:

```
yum install postgresql-server
```

Теперь выполним начальную инициализацию:

```
/sbin/service postgresql initdb
```

Далее необходимо задать базовые настройки сервера. Для этого нужно отредактировать файлы `pg_hba.conf` и `postgresql.conf`, находящиеся в `/var/lib/pgsql/dat`.

Правим файл `pg_hba.conf`:

```
local all all trust
host all all 127.0.0.1/32 trust
host all all ::1/128 trust
```

Файл `postgresql.conf`:

```
listen_addresses = 'localhost'
```

Еще раз напомним, что это тестовая установка и конфигурационные файлы очень упрощены, проблемы безопасности полностью игнорируются и доступ к серверу БД никак не защищен.

Запускаем сервер:

```
/sbin/service postgresql start
```

и переходим к следующему шагу.

Установка OpenNMS

Есть несколько веток программы для установки: stable, development и nightly snapshot, причем разработчикам рекомендуется использовать ветку development. На момент написания статьи была доступна версия 1.5.90.

Для установки можно скачать установочный jar-файл, но проще будет установить дополнительный репозиторий OpenNMS для yum:

```
rpm -Uvh http://yum.opennms.org/reposfiles/ \
opennms-repo-unstable-fc8.noarch.rpm
```

Данной командой мы установили репозиторий yum для development-ветки OpenNMS. Теперь необходимо установить пакет opennms с помощью yum:

```
yum install opennms
```

Дополнительно необходимо установить пакеты jrrd (опционально вместо rrd), jicmp и iplike:

```
yum install jrrd
yum install jicmp
yum install iplike
```

По умолчанию устанавливается веб-интерфейс opennms-webapp-jetty для встроенного в OpenNMS контейнера сервлетов Jetty, но есть также версия веб-интерфейса для Tomcat и называется opennms-webapp-standalone.

После успешной установки необходимо запустить скрипт поиска установленной виртуальной машины Java:

```
cd /opt/opennms/bin
./runjava -s
```

Если по какой-то причине скрипт отработает некорректно (например, вы установили JDK по нестандартному пути), либо у вас установлено несколько разных JDK и вы хотите указать скрипту конкретную, то можно задать явный путь:

```
./runjava -S /usr/java/jdk1.6.0_5/bin/java
```

Теперь можно запускать скрипт начальной конфигурации:

```
./install -dis
```

Итак, если мы ничего не упустили, то скрипт отработает без ошибок. После завершения установочного скрипта не забываем посмотреть, все ли прошло хорошо:

```
- searching for jicmp:
- trying to load /usr/lib/libjicmp.so: OK
- searching for jrrd:
- trying to load /usr/lib/libjrrd.so: OK
- checking database version... 8.2
.....
```

Должны быть найдены библиотеки jicmp и jrrd. Если что-то пошло не так, то можно явно задать пути поиска:

```
./install -disU -l /usr/lib/jni:/usr/lib
```

Также не забываем, что если мы устанавливаем базу данных OpenNMS на свежую инсталляцию СУБД PostgreSQL, то пароль пользователя postgres пока пустой. Если же у вас уже был установлен сервер PostgreSQL, то вам могут понадобиться опциональные аргументы для скрипта инсталляции:

```
-A,--admin-password <arg>
// Пароль администратора сервера postgres
// (по умолчанию: '')

-a,--admin-username <arg>
// Имя администратора сервера postgres
// (по умолчанию: 'postgres')

-c,--clean-database
// Очистить существующую базу при создании

-D,--database-url <arg>
// JDBC URL базы данных (по умолчанию:
// jdbc:postgresql://localhost:5432/)

-P,--database-name <arg>
// Имя базы данных PostgreSQL (по умолчанию: opennms)

-p,--password <arg>
// Пароль для базы данных opennms
// (по умолчанию: 'opennms')

-u,--username <arg>
// Имя пользователя БД opennms (по умолчанию: 'opennms')
```

Список всех возможных атрибутов можно просмотреть, набрав:

```
./install -help
```

После успешной начальной установки и конфигурации можно запускать OpenNMS:

```
./opennms start
```

Теперь набираем в браузере строку `http://127.0.0.1:8980/opennms/` и вводим имя пользователя «admin» и пароль «admin».

Установка в Windows XP

Установка для Windows [7] не должна вызвать каких-либо затруднений и сводится к запуску файла opennms-installer-1.5.90.jar и следованию указаниям графического установщика, поэтому подробно описывать процесс не имеет смысла. Перед установкой необходимо предварительно установить JDK не ниже 5 версии [3] и PostgreSQL ветки 8.2 (8.3 не поддерживается)[4]. Также отдельно устанавливается библиотека jicmp [2], и путь к ней должен содержаться в переменной окружения PATH.

Описание сетевой инфраструктуры для мониторинга

Для более наглядного описания системы представим себе, что мы настраиваем ее для мониторинга корпоративной сети предприятия с головным офисом и небольшой се-

тью филиалов. Каждый из филиалов объединен с головным офисом в единую сеть (см. рис. 1).

Пусть адресное пространство корпоративной сети будет задано следующим образом:

- **10.10.10.0/24** – адресное пространство головного офиса, выделенное для серверов, маршрутизаторов, коммутаторов, сетевых принтеров и т. п.;
- **10.10.11.0/24** – адресное пространство головного офиса для рабочих станций пользователей;
- **10.10.12.0/24** – адресное пространство офиса №1;
- **10.10.13.0/24** – адресное пространство офиса №2;
- **10.10.14.0/24** – адресное пространство офиса №3.

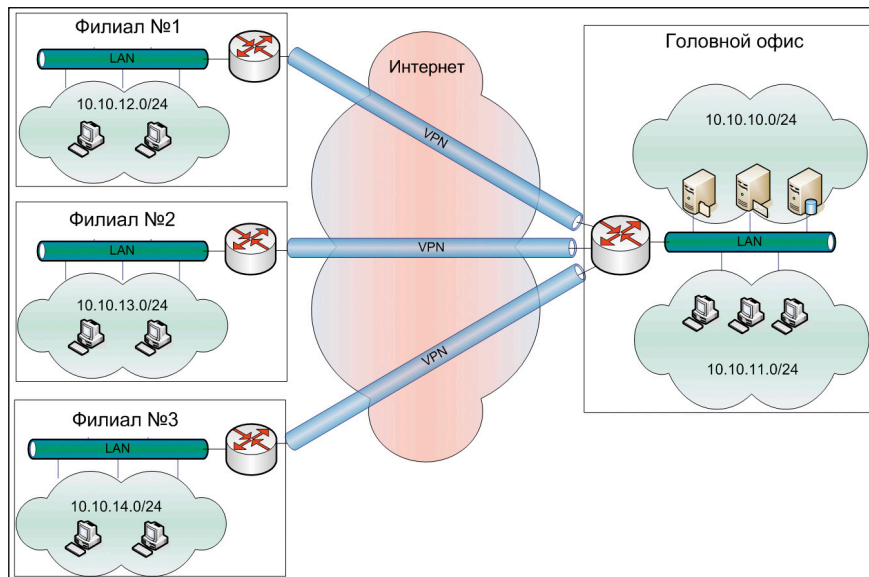


Рисунок 1. Общая схема сети организации с удаленными офисами

Наша сеть состоит из достаточного количества маршрутизаторов (например Cisco 2800 серии в головном офисе и маршрутизаторов попроще – Cisco серий 2600, 1800 и т. п.). На маршрутизаторах настроены SNMP-агенты различных версий, включая третью. Кроме того, имеются также серверы, среди которых терминальные, серверы баз данных, почтовые и т. п., на многих из которых также установлены SNMP-агенты (например, с помощью Net-SNMP).

Настройка диапазонов сетевого обнаружения (Discovery)

Параметры обнаружения устройств описываются в файле `discovery-cofiguration.xml`. Уточню, что все конфигурационные файлы хранятся в каталоге `openNMS-install-path/etc`, где `openNMS-install-path` – путь установки OpenNMS. Ниже представлен пример файла `discovery-cofiguration.xml` для нашего варианта сетевой инфраструктуры:

```
<discovery-configuration threads="1" packets-per-second="1"
  initial-sleep-time="30000"
  restart-sleep-time="86400000"
  retries="3" timeout="800">

  <include-range retries="2" timeout="1500">
    <begin>10.10.10.1</begin>
    <end>10.10.14.254</end>
  </include-range>
  <exclude-range>
    <begin>10.10.11.1</begin>
    <end>10.10.11.254</end>
  </exclude-range>
  <specific>10.10.11.1</specific>
  <include-url>file:/opt/OpenNMS/etc/moreip.txt
  </include-url>
</discovery-configuration>
```

Согласно приведенному файлу, будет осуществляться опрос (посылаться ICMP ping) диапазона адресов 10.10.10.0/24, 10.10.12.0/24, 10.10.13.0/24 и 10.10.14.0/24, а также одиночного IP-адреса 10.10.11.1 и списка IP-адресов, указанных в файле `/opt/OpenNMS/etc/moreip.txt`.

Опишем некоторые глобальные атрибуты по умолчанию, задаваемые в теге `<discovery-configuration>`:

- **threads** – количество потоков для опроса.

- **packets-per-second** – количество генерируемых ICMP-пакетов в секунду. Не стоит указывать слишком большое число, если задержки в вашей сети велики.
- **initial-sleep-time** – время перед стартом процесса обнаружения (в миллисекундах, значение по умолчанию – 5 минут). Задержка необходима для полного старта системы, перед тем как начнут генерироваться события (events).
- **restart-sleep-time** – время после окончания процесса обнаружения, через которое процесс будет повторно запущен (в миллисекундах, значение по умолчанию – 24 часа).
- **timeout** – время ожидания ответа от обнаруживаемого IP-адреса.
- **retries** – количество попыток обнаружения, если с первой попытки ничего не обнаружено.

Глобальные атрибуты можно переопределить внутри вложенных в тег `<discovery-configuration>` тегов.

- **Ter <specific>** – указывает определенный IP-адрес, включенных в процесс обнаружения.
- **Ter <include-range>** – указывает диапазон IP-адресов, включенных в процесс обнаружения.
- **Ter <exclude-range>** – указывает диапазон IP-адресов, исключенных из процесса обнаружения.
- **Ter <include-url>** – определяет файл со списком включенных в процесс обнаружения IP-адресов, один IP-адрес в каждой строке.

Все вложенные теги необязательны, но возможно также их множественное добавление. Так, можно задать несколько диапазонов с помощью `<include-range>`, определить несколько тегов `<specific>` и `<include-url>`.

Итак, при запуске системы OpenNMS по истечении времени `initial-sleep-time` запускается процесс обнаружения интерфейсов. Если получен ответ на ICMP-запрос, то для обнаруженного интерфейса регистрируется событие (event) `newSuspect`, на основе которого в дальнейшем будет осуществлена попытка обнаружения сервисов, функционирующих на данном интерфейсе. А что делать, если ICMP-за-

просы блокируются сетевым экраном? Для таких случаев существует альтернативный метод для регистрации событий newSuspect. В подкаталоге bin установленной системы openNMS находится Perl-скрипт send-event.pl, который можно использовать для самостоятельной регистрации данного события:

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

где «ip-address» – нужный IP-адрес.

Посмотреть на ход процесса обнаружения можно, заглянув в файл discovery.log, находящийся в подкаталоге logs установленной системы openNMS.

Настройка процесса обнаружения сервисов (Capabilities daemon)

После того как зарегистрированы события newSuspect для обнаруженных интерфейсов, приходит время работы демона capsd (capabilities daemon). Его задачей является распознавание всех функционирующих на интерфейсе сервисов. Рассмотрим пример конфигурационного файла capsd-configuration.xml. Конфигурация файла позволяет очень гибко управлять ходом процесса обнаружения сервисов. В первых строчках определяются глобальные параметры функционирования сервиса:

```
<capsd-configuration rescan-frequency="86400000"
  initial-sleep-time="300000"
  management-policy="managed"
  max-suspect-thread-pool-size="6"
  max-rescan-thread-pool-size="3"
  abort-protocol-scans-if-no-route="false">
```

где:

- **rescan-frequency** – время, через которое происходит повторное сканирование интерфейса на предмет обнаружения сервисов.
- **initial-sleep-time** – время задержки после запуска openNMS перед началом сканирования сервисов.
- **management-policy** – контролирует процесс сканирования сервисов. Если установлен в «managed» – то все события newSuspect будут обработаны, если же параметр установлен в значение «unmanaged», то все события newSuspect будут проигнорированы, и сканирование не произойдет. Параметр можно переопределить в теге <protocol-configuration>, о котором будет рассказано дальше.
- **max-suspect-thread-pool-size** – количество одновременных потоков сканирования при первоначальном обходе адресов. Увеличение значения ускоряет сканирование ценой потребляемых ресурсов.
- **max-rescan-thread-pool-size** – количество создаваемых потоков на интерфейсах с уже обнаруженными сервисами при повторном сканировании.
- **abort-protocol-scans-if-no-route** – если параметр выставлен в «false», то попытки сканирования сервисов при получении сообщения «no route to host» продолжатся, так как это сообщение может быть в некоторых случаях вызвано наличием межсетевого экрана. Если параметр примет значение «true», то сканирование интерфейса будет остановлено.

Далее в конфигурационном файле идет описание сервисов (protocols). Определение сервисов основано на установлении TCP-подключения на определенный порт, но есть также специальные классы для некоторых сервисов. Список сервисов, поддерживаемых «из коробки» можно просмотреть в конфигурационном файле (возможно также добавление дополнительных сервисов с помощью тега <protocol-plugin>).

При обработке события newEvent, в случае, если IP-адрес определен как «managed» демон capsd начинает сканирование сервисов в порядке их следования в конфигурационном файле. Первым в конфигурационном файле указан протокол ICMP:

```
<protocol-plugin protocol="ICMP"
  class-name="org.opennms.netmgt.capsd.IcmpPlugin"
  scan="on" user-defined="false">
  <property key="timeout" value="2000"/>
  <property key="retry" value="2"/>
</protocol-plugin>
```

Описание каждого сервиса заключено внутри тега <protocol-plugin>. Данный тег содержит следующие атрибуты:

- **protocol** – название сервиса;
- **class-name** – определяет класс, который будет использоваться для сканирования сервиса;
- **scan** – признак включения сканирования данного сервиса («on» – сканирование сервиса включено, «off» – сервис не будет сканироваться);
- **user-defined** – добавление новых сервисов возможно через веб-интерфейс. В таком случае атрибут примет значение «true».

В каждом сервисе можно указывать дополнительные параметры, определяемые через значения key и value внутри тега <property>. Кроме того, применительно к каждому сервису можно применять дополнительный тег <protocol-configuration>, в котором можно описывать группы IP-адресов, исключенные (или включенные) из сканирования. Например:

```
<protocol-plugin protocol="SNMP" class-name="
  "org.opennms.netmgt.capsd.plugins.SnmpPlugin"
  scan="on" user-defined="false">

  <protocol-configuration scan="off" user-defined="false">
    <range begin="10.10.12.1" end="10.10.14.254"/>
    <property key="timeout" value="4000"/>
    <property key="retry" value="3"/>
  </protocol-configuration>

  <protocol-configuration scan="on" user-defined="false">
    <specific>10.10.11.1</specific>
  </protocol-configuration>

  <property key="timeout" value="2000" />
  <property key="retry" value="1" />
</protocol-plugin>
```

В приведенном выше примере мы переопределили группу адресов (10.10.12.1 -10.10.14.254), для которых определение сервиса SNMP будет отключено, и добавили адрес 10.10.11.1, для которого наличие сервиса будет определяться. Также можно переопределить общие для всех настройки обнаружения, заданные в головном теге <capsd-configuration> с помощью тега <ip-management>:

```
<ip-management policy="managed">
  <range begin="10.10.10.1" end="10.10.14.254"/>
  <include-url>file:/opt/OpenNMS/etc/include</include-url>
</ip-management>
```

где значение атрибута `policy` может принимать значение «`managed`» и «`unmanaged`».

Данные настройки действуют на все указанные в `capsd-configuration` сервисы, если только диапазон адресов для какого-либо сервиса (в нашем случае – SNMP) не переопределен с помощью тега `<protocol-configuration>`.

Необходимо заметить, что если в процессе обнаружения (`discovery`) интерфейс по каким-то причинам не был определен, и не было зарегистрировано событие `newSuspect`, демон `capsd` для такого интерфейса не будет запущен, даже если указать IP-адрес в файле конфигурации `capsd` явно.

Настройка периодичности опросов (polling)

После сбора информации об узлах, интерфейсах и функционирующих на них сервисах приходит время мониторинга. Информация о состоянии сервисов, интерфейсов и узлов собирается двумя основными способами.

Первый способ основан на периодических опросах (`polling`). Процессы-мониторы подключаются к ресурсу и производят простой тест, чтобы определить текущее состояние ресурса. Если ресурс недоступен – генерируется определенное событие.

Настройка периодичности опросов описывается в файле `poller-configuration.xml`. Для удобства введено понятие пакета (`package`) сервисов. В файле можно описать несколько пакетов, в каждом из которых можно указать собственную периодичность опросов, а также определить уникальное подмножество опрашиваемых сервисов. Кроме того, в каждом пакете можно задавать собственную модель действий при недоступности сервиса. Период опроса в случае недоступности сервиса меняется динамически и может гибко настраиваться. Также для каждого пакета можно задать время простоя, когда не будет производиться опрос сервисов. Глобально периоды простоя для всех пакетов можно задать в файле `poll-outages.xml`.

Данные опросов собираются с помощью библиотеки `jrrd` – Java-реализации всем известной RRD (Round Robin Database).

Рассмотрим файл `poller-configuration.xml`:

```
<poller-configuration threads="30"
  serviceUnresponsiveEnabled="false"
  nextOutageId="SELECT nextval('outageNextId') "
  xmllrpc="false">
  <node-outage status="on"
    pollAllIfNoCriticalServiceDefined="true">
    <critical-service name="ICMP" />
  </node-outage>
```

Параметр `threads` определяет максимальное количество потоков для опроса. Варьируется в зависимости от количества опрашиваемых устройств и мощности сервера. Будьте внимательны с данным параметром, при большом количестве опрашиваемых устройств может потребоваться увеличить количество потоков для опроса, так как демон может не успеть опросить все устройства в течение

заданного времени. Узнать, успевает ли демон опросить все устройства и сервисы, можно, заглянув в файл `logs/daemon/poller.log`. В файле нужно найти строку, содержащую максимальное значение параметра `alive`:

```
2008-05-15 17:01:16,201 DEBUG [PollerScheduler-40 Pool]
RunnableConsumerThreadPool$SizingFifoQueue:
adjust: started fiber PollerScheduler-40 Pool-fiber21
ratio = 1.0476191, alive = 21
```

Если значение параметра `alive+1` (так как есть еще родительский поток) близко к значению параметра `threads`, то есть смысл увеличить количество потоков опроса.

Параметр `serviceUnresponsiveEnabled` определяет, какое событие будет генерироваться в случае кратковременного сбоя – «выход из строя» (`outage`) или только «сервис не отвечает» (`unresponsive`).

Чтобы не генерировать событие «выход из строя» при кратковременной недоступности, нужно выставить этот параметр в «`true`».

Тег `<node-outage>` определяет поведение в случае недоступности всех интерфейсов на узле.

Если во время опроса какой-либо сервис на интерфейсе не отвечает, генерируется событие `NodeLostService`, если все сервисы на интерфейсе не отвечают, то генерируется событие `InterfaceDown`. Если параметру `status` присвоено значение «`on`», то в случае недоступности всех интерфейсов узла не генерируется множество событий `NodeLostService` и `InterfaceDown`, а лишь одно событие – `NodeDown`. Во время недоступности узла, в случае, если с помощью тега `<critical-service>` определен критический сервис (в нашем случае ICMP), будет опрашиваться только критический сервис. После того как критический сервис будет восстановлен, начнут опрашиваться все остальные сервисы. Если тег `<critical-service>` не задан, то опрос всех сервисов определяется параметром `pollAllIfNoCriticalServiceDefined`. Если данное свойство имеет значение «`false`», то будет опрашиваться только первый сервис в пакете, иначе – все. Для удобства создадим на основе уже готового пакета `exmaple1` пакет ICMP-PKG и поместим туда единственный сервис ICMP, так как хотим опрашивать устройства по данному протоколу чаще, чем, например, собирать статистику по SNMP:

```
<package name="ICMP-PKG">
  <filter>IPADDR != '0.0.0.0'</filter>
  <include-range begin="1.1.1.1"
    end="254.254.254.254" />
  <rrd step="60">
    <rra>RRA:AVERAGE:0.5:1:89280</rra>
    <rra>RRA:AVERAGE:0.5:60:8784</rra>
    <rra>RRA:AVERAGE:0.5:1440:366</rra>
    <rra>RRA:MAX:0.5:1440:366</rra>
    <rra>RRA:MIN:0.5:1440:366</rra>
  </rrd>
  <service name="ICMP" interval="30000"
    user-defined="false" status="on">
    <parameter key="retry" value="2" />
    <parameter key="timeout" value="3000" />
    <parameter key="rrd-repository"
      value="D:/PROGRA~1/OpenNMS/share/rrd/response" />
    <parameter key="rrd-base-name" value="icmp" />
    <parameter key="ds-name" value="icmp" />
  </service>
  <downtime interval="20000" begin="0" end="300000"/>
  <!-- 20s, 0, 5m -->
  <downtime interval="120000" begin="300000"
    end="3600000"/>
  <!-- 2m, 5m, 1h -->
```

```
<downtime interval="300000" begin="3600000" _J
end="432000000"/>
<!-- 5m, 1h, 5d -->
<downtime begin="432000000" delete="true" />
<!-- anything after 5 days delete -->
</package>
```

Обратите внимание на тег `<filter>`. Внутри пакета он может быть задан в единственном числе. В теге можно задать фильтрацию IP-адресов по маскам. Например, так: `<filter>IPADDR = '10.*.*'</filter>` – будут опрашиваться лишь адреса 10.0.0.0/8. Также можно задавать диапазоны адресов с помощью уже известного тега `<include-range>`, внутри которого задан диапазон IP-адресов для опроса. Напомню, что опрашиваться будут лишь сервисы на интерфейсах, найденных в процессе обнаружения. То есть сервисов на интерфейсе может быть найдено множество, однако сбор статистики будет происходить лишь по сервисам, указанным в пакетах файла `poller-configuration`. Внутри пакета с помощью тега `<service>` может быть определено несколько сервисов (в нашем пакете только сервис ICMP). Рассмотрим атрибуты и вложенные теги тега `<service>`:

Атрибут `status` может принимать значения «on» – статистика собирается и «off» – сбор статистики отключен.

В тегах `<parameter>` с помощью атрибутов `key` и `value` задаются дополнительные параметры, среди которых следует обратить внимание на параметр `rrd-repository`, в котором задается путь для хранения данных опросов. Учтите, что этот параметр связан с параметром `rrd.base.dir` в файле `opennms.properties`. Не забывайте об этом, если будете переносить конфигурационные файлы с сервера на сервер.

В тегах `<downtime>` описывается поведение опроса сервиса в случае его недоступности:

- **Атрибут interval** – задает периодичность опроса при недоступности;
- **Атрибут begin** – время начала заданного периода опроса в миллисекундах;
- **Атрибут end** – время с окончания заданного периода опроса в миллисекундах.

То есть в строке:

```
interval="20000" begin="0" end="300000"/>
```

указано, что в случае недоступности сервиса производить опрос каждые 20 секунд, начиная с нулевой секунды и по достижении 5 минут времени недоступности. Начиная с пяти минут недоступности сервиса можно увеличить интервал опроса до 2 минут, таким образом уменьшив нагрузку на вычислительные ресурсы:

```
<downtime interval="120000" begin="300000" end="3600000"/>
```

а после недоступности сервиса в течение часа увеличить интервал опроса еще больше.

Тег `<rrd>`, в котором задаются параметры сбора и хранения данных, заслуживает более пристального изучения и будет рассмотрен позже.

Второй способ сбора статистики основан на коллекторах (collectors). Он немного сложнее в настройке, но может дать больше информации. Существует возможность

использования нескольких коллекторов, мы остановимся подробнее на сборе статистики с помощью опросов SNMP-агентов.

Настройка сбора статистики по протоколу SNMP

При настройках по умолчанию в OpenNMS 1.5.90 для сбора статистики на основе коллекторов используется протокол SNMP версии 1 и 2(c). Однако существует также возможность использовать протокол SNMP-версии 3. Для активации возможности сбора информации по SNMPv3 необходимо использовать библиотеку SNMP4J [1], поддерживающую протокол SNMPv3 (по умолчанию используется библиотека JoeSNMP). Однако для большинства случаев достаточно SNMPv1 и SNMPv2. Для активации SNMPv3 необходимо раскомментировать строчку:

```
org.opennms.snmp.strategyClass=
org.opennms.netmgt.snmp.snmp4j.Snmp4JStrategy
```

в файле `opennms.properties` и поставить символ комментария перед строчкой с JoeSNMP.

Теперь следует возвратиться к файлу `capsd-configuration.xml` и добавить для удобства новый сервис с названием SNMPv3 (название может быть любым, удобным вам). В будущем это поможет отличать узлы с обнаруженными SNMP-агентами разных версий:

```
<protocol-plugin protocol="SNMPv3"
class-name="org.opennms.netmgt.capsd.SnmpPlugin"
scan="on" user-defined="false">
<property key="force version" value="SNMPv3"/>
<property key="timeout" value="2000"/>
<property key="retry" value="3"/>
</protocol-plugin>
```

Также необходимо добавить сервис с таким же именем в файл `poller-configuration.xml`, если этого не сделать, сервис определится, но опрашиваться не будет. Здесь есть одна тонкость – если не указывать конкретную версию протокола и вообще исключить строчку `<property key="force version" value="SNMPv3"/>` (как это сделано в конфигурации по умолчанию), то будут найдены SNMP-агенты версий, указанных в файле `snmp-config.xml`.

При сканировании сервиса SNMP демон `capsd` обращается за дополнительной информацией в файл `snmp-config.xml`:

```
<?xml version="1.0"?>
<snmp-config port="161" retry="3" timeout="1500"
read-community="public" write-community="private">
<definition version="v3" security-name="myname"
auth-protocol="MD5" auth-passphrase="yourmd5pass">
<specific>10.10.10.1</specific>
<specific>10.10.11.1</specific>
</definition>
<definition version="v1" read-community="charoday">
<specific>10.10.14.1</specific>
</definition>
</snmp-config>
```

В файле находится дополнительная информация для опроса SNMP-агентов.

Корневой тег файла называется `<snmp-config>` и содержит следующие атрибуты:

- **port** – порт, через который идет обращение к SNMP-агенту;
- **retry** – количество попыток подключения к SNMP-агенту;
- **timeout** – время ожидания ответа от агента;
- **read-community** – строка для доступа на чтение к SNMP-агенту;
- **write-community** – строка для доступа на запись к SNMP-агенту;
- **version** – версия SNMP-протокола.

Можно переопределить атрибуты, указанные в корневом теге, указав внутри него тег `<definition>`. В нашем случае в первом вложенном теге `<definition>` мы переопределили версию протокола SNMP. Внутри этого тега также можно определять диапазоны IP-адресов с помощью тегов `<include-range>`, `<specific>` и `<exclude-range>`.

Для определения протокола SNMPv3 существуют дополнительные атрибуты:

- **security-name** – имя для аутентификации;
- **auth-protocol** – протокол шифрования пароля аутентификации (SHA или MD5);
- **auth-passphrase** – пароль аутентификации (шифруется MD5 или SHA);
- **privacy-protocol** – протокол шифрования данных (DES);
- **privacy-passphrase** – пароль шифрования данных.

Протокол SNMPv3 поддерживает три типа аутентификации: `noAuthNoPriv`, `authNoPriv` и `authPriv`.

Если указать все атрибуты – получим тип `authPriv` (самый защищенный тип: шифруются данные авторизации и все передаваемые данные).

Если не указывать атрибуты `privacy-protocol` и `privacy-passphrase`, то получим тип `authNoPriv`, когда в защищенном виде передаются лишь аутентификационные данные, остальные данные не шифруются.

Если указать только атрибут `security-name`, то получим самый незащищенный тип `noAuthNoPriv`.

Подробнее о модели безопасности SNMPv3 (User-Based Security Model – USM и VACM – View-based Access-Control Model) можно найти в RFC3414 и RFC3415.

Обратите внимание, что библиотека SNMP4J [8] ревностно относится к соблюдению стандартов, согласно которым `auth-passphrase` должен быть не менее 8 символов, то же касается и `privacy-passphrase`. Поэтому не советую испытывать судьбу коротким паролем. Подробнее о настройке SNMPv3 на оборудовании фирмы Cisco можно узнать из [9].

Что дальше?

После осуществления всех приведенных выше настроек можно запустить openNMS:

```
./opennms start
```

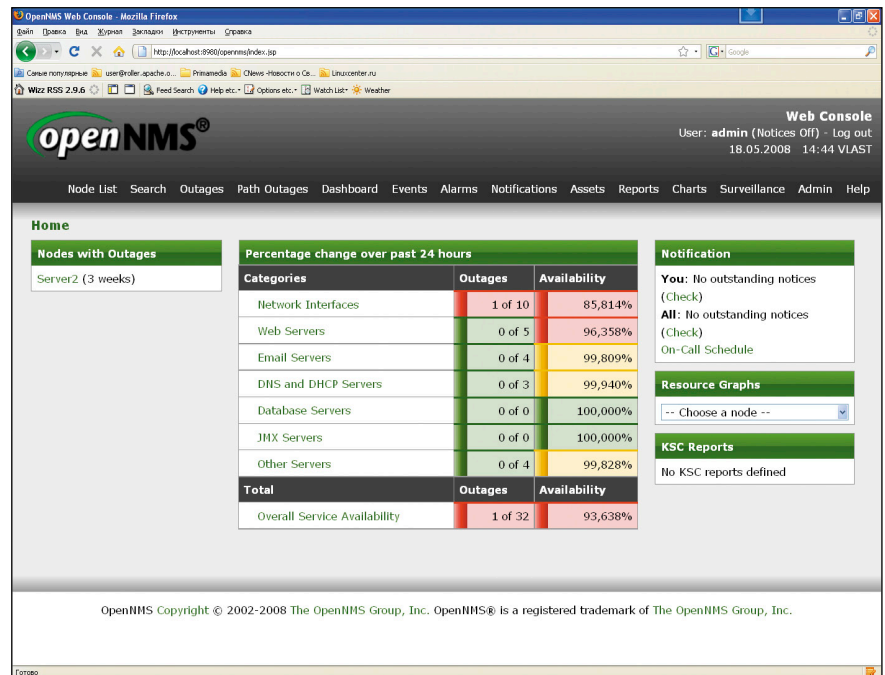


Рисунок 2. Главная страница веб-интерфейса

или для Windows:

```
opennms.bat start
```

Через некоторое время после запуска в списке узлов появятся первые обнаруженные устройства с найденными интерфейсами и сервисами. Это уже работающая система. Через веб-интерфейс мы можем получать информацию о событиях, таких как доступность сервисов и устройств, просматривать графики загрузки каналов связи и ряд других, уже внесенных в типовую конфигурацию графиков. На рис. 2 показана основная страница веб-интерфейса OpenNMS.

Однако многие вопросы дополнительной настройки остались неосвещенными, например, настройка уведомлений о событиях на e-mail, кастомизация веб-интерфейса, отвечающая требованиям вашей сети, тонкая настройка хранения в базах RRD данных, а также настройка параметров отображения графиков и создание отчетов о состоянии сервисов и устройств. Подробнее изучим данные вопросы в следующей статье. 🔄

1. <http://www.opennms.org> – сайт OpenNMS.
2. http://sourceforge.net/project/showfiles.php?group_id=4141&package_id=240236 – ссылка для загрузки библиотеки jicmp.
3. <http://java.sun.com/javase/6/> – ссылка для загрузки JDK6.
4. <http://www.postgresql.org> – СУБД PostgreSQL.
5. <http://tigro.info> – блог Tigro со ссылками на репозитории Fedora.
6. http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6532373 – описание ошибки запуска JVM.
7. http://www.opennms.org/index.php/GUI_Installer#Microsoft_Windows_XP – пошаговая установка в Windows XP.
8. <http://snmp4j.org> – сайт библиотеки snmp4j.
9. http://www.cisco.com/en/US/docs/ios/12_0t/12_0t3/feature/guide/Snmp3.html – настройка SNMPv3 агента на маршрутизаторах Cisco.