Маршрутизаторы NSG

Программное обеспечение NSG Linux 2.0

Руководство пользователя

Часть 4

Приложения и службы IP

Версия программного обеспечения 2.0 build 7

Обновлено 02.06.2016

Москва 2016

АННОТАЦИЯ

Данный документ содержит руководство по настройке и применению маршрутизаторов NSG, оснащенных программным обеспечением NSG Linux 2.0. Документ имеет следующую структуру:

Часть 1. Общесистемная конфигурация.
Часть 2. Настройка физических интерфейсов, портов и сетевых интерфейсов. Обработка трафика Ethernet.
Часть 3. Обработка IP-трафика.
Часть 4. Приложения и службы IP.
Часть 5. Туннелирование и виртуальные частные сети (VPN).
Часть 6. Система обеспечения бесперебойных соединений ***ui*TCP**.
Часть 7. Основные команды и утилиты NSG Linux.

Руководства по применению маршрутизаторов под управлением NSG Linux 1.0 и базового программного обеспечения NSG, а также других продуктов NSG (модемов, мостов и т.п.), содержатся в отдельных документах.

ВНИМАНИЕ

Данное Руководство пользователя предназначено для лучшего понимания процедуры настройки в целом и описывает сущность выполняемых действий — а именно, *что* необходимо настраивать. Рассматриваемые вопросы относятся, как правило, к сущности используемой технологии и являются общими для любых её реализаций, независимо от конкретного производителя и устройства. (Исключением являются вопросы, специфичные для оборудования NSG — таких, как организация пользовательского интерфейса или настройка системы бесперебойных соединений ***ui*TCP**.)

Основной документацией по NSG Linux 2.0 является встроенная справка на борту устройства. Она описывает конкретные команды и параметры настройки — т.е. *как* настраивать функции и возможности, описанные в данном Руководстве. Для просмотра справки по каждому из параметров следует использовать кнопку в Web-интерфейсе или команду _manual (_m) в консольном интерфейсе. Если справка на вашем языке отсутствует, следует установить на устройстве русскую локаль.

ВНИМАНИЕ Продукция компании непрерывно совершенствуется, в связи с чем возможны изменения отдельных аппаратных и программных характеристик по сравнению с настоящим описанием. Сведения о последних изменениях приведены в файлах README.TXT, CHANGES, а также в документации на отдельные устройства.

Замечания и комментарии по документации NSG принимаются по адресу: doc@nsg.net.ru.

ООО «Эн–Эс–Джи»
Россия 105187 Москва
ул. Вольная, д.35
Тел./факс: (+7–495) 727–19–59 (многоканальный)

http://www.nsg.ru/
mailto:info@nsg.net.ru
mailto:sales@nsg.net.ru
mailto:support@nsg.net.ru

§ СОДЕРЖАНИЕ §

Часть 4. Приложения и службы IP

§4.1. Общие сведения о приложениях и службах IP в NSG Linux 2.0

К приложениям и службам IP относятся разнообразные программные процессы, работа которых предполагает обмен пакетами по сети поверх протокола IP третьего уровня. В число этих процессов входят:

— Клиенты различных протоколов, предназначенные для автоматизированной настройки системы NSG или отдельных её компонент. Общесистемные клиенты (напр., серверы DNS) настраиваются, как правило, в узле .system , специфические — в меню соответствующих компонент (напр., клиенты DHCP — индивидуально на каждом порту Ethernet). Данные программные компоненты инициируют исходящие TCP-соединения, датаграммы UDP, ICMP и других протоколов с устройства NSG.

— Серверы, с помощью которых устройство NSG предоставляет удалённый доступ для настройки самого себя, а также услуги автоматизированной настройки других устройств в сети. Все эти услуги настраиваются в меню .services . Данные компоненты открывают на устройстве NSG соответствующие порты TCP или UDP, на которых принимаются входящие соединения и потоки.

— Сетевые утилиты (*ping*, *traceroute* и др.) и клиенты различных протоколов (*telnet*, *ssh* и др.), предназначенные для выполнения администратором различных действий в сети (отладка, управление) с устройства NSG. Данные инструменты администрирования содержатся в узле .tools Web-интерфейса и консольной оболочки nsgsh, а также доступны (только для пользователя root) в виде стандартных команд ОС Linux. Данные программные компоненты инициируют исходящие TCP-соединения, потоки UDP, ICMP и других протоколов с устройства NSG.

§4.2. Клиенты сетевых служб

§4.2.1. Клиент DHCP

Клиент DHCP предназначен для автоматизированной настройки сетевых интерфейсов устройства от стороннего сервера DHCP. Клиент DHCP имеется на интерфейсах с протоколом второго уровня Ethernet — как на работающих по физической среде передачи Ethernet, так и на их эквивалентах для других физических сред: Ethernet-over-E12, Wi-Fi, WiMAX.

Клиент включается на каждом интерфейсе устройства NSG следующей установкой:

```
ifAddress.configurable = "dhcp"
```

В этом случае при каждом рестарте интерфейс рассылает по сети Ethernet широковещательный запрос и принимает все настройки, присланные ему сервером: IP-адрес и маску, адрес шлюза по умолчанию (с метрикой 1), адреса серверов DNS и др. Более тонкая настройка возможна в узле dhcp-options. Здесь возможно отказаться полностью от предлагаемого маршрута по умолчанию или выставить ему бóльшую метрику, выбрать таблицу маршрутизации ядра, в которую будет помещён этот маршрут (для последующего построения маршрутизации на основе установленных правил, см. Часть 3), отказаться от использования предложенных DNS,

Полученный маршрут по умолчанию вставляется в общую таблицу маршрутизации с метрикой 0. Если таких маршрутов оказывается несколько, то результат может быть случайным, в зависимости от того, в каком порядке они создавались. Настройка метрики для маршрута, полученного по DHCP, и некоторых других принимаемых параметров, возможна в подузле dhcp-options .

При переходе интерфейса в состояние down все связанные с ним изменения (как его собственный IP-адрес, так и изменения в таблице маршрутизации, списке серверов DNS и т.п.) удаляются.

ПРИМЕЧАНИЕ Клиент DHCP, по определению, не имеет заранее никакой информации о структуре сети и принимает первый же ответ, который к нему поступит. Если в одной сети Ethernet имеется несколько серверов DHCP, результат будет непредсказуемым, поэтому сервер должен быть всегда единственным.

§4.2.2. Клиент DNS

Клиент DNS преобразует доменные имена, используемые локально на устройстве NSG, в соответствующие IP-адреса. Если он включён, то в командах на этом устройстве можно использовать имена удалённых хостов, если нет — то только их IP-адреса. Для работы других устройств в сети он не требуется, за исключением случая, когда устройство NSG исполняет функцию DNS-прокси по отношению к ним.

Настройка клиента DNS производится в узле .system.dns-client . Клиент включается автоматически, если заданы IP-адреса одного или нескольких сторонних серверов DNS. Серверы могут назначаться любым из следующих способов:

— Статически с помощью списка .system.dns-client.name-servers .

— Динамически по протоколу IPCP для соединений по PPP и производным от него протоколам (PPPoE, PPTP) в качестве клиента. По умолчанию, клиент PPP в NSG Linux 2.0 принимает адреса DNS, переданные ему удалённой стороной. Подробно о сеансовых соединениях PPP см. Часть 2.

— Динамически по протоколу DHCP для интерфейсов Ethernet и Ethernet-подобных (Ethernet-over-E12, Wi-Fi, WiMAX).

Статические DNS назначаются администратором вручную и имеют наинизший приоритет. Если устройству NSG дополнительно назначаются динамические DNS, то они ставятся в начало списка в том порядке, в каком они становятся известны системе (и удаляются по мере разрыва соответствующих соединений). Иначе говоря, в этом случае используются DNS от того интерфейса, который последним перешёл в состояние up и находится в этом состоянии в текущий момент.

Другие параметры в узле .system.dns-client предназначены для тонкой настройки взаимодействия клиента DNS с серверами. В большинстве случаев пригодны их значения по умолчанию.

§4.2.3. Клиент NTP

Клиент NTP устанавливает время на устройстве NSG по стороннему серверу точного времени. Настройка клиента производится в узле .system.ntp , обязательным параметром является имя или адрес сервера NTP. По полученному времени устанавливается текущее системное время, а также время аппаратных часов (Real-Time Clock, RTC), если они имеются в устройстве.

Если пользователь не имеет каких-либо особых на то соображений, то для большинства применений можно использовать национальный пул серверов, входящий в систему ntp.org (настройка по умолчанию). В крупных корпоративных сетях, особенно при высоких требованиях к их надёжности и безопасности, как правило, целесообразно синхронизировать все без исключения сетевые устройства от единого корпоративного сервера точного времени, а уже этот сервер — от вышестоящего публичного сервера или иного источника.

Использование клиента NTP необходимо или настоятельно рекомендуется в следующих случаях:

— В больших сетях — для синхронизации времени на всех устройствах сети. Такая синхронизация требуется для отладки работы сети, поиска неисправностей и анализа других процессов в сети.

— На устройствах, не имеющих встроенных аппаратных часов (серия NSG–600). При выключении питания системное время на этих устройствах устанавливается в 00:00 1 января 2010 года. При программном рестарте устройства текущее системное время сохраняется.

ВНИМАНИЕ Корректная установка текущего времени абсолютно необходима, если используются сертификаты X.509 для подсистем IPsec, SSL и др. Поскольку сертификаты имеют ограниченный срок действия (как сверху, так и снизу), при неправильной дате установка защищённых соединений будет невозможна.
Как следствие, если в устройстве нет аппаратных часов, то связь с сервером точного времени не должна производиться через соединения, защищённые сертификатами. В случае крайней необходимости, возможно сначала грубо установить время по серверу, доступному через незащищённое соединение, а затем перенастроить клиента NTP с помощью скриптов службы netping (см. §4.4.4), контролирующей защищённое соединение.

Во всех других случаях синхронизация времени по протоколу NTP также желательна.

§4.2.4. Клиент служб Dynamic DNS

Служба Dynamic DNS (DynDNS, DDNS) позволяет хосту, IP-адрес которого априори неизвестен, зарегистрироваться на специализированном сервере DNS (для входа на сервер клиент аутентифицируется с помощью имени и пароля) и установить соответствие между своим именем и своим текущим IP-адресом. Сервер рассылает по системе DNS сообщения о разрешении доменного имени данного хоста в его текущий IP-адрес. Эти обновления DNS имеют статус Dynamic и распространяются глобально в течение времени, не превышающего 5 мин. После этого любые третьи хосты могут определить текущий IP-адрес данного хоста по его имени.

Использование Dynamic DNS актуально для устройств, динамически получающих IP-адреса при каждом подключении к Интернет или корпоративной интрасети. В первую очередь, это относится к сеансовым соединениям PPP dial-up, PPP по сотовым сетям в пакетном режиме (GPRS, CDMA, 3G), PPPoE по городским сетям Ethernet и линиям ADSL, и др. Удалённые филиалы организации, подключённые таким образом, с помощью DynDNS становятся полноправными узлами корпоративной сети, доступными для обращений со стороны центрального офиса.

ПРИМЕЧАНИЕ Услуга Dynamic DNS по существу предназначена только для серверов, использующих глобальные адреса данной сети, т.е. подключённых к ней напрямую, без использования NAT на вышестоящих узлах. У поставщиков услуг такая услуга обычно имеет название типа Real IP (т.е. адрес в Интернет реальный, но динамический).

В качестве серверов DynDNS могут использоваться различные платные и бесплатные серверы, предлагающие свои услуги в Интернет. Для корпоративных приложений, как правило, предпочтительно — с точки зрения надёжности и безопасности — организовать собственный сервер DynDNS на границе своей интрасети.

По существу, услуга DynDNS организуется с помощью двух (в общем случае) серверов: один используется для установления текущего IP-адреса данного хоста, по которому он виден в Интернете в данный момент, а другой — это непосредственно сервер DNS, который создаёт динамическую запись и рассылает её всем другим серверам DNS. Обе процедуры на практике реализуются с помощью HTTP-запросов и соответствующих скриптов для их обработки. Формат этих запросов, в общем случае, не регламентирован стандартами и является специфичным для того или иного сервера. Программное обеспечение NSG Linux содержит типовые настройки для наиболее популярных поставщиков услуг Dynamic DNS, а также предусматривает построение системы с настройками, отличными от типовых (например, собственной системы DynDNS в корпоративной сети).

Настройка DynDNS производится в узле меню .services.dynamic-dns . На устройстве могут быть созданы одновременно несколько клиентов, работающих с различными доменными именами. Основным параметром клиента DynDNS является имя системы. Для типовых систем (таких, как dyndns.org) оно сразу определяет набор остальных параметров. В случае использования собственной системы DynDNS, неизвестной заранее, требуется вручную указать имена либо IP-адреса обоих серверов (определения IP-адреса и собственно DNS), номера портов TCP на них (опционально) и скрипты для работы с ними.

Для регистрации на сервере DynDNS необходимо иметь имя и пароль пользователя, а также имя хоста, для которого создаётся данная запись DNS (оно зафиксировано в учётной записи пользователя на сервере).

Работающий клиент DynDNS регулярно проверяет свой IP-адрес. Интервал проверки может составлять от 10 сек. до 10 суток. Если при очередной проверке обнаруживается изменение адреса, то происходит обновление информации на сервере DynDNS. Кроме того, информация на сервере периодически обновляется принудительным образом, даже при неизменном IP-адресе, чтобы поддерживать запись DNS в статусе активной.

ПРИМЕЧАНИЕ Вместо регулярной проверки изменения IP-адреса, или наряду с ней, для поддержания информации DynDNS в максимально актуальном состоянии можно использовать принудительный рестарт клиента DynDNS при изменении статуса IP-интерфейса (например, при разрыве и переустановлении PPP-соединения). Такой рестарт может быть реализован с помощью механизма event-handler (см. §4.7).

§4.3. Серверы сетевых служб

§4.3.1. Стандартные отладочные службы

В составе NSG Linux 2.0 имеются стандартные службы *chargen, echo, discard* и *daytime*, предназначенные для отладки сети. Данные службы могут работать как по TCP, так и по UDP. Если эти службы включены, они открывают на устройстве порты с соответствующими стандартными номерами.

§4.3.2. Сервер DHCP

Сервер DHCP предназначен для автоматической настройки хостов в сетях LAN и WLAN, подключённых к устройству NSG. (В более общем случае — любой сети второго уровня, включая удаленные сегменты.) При включении хост-клиент рассылает по сети широковещательный запрос, содержащий единственную доступную ему на данный момент информацию — его собственный MAC-адрес в качестве адреса источника. Сервер DHCP, получив такой запрос, посылает на этот MAC-адрес ответ с настройками, предписанными для данного хоста индивидуально или по умолчанию. Получив ответ, хост настраивает свой IP-интерфейс согласно принятой информации, и далее он готов к работе по протоколу IP. Кроме того, если клиент представляет собой бездисковую станцию, то он получает IP-адрес сервера BOOTP или TFTP, на котором находится загрузочный файл для него, имя файла, и загружается по сети.

На устройстве одновременно могут работать несколько серверов DHCP, каждый из которых обслуживает клиентов в сети, подключённой к определённому сетевому интерфейсу. Настройка службы DHCP в целом и отдельных серверов, входящих в её состав, производится в узле .services.dhcp .

Именем сервера должно являться имя интерфейса, на котором он работает. В конфигурации каждого сервера необходимо указать, как правило, следующий минимальный набор параметров, передаваемых клиентам:

— Пул IP-адресов (назначается в виде диапазона от начального до конечного адреса).
— Маска IP-сети.
— IP-адрес шлюза по умолчанию (в большинстве практических задач им является само устройство NSG).
— Адреса серверов или ретрансляторов DNS (кроме технологических сетей, где клиенты обращаются только к конечному числу технологических серверов, например, банкоматы — к процессингу, по заранее известным IP-адресам). Во многих ситуациях, таких как сети малых офисов, функции ретранслятора DNS может исполнять само устройство NSG (см. след. параграф).

Адреса и маска, назначаемые клиентам, должны быть согласованы с IP-префиксом сетевого интерфейса NSG, чтобы устройства находились в одной IP-подсети.

В большинстве типов современного оборудования, предназначенного для работы в IP-сети в качестве конечных хостов, сетевой интерфейс по умолчанию настроен именно для автоматического конфигурирования по DHCP. Таким образом, настройка одного сервера DHCP позволяет избежать ручной настройки большого числа клиентов (например, ПК в локальной сети). Она также позволяет разделить обязанности настройки конечного хоста и сетевого устройства между различными лицами: например, за устройство доступа NSG полностью отвечает сетевой администратор, а за подключённый к нему банкомат — специалист по банкоматам, и ни один из них не имеет доступа к конфигурации "чужого" устройства.

Параметр max-leases (суммарный для всех DHCP-серверов, определённых в устройстве) позволяет ограничить максимальное число динамических IP-адресов, выдаваемых одновременно. Это число может быть меньше, чем размер пула. В таком случае число клиентов, работающих в каждый момент времени, будет ограничено, но сервер имеет возможность сохранять за клиентами, неактивными в данный момент, их прежние IP-адреса для последующих сессий.

Другие параметры сервера DHCP и ретранслятора DNS могут быть заданы в поле extra . Обе службы реализованы на основе одного проекта dnsmasq. Подробно о синтаксисе командной строки dnsmasq см. документацию по проекту в первоисточнике на сайте http://www.thekelleys.org.uk/dnsmasq/ .

§4.3.3. Ретранслятор (прокси) DHCP

Ретранслятор (прокси) DCHP — более простая служба, которая не назначает клиентам настройки самостоятельно, а отлавливает широковещательные запросы DHCP в одной из непосредственно подключённых сетей и ретранслирует их либо широковещательно в другую непосредственно подключённую сеть, в которой должен находиться фактический сервер, либо на заданный IP-адрес или имя сервера. При этом в поле IP-адреса источника в запросе подставляется адрес того интерфейса устройства NSG, с которого этот пакет отправляется дальше. Сервер может даже находиться не в непосредственно подключённой сети — но в этом случае ему должен быть известен маршрут к данному адресу, чтобы его ответ мог дойти обратно до прокси. Прокси, получив ответ, удаляет из него (заменяет нулями) свой собственный IP-адрес назначения и ретранслирует пакет в исходную сеть по известному MAC-адресу вопрошающего хоста.

Как и для сервер DHCP, прокси может работать независимо на нескольких интерфейсах; именем каждой отдельной прокси является имя входного интерфейса, т.е. того, на котором она слушает запросы DHCP. Обязательным параметром для DHCP-прокси является имя выходного интерфейса — того, через который эти запросы ретранслируются дальше в сторону сервера. Адрес фактического сервера DHCP является опциональным. Если он указан, запросы отправляются на него, если нет — ретранслируются широковещательным образом в сеть, находящуюся за указанным выше выходным интерфейсом.

§4.3.4. Ретранслятор (прокси) DNS

Служба DNS-прокси позволяет устройству NSG выполнять функции сервера DNS по отношению к хостам локальной сети. Клиенты направляют свои запросы DNS ему, а оно переадресует эти запросы некоторым реальным серверам DNS. Ответы серверов ретранслируются клиентам.

Основной практический смысл данной службы состоит в том, что адреса серверов DNS могут быть заранее не известны, например, назначаться поставщиком услуг каждый раз при установлении сеанса по PPP или DHCP. Более того, на практике они могут время от времени изменяться по усмотрению оператора, поскольку его сеть со временем изменяется и модернизируется. В этом случае для клиентов локальной сети удобно назначить выходной маршрутизатор (устройство NSG) одновременно и сервером DNS, а оно уже будет автоматически получать адреса реальных серверов.

На устройстве одновременно могут работать несколько DNS-прокси, каждая из которых обслуживает клиентов в сети, подключённой к определённому сетевому интерфейсу. Настройка службы DNS-прокси в целом и отдельных прокси, входящих в её состав, производится в узле .services.dns . Именем прокси должно являться имя интерфейса, на котором она работает.

Адреса реальных серверов DNS, к которым будут обращаться все прокси, могут быть заданы статически параметрами .services.dns.dns1 и .services.dns.dns2, но это не является обязательным. Если эти параметры не заданы, то прокси обращаются к серверам, которые в данный момент известны локальному клиенту DNS — в том числе, к серверам, полученным динамически по PPP или DHCP. Подробно о работе клиента DNS см. §4.2.2.

Другие параметры сервера DHCP и ретранслятора DNS могут быть заданы в поле extra . Обе службы реализованы на основе одного проекта dnsmasq. Подробно о синтаксисе командной строки dnsmasq см. документацию по проекту в первоисточнике на сайте http://www.thekelleys.org.uk/dnsmasq/ .

Пример совместной конфигурации PPP, DCHP-сервера и DNS-прокси. Устройство NSG–605HF используется для подключения малого офиса к Интернет через оператора сети UMTS (в просторечии — 3G). Внутри офиса создаётся локальная сеть, в данном случае — беспроводная. Для пользовательских ПК, расположенных в этой сети, устройство NSG является одновременно сервером DHCP, шлюзом по умолчанию и сервером DNS. Само устройство в ходе подключения к сотовому оператору узнаёт адреса вышестоящего шлюза и реальных серверов DNS.

```
port
: 3g
: : encapsulation          = "ppp"
: : : ppp
: : : : main
: : : : : set-defaultroute  = true
: : : : : ipcp
: : : : : : accept-dns      = true          (установлено по умолчанию)
.....................................................
: wlan0
: : mode                   = "access-point"
: : adm-state              = "up"
: : ssid                   = "myofficewlan"
: : security
: : : wpa
: : : : passphrase         = "qwertyuiop"
: : : : wpa2                = true
: : ifAddress
: : : prefix               = "172.16.0.1/16"
services
: dhcp
: : wlan0
: : : adm-state            = "up"
: : : dns1                 = "172.16.0.1"
: : : gateway              = "172.16.0.1"
: : : ip-address-pool                (нижеприведённый пул установлен по умолчанию)
: : : : from               = "172.16.0.2"
: : : : to                 = "172.16.255.255"
: : : mask                 = "255.255.0.0"
: dns
: : wlan0
: : : adm-state            = "up"
```

§4.3.5. Запуск пользовательских служб на устройстве NSG

Помимо наиболее важных служб, определённых в конфигурации устройства, пользователь может запускать на устройстве NSG в качестве системной службы (демона) любые другие программы и скрипты. Настройка пользовательских служб производится в узле .services.daemons .

Основным параметром, определяющим сущность пользовательской службы, является command — команда ОС Linux для запуска требуемой программы (если она имеется в системе) или скрипта (скрипты могут создаваться пользователем по мере необходимости). Вывод службы сохраняется в журнале. Объём, формат вывода и т.п. настраиваются параметрами этой же командной строки.

Служба исполняется в изолированной командной оболочке внутри ОС Linux. Вводить обрамление #!/bin/sh (...)& для этой цели не требуется, оно добавляется к команде автоматически.

§4.4. Доступ к устройству NSG, мониторинг и автоконфигурирование

§4.4.1. Сервер Telnet

Доступ по Telnet является общепринятым и простейшим способом управления сетевыми устройствами (за исключением консольного порта, который ныне является вымирающим видом и по этой причине в NSG Linux 2.0 не является основным способом). По умолчанию, сервер включён на стандартном порту TCP 23, а порт eth0 устройства имеет предустановленный IP-адрес 192.168.1.1/24. По этому адресу к устройству можно подключиться для выполнения его первоначальной настройки.

В узле меню .services.telnet можно включить/выключить сервер Telnet, используемый для управления, а также изменить номер порта, на котором он работает.

Доступ по Telnet является незащищённым, поэтому использовать его в рабочих системах не рекомендуется. Как максимум, его можно использовать внутри защищённого периметра корпоративной сети, или через безопасные туннели. В большинстве практических задач целесообразно после выполнения первоначальной настройки отключить сервер Telnet и включить вместо него безопасный протокол доступа SSH (см. след. параграф).

Если доступ по Telnet требуется по существу, но только через определённые порты, то необходимо закрыть используемый порт TCP фильтрами на всех портах, на которых он не требуется. Пример: устройство NSG–605, порт eth0 подключён к локальной сети офиса, порт eth1 — к поставщику услуг общего пользования. Следующая настройка запрещает доступ по Telnet через порт eth1:

```
ip
: filter
: : INPUT
: : : 1
: : : : protocol              = "tcp"
: : : : in-interface          = "eth1"
: : : : destination-port      = "23"
: : : : target                = "DROP"
```

Более сильный вариант того же фильтра: запрещается доступ по Telnet через все сетевые интерфейсы, *кроме* eth0:

```
ip
: filter
: : INPUT
: : : 1
: : : : protocol              = "tcp"
: : : : in-interface          = "!eth0"
: : : : destination-port      = "23"
: : : : target                = "DROP"
```

Подробно о настройке фильтров см. Часть 3.

§4.4.2. Сервер SSH

Протокол SSH, как и Telnet, обеспечивает доступ к устройству в консольном режиме, но при этом обеспечивает защиту всех передаваемых данных — в первую очередь, имени и пароля пользователя. Для использования SSH необходимо выполнить следующие предварительные действия:

— Назначить пользователям реальные непустые пароли. В заводской конфигурации для пользователя nsg установлен пустой пароль, а для пользователя root — случайный хэш, пароль от которого неизвестен никому (в т.ч. и компании NSG). Управление пользователями (назначение паролей, создание/удаление дополнительных пользователей) производится в узле .system.users .

— Сгенерировать на устройстве ключи и сертификаты с помощью команды .services.ssh.keygen . Поскольку процедура генерации ключей длительная и ресурсоёмкая (до нескольких минут на младших устройствах), для её выполнения необходимо подтверждение (в Web-интерфейсе - набрать yes в поле ввода).
Ключи можно повторно сгенерировать в любой момент, при этом будет выведено предупреждение, что они уже существуют.

— Включить сервер SSH.

Дополнительно в узле .services.ssh возможно изменить номер порта TCP, на котором работает сервер SSH. В частности, это может быть полезно для некоторого повышения безопасности.

ВНИМАНИЕ Ключи SSH, сертификаты X.509 и другие средства аутентификации хранятся на устройстве в отдельных файлах в директории /etc. При выполнении обновления программного обеспечения в сервисном режиме они удаляются. Чтобы сохранить их, необходимо перед обновлением сделать резервную копию всей конфигурации, а не только конфигурационного дерева. Подробно о структуре файлов конфигурации и способах её резервирования/восстановления см. Часть 1.

Возможность удалённого доступа по SSH непосредственно к устройству целесообразно предусмотреть, как правило, в большинстве практических конфигураций — кроме тех сетевых решений, где абсолютно все излишние порты категорически должны быть закрыты по требованиям безопасности. Это может пригодиться в самый неподходящий момент для устранения отказов, приводящих к потере связи с устройством иными способами (напр., через VPN). Для повышения безопасности можно ограничить список адресов, с которых разрешён доступ, заранее известными адресами данной организации. Пример фильтра, разрешающего доступ по SSH через порт eth1 только из сети 123.45.67.96/28:

```
ip
: filter
: : INPUT
: : : 1
: : : : protocol              = "tcp"
: : : : in-interface          = "eth1"
: : : : source                = "!123.45.67.96/28"
: : : : destination-port      = "22"
: : : : target                = "DROP"
```

Подробно о настройке фильтров см. Часть 3.

Сервер SSH в NSG Linux 2.0 сконфигурирован для работы только по современной версии протокола SSH v2. Доступ по устаревшей и небезопасной версии SSH v1 запрещён. Разрешить его, в принципе, возможно, но это категорически не рекомендуется; все современные клиенты SSH поддерживают версию 2.

Протокол SSH, в отличие от Telnet, предусматривает передачу данных в бинарном режиме, поэтому он позволяет передавать не только текстовые команды и их ответы, а произвольные бинарные данные. В частности, он позволяет копировать бинарные файлы с одного устройства на другое. Этот приём весьма удобен и широко применяется в Unix-системах. Подробно о возможностях SSH см. Часть 7.

§4.4.3. Сервер HTTP/HTTPS

Сервер HTTP обеспечивает работу Web-интерфейса и является основным инструментом для настройки и управления устройством. По умолчанию, сервер включён на порту TCP 80. Настройка сервера производится в узле .services.http .

Протокол HTTP не является безопасным и не рекомендуется для использования вне защищённого периметра корпоративной сети. Для безопасного доступа предназначен протокол HTTPS, представляющий собой передачу HTTP внутри безопасного туннеля SSL/TLS. В NSG Linux 2.0 этот протокол реализуется в виде комбинации HTTP и STunnel. Для включения HTTPS необходимо выполнить следующие действия:

— Включить HTTPS параметром .services.http.https.enable . При этом автоматически создаётся туннель .tunnel.stunnel.tunnels.~HTTPS~ с заранее предопределёнными настройками (внешний порт TCP 443 и др.). Данный туннель не может быть изменён пользователем и автоматически удаляется при выключении HTTPS. Если пользователю требуется HTTPS с какими-либо другими параметрами (напр., нестандартным номером порта, с двусторонней аутентификацией на основе сертификатов, с централизованно выданными сертификатами и т.п.), то такой туннель следует создать и настроить вручную в узле .tunnel.stunnel .

— Рестартовать сервер HTTP (напр., через консольную командную оболочку, или рестартом всего устройства). При запуске сервера генерируется минимальный набор ключей и сертификатов для туннеля.

ПРИМЕЧАНИЕ Сертификат устройства NSG, генерируемый в ходе данной процедуры, является самоподписанным. При первом подключении к устройству по HTTPS будет выдано предупреждение броузера. Для продолжения работы необходимо вручную добавить данный сертификат в исключения (в список доверенных сертификатов).

ВНИМАНИЕ Команда .services.http.https.enable только создаёт конфигурацию туннеля, но не применяет её. После включения/выключения HTTPS необходимо отдельно применить изменения в узле .tunnel.stunnel .

После того, как включён HTTPS, необходимо запретить доступ по HTTP через все порты (или хотя бы через внешние). Для быстрой настройки можно воспользоваться командами deny-/allow-http, создающие и удаляющие следующий фильтр для запрета доступа по HTTP через любой интерфейс:

```
ip
: filter
: : INPUT
: : : 1
: : : : protocol              = "tcp"
: : : : destination-port      = "80"
: : : : target                = "DROP"
```

Данная команда создаёт фильтр, но не включает его автоматически. Чтобы сделанные изменения вступили в силу, необходимо выполнить команду _apply в данном узле При этом доступ к устройству по HTTP будет утрачен; если в данный момент управление осуществляется по HTTP, то эта сессия останется в "подвешенном" состоянии и её необходимо будет снять принудительно командой .system.sessions.close (или дождаться её снятия по таймауту).

Для более избирательного запрета HTTP (например, только по отдельным интерфейсам или по нестандартному номеру порта) следует создать соответствующие фильтры вручную.

ВНИМАНИЕ Чтобы обеспечить доступ к устройству исключительно по HTTPS, выключать сервер HTTP не следует, поскольку он обеспечивает работу обоих протоколов.

Разовая команда renew-cert удаляет имеющиеся ключи и сертификаты туннеля ~HTTPS~. После этой команды следует также рестартовать сервер, чтобы сгенерировать новый набор.

Чтобы полностью отключить доступ через Web-интерфейс (как по HTTP, так и по HTTPS), следует выключить сервер HTTP. Управление устройством в этом случае возможно через консольный интерфейс через Telnet или SSH.

§4.4.4. Служба *netping*

netping — специализированная системная служба, позволяющая регулярно исполнять заданные скрипты или команды и в зависимости от их результатов производить определённые действия. Наиболее употребительной из таких команд является *ping* для проверки доступности заданного узла сети, что и обусловило название службы. Служба может исполняться в двух режимах: ping для относительно простого мониторинга сети в режиме "да/нет" и expert для написания более сложных алгоритмов. В узле .services.netping могут быть сконфигурированы несколько таких служб, предназначенных для разных целей.

В режиме ping основными параметрами команды являются адрес контрольного хоста (destination) и два скрипта, из которых обязательно должен быть задан хотя бы один: failure-script, исполняемый при пропадании *ping*, и restore-script, исполняемый после его восстановления. Оба скрипта могут содержать любые команды и вызывать любые другие скрипты ОС Linux, в том числе и вызов командной оболочки nsgsh в пакетном режиме, и команду reboot для рестарта системы.

ПРИМЕЧАНИЕ Если netping используется для отслеживания каких-либо аномальных ситуаций (действительных или предполагаемых) и рестарта устройства, чтобы привести его в работоспособное состояние, то рекомендуется вместо стандартной команды reboot использовать скрипт nsgreboot. Данный скрипт сохраняет текущую информацию о состоянии устройства в файле /etc/postmortem.dump и затем рестартует устройство. Полученная информация может быть полезна для анализа и поиска проблем.

Говоря более строго, оба скрипта failure-script и restore-script реагируют на *изменение* состояния *netping* из ON (успешно) в OFF (неуспешно) и обратно и исполняются только 1 раз при каждом таком изменении. Начальное состояние *netping* считается неопределённым, поэтому при первом исполнении *netping* обязательно выполняется один из двух этих скриптов.

Если *netping* используется в своём основном, буквальном смысле — для посылки *ping* на заданный контрольный хост, то пакеты ICMP Echo Request посылаются сериями по packets штук в одной попытке. Если хотя бы на один из них получен ответ, то попытка считается удачной, если ни одного ответа — неудачной. Попытки повторяются через каждые interval секунд. Если неудачными оказываются retry попыток подряд, то регистрируется переход из ON в OFF и исполняется failure-script.

Дополнительно может быть указан IP-адрес, который будет принудительно указываться в посылаемых пакетах в качестве источника (source). Адрес должен принадлежать какому-либо из интерфейсов данного устройства. По умолчанию, указывается адрес того интерфейса, через который отправляются пакеты, но в некоторых случаях он не годится для того, чтобы пакеты обрабатывались, маршрутизировались, фильтровались на промежуточных узлах и т.п. желаемым образом (см. пример ниже).

Вместо *ping* может использоваться, в общем случае, любая другая команда. Например, если пакеты ICMP Echo в сети блокируются по каким-либо соображениям, то вместо *ping* можно использовать команду traceroute с опциями -f и -w — она выполняет аналогичный обмен пакетами, но использует протокол UDP и порты, начиная с 33434 (по умолчанию). Альтернативная тестовая команда или скрипт записываются в поле test-script (если оно пустое, то исполняется обычный *ping*, как описано выше). Параметры interval и retry в этом случае

сохраняют свой смысл; специфические параметры для вызова *ping* (packets, destination, source) игнорируются, вместо них следует явно указать в командной строке все параметры для разового вызова теста.

Дополнительный параметр start-delay позволяет задержать начало посылки *ping* при старте системы, а также каждый раз после отработки failure-script. Такая разовая задержка бывает необходима для того, чтобы система успела прийти в рабочее состояние. Она особенно актуальна при работе *netping* через сотовые интерфейсы, поскольку для них требуется значительное время на старт/рестарт модуля и регистрацию в сети. С её помощью можно избежать ложных срабатываний оттого, что *netping* уже запущен, а соединение, которое он должен контролировать, ещё не установлено.

Все действия *netping* регистрируются в журнале данной службы.

Помимо непосредственного исполнения скриптов, *netping* может рассматриваться как один из системных датчиков, генерирующий события failure и restore . Это позволяет включить его в общую структуру обработчика событий по приницпу "событие — действие" (см. п.§4.7). Такое решение предоставляет более широкие возможности (например, использование tcp-уведомлений и сервера централизованного мониторинга), а также может быть несколько более удобным в адиминстрировании.

ПРИМЕЧАНИЕ *netping* является, по существу, вспомогательным механизмом и ни в коей степени не предназначен для полной замены штатных механизмов контроля и управления: контроля аппаратного состояния ON/OFF интерфейсов, LCP Echo для соединений PPP/PPPoE/PPTP, других встроенных механизмов *echo* и *keepalive* для различных сетевых протоколов, протоколов динамической маршрутизации. Использовать его рекомендуется только в тех случаях, когда такие механизмы неприменимы (напр., некоторые сотовые операторы не отвечают на LCP Echo Request), не информативны (напр., при подключении к поставщику услуг Ethernet отказ может произойти выше по сети, и порт Ethernet устройства остаётся в состоянии UP), или работают некорректно (напр., известная особенность в работе OSPF поверх динамически создаваемых туннелей в маршрутизаторах Cisco Systems).

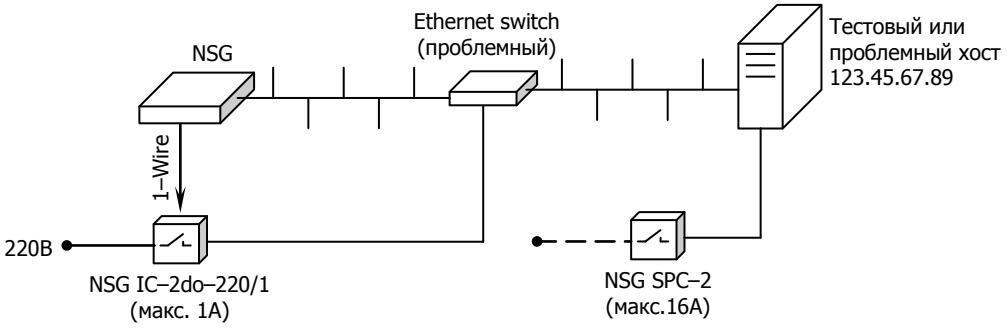
Если в системе используются несколько различных механизмов контроля на разных уровнях, то практическое правило "большого пальца" состоит в том, чтобы время срабатывания каждого вышестоящего механизма было, по крайней мере, в 3 раза больше, чем нижележащего (а также, по возможности, не кратным строго). Например, если устройство устанавливает туннель PPTP поверх сотового соединения, то рекомендуется использовать следующие соотношения времён:

— LCP Echo в сотовом соединении PPP — 3 попытки по 30 сек;
— LCP Echo в контрольном соединении PPTP — 3 попытки по 95 сек;
— *netping* для рестарта всего устройства — 3 попытки по 300 сек.

При работе в режиме expert служба *netping* позволяет реализовать более сложные алгоритмы слежения и реагирования, включая последовательности из нескольких тестов, несколько (более двоичного "да/нет") возможных результатов каждого теста, условные переходы между ветвями алгоритма и т.п. Результат каждого теста передаются в виде кода завершения тестовой программы, в зависимости от которого может выбираться следующий шаг алгоритма.

Примеры:

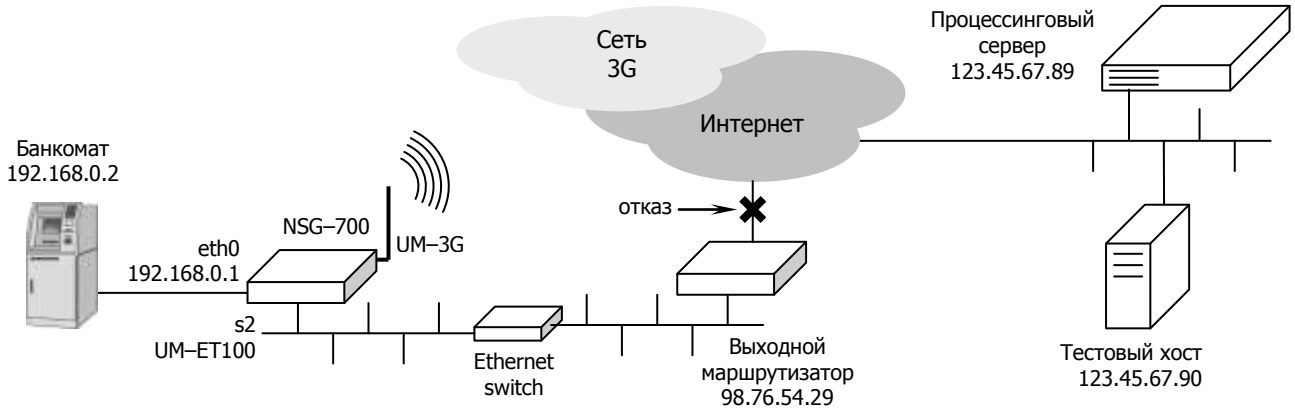
1. Рестарт проблемного устройства. Некоторое устройство в сети может "зависать" и перестаёт отзываться на *ping*. Другой вариант — в сети имеется ненадёжный коммутатор, "зависание" которого приводит к потере связи с целевым хостом. В обоих случаях производится рестарт проблемного устройства по питанию с помощью контроллера NSG IC–2do–220/1 или SPC–2, подключённого к порту 1–Wire устройства NSG. Чтобы не вдаваться во внутреннюю номенклатуру портов в системе NSG Linux на данной модели, скрипт исполняет команды nsgsh в пакетном режиме. Для параметров *ping* используются значения по умолчанию.



```
services
: netping
: : hw_restart
: : : adm-state        = "up"
: : : description      = "Restart this poopy piece of hardware once it hangs"
: : : destination      = "123.45.67.89"
: : : failure-script   = "nsgsh -q .port.1-wire.device.3A6E0E0100000062.circuit.A.drop"
```

Предполагается, что контроллер имеет аппаратный адрес 3A6E0E0100000062. Подробно о работе с устройствами 1–Wire см. Часть 2.

2. Контроль доступности заданного узла по каналу Ethernet MAN. Банкомат штатно соединяется с процессинговым сервером по городской сети Ethernet. При отказе этого канала связи маршрут на процессинг через него удаляется и остаётся только резервный маршрут (с большей метрикой) через сотовое соединение.




```
ip
: route
: : 1
: : : device           = "s1"
: : : metric           = 3
: : : network          = "123.45.67.89/32"
: : 2
: : : gateway          = "98.76.54.29"
: : : network          = "123.45.67.90/32"
: filter
: : OUTPUT
: : : 1
: : : : out-interface  = "s1"
: : : : destination    = "123.45.67.90"
: : : : target         = "DROP"
services
: netping
: : route_swap
: : : adm-state        = "up"
: : : description      = "Re-route ATM packets to 3G back-up"
: : : destination      = "123.45.67.90"
: : : failure-script   = "ip route del 123.45.67.89/32 via 98.76.54.29;"
: : : restore-script   = "ip route add 123.45.67.89/32 via 98.76.54.29;"
```

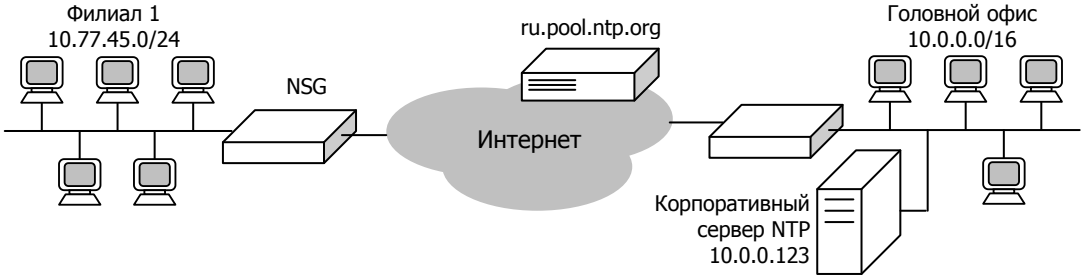
Поскольку посылать *ping* необходимо в канал Ethernet независимо от того, работает он или нет, то в качестве адреса назначения используется другой хост, по возможности максимально близкий к процессинговому серверу. Маршрут на этот хост, с маской /32, всегда указывает в канал Ethernet.

Фильтр запрещает отправку пакетов контрольному хосту через интерфейс s1. В противном случае возможна такая ситуация, что порт Ethernet физически отключён и маршрут через него пропадает, контрольные пакеты уходят в оставшийся маршрут по умолчанию — через сотовый интерфейс — и доходят до цели, в результате событие восстановления *ping* никогда не наступает и маршрут в основной канал никогда не восстанавливается.

Примеры других действий при смене канала связи (добавить в failure-script, restore-script):

— Рестарт IPsec: ipsec setup restart;
— Очистка таблицы NAT: conntarck -F;
— Светодиодная индикация: nsgsh -q .tools.led.green.off .tools.led.red.on;
и наоборот.

Динамическая реконфигурация устройства NSG. Устройство NSG–605 работает в качестве шлюза для подключения к корпоративной сети VPN (IPsec, OpenVPN, NSG *ui*TCP). Регламент работы данной сети требует, чтобы время на устройстве было синхронизировано с единым сетевым сервером NTP. Проблема в том, что для подключения к VPN используются сертификаты X.509 с ограниченным сроком действия. Но встроенных аппаратных часов на этом устройстве нет, поэтому после выключения питания время устанавливается в 1 января 2000 г. и установить безопасное соединение невозможно. Соответственно, нет и доступа к серверу NTP!



Решение в данном случае состоит в том, что время сначала устанавливается по публичному серверу NTP — для проверки сертификатов этого достаточно. После подключения к VPN клиент NTP на устройстве NSG перенастраивается для работы с корпоративным сервером. Для *ping* в данном случае принудительно устанавливается адрес источника, соответствующий защищённому интерфейсу устройства NSG — в этом случае источник и назначение пакета находятся в двух подсетях корпоративной сети, и пакет классифицируется как подлежащий передаче по защищённому туннелю.

```
services
: netping
: : time_reset
: : : adm-state        = "up"
: : : description      = "Switch NTP to corporate server"
: : : destination      = "10.0.0.123"
: : : interval         = 1200
: : : retry            = 3
: : : source           = "10.77.45.1"
: : : start-delay      = 60
: : : failure-script   = "nsgsh -q .system.ntp.host.ru=pool.ntp.org .system.ntp._apply"
: : : restore-script   = "nsgsh -q .system.ntp.host=10.0.0.123 .system.ntp._apply"
system
: ntp
: : enable             = true
: : host               = "ru.pool.ntp.org"
tunnel
: ipsec
: : connections
: : : head_office
: : : : leftsourceip   = "10.77.45.1"
: : : : leftsubnet     = "10.77.45.0/24"
: : : : rightsubnet    = "10.0.0.0/16"
.....................................................
```

ПРИМЕЧАНИЕ Приведённые скрипты требуют запуска nsgsh с правами администратора. Однако поскольку они исполняются, по существу, один раз при старте системы, то можно полагать, что никакой другой сессии в этот момент открыто не будет и скрипт успешно исполнится.

§4.4.5. Клиент VRRP

Клиент VRRP (Virtual Router Redundancy Protocol, IETF RFC–2338) предназначен для построения отказоустойчивых пулов маршрутизаторов. Два или более маршрутизаторов объединяются в виртуальный роутер, имеющий уникальный идентификатор (VRID), один или несколько IP-адресов (которые могут принадлежать одному из физических устройств, или не принадлежать ни одному из них) и виртуальный же MAC-адрес (не совпадающий ни с одним из MAC-адресов физических устройств). Каждому из устройств, входящих в состав виртуального роутера, назначается некоторый приоритет от 1 до 255. Устройство, имеющее наибольший приоритет, объявляется ведущим (Master) и начинает обрабатывать все пакеты с IP- и MAC-адресами виртуального устройства. Таким образом, весь трафик клиентов направляется через него.

Ведущий маршрутизатор регулярно рассылает ведомым *multicast*-сообщения, свидетельствующие об его работоспособности. Если он выходит из строя, то остальные члены группы сравнивают свои приоритеты, и устройство с наибольшим приоритетом снова становится ведущим.

Специальным является случай, когда IP-адрес виртуального роутера принадлежит одному из физических устройств, входящих в него. В этом случае оно всегда становится ведущим и имеет наивысший приоритет — 255.

Говоря более строго, членами виртуального роутера являются не физические устройства в целом, а их IP-интерфейсы. Интерфейсы должны быть широковещательного типа (физические порты Ethernet, VLAN). Один и тот же интерфейс может одновременно входить в состав нескольких виртуальных роутеров с различными VRID и адресами. Настройка VRRP выполняется в меню портов соответствующих типов в узле vrrp . Основными параметрами, по существу механизма VRRP являются:

— Список виртуальных роутеров, в которые входит данный интерфейс. (Интерфейс может входить более чем в один роутер.) Элементами списка являются VRID, которые могут быть целочисленными значениями от 1 до 255. Однако в данном случае они рассматриваются не как целые числа, а как текстовые имена (состоящие из одних только цифр). В частности, данный список не переупорядочивается автоматически и может быть не сплошным.

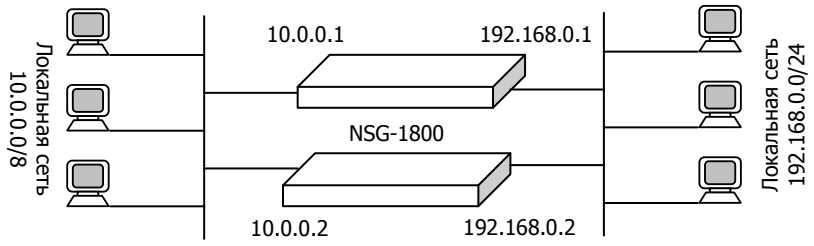
— IP-адрес(-а) виртуального роутера. Если данный интерфейс физического роутера становится ведущим, то эти адреса добавляются к списку IP-адресов, присвоенных ему.

— Приоритет данного физического устройства перед остальными членами группы. Бóльшие значения соответствуют более высокому приоритету. Приоритет 255 следует назначать в том и только в том случае, если IP-адрес виртуального роутера принадлежит данному физическому интерфейсу; в остальных случаях приоритет может принимать значения от 1 до 254.

— Административное состояние данного интерфейса в составе виртуального роутера. Особый случай имеет место, если выключается интерфейс, который в данный момент является ведущим маршрутизатором в группе. В этом случае он рассылает остальным членам группы специальное оповещение со значением приоритета 0, чтобы побудить их приступить к немедленным выборам нового ведущего.

Дополнительно можно изменить интервал рассылки оповещений. По умолчанию, он установлен в минимальное значение — 1 сек. — чтобы обеспечить наиболее оперативную перестройку виртуального роутера. Увеличение этого параметра позволит немного сэкономить служебный трафик (что едва ли актуально в современных сетях Ethernet), но приведёт к замедлению реакции.

Пример. Два устройства образуют виртуальный шлюз между двумя локальными сетями, резервируя друг друга. IP-адреса, статически назначенные их интерфейсам, показаны на рисунке. Для балансировки нагрузки при штатной работе обоих устройств на клиентских хостах указаны в качестве шлюзов по умолчанию примерно поровну 192.168.0.1 и 192.168.0.2, 10.0.0.1 и 10.0.0.2.



Конфигурация устройства А:

```
port
: eth0
: : ifAddress
: : : prefix        = "10.0.0.1/8"
: : vrrp
: : : 1
: : : : ifAddress
: : : : : 1         = "10.0.0.1"
: : : : prio        = 255
: : : 2
: : : : ifAddress
: : : : : 1         = "10.0.0.2"
: eth1
: : ifAddress
: : : prefix        = "192.168.0.1/24"
: : vrrp
: : : 1
: : : : ifAddress
: : : : : 1         = "192.168.0.1"
: : : : prio        = 255
: : : 3
: : : : ifAddress
: : : : : 1         = "192.168.0.2"
```

Конфигурация устройства Б:

```
port
: eth0
: : ifAddress
: : : prefix        = "10.0.0.2/8"
: : vrrp
: : : 1
: : : : ifAddress
: : : : : 1         = "10.0.0.1"
: : : 2
: : : : ifAddress
: : : : : 1         = "10.0.0.2"
: : : : prio        = 255
: eth1
: : ifAddress
: : : prefix        = "192.168.0.2/24"
: : vrrp
: : : 1
: : : : ifAddress
: : : : : 1         = "192.168.0.1"
: : : 3
: : : : ifAddress
: : : : : 1         = "192.168.0.2"
: : : : prio        = 255
```

В данном случае в одной локальной сети определены виртуальные роутеры с идентификаторами 1 и 2, в другой — 1 и 3. Виртуальные роутеры с идентификатором 1 существуют в обеих сетях, что не противоречит протоколу: это две независимые сущности, никак не пересекающиеся друг с другом.

ПРИМЕЧАНИЕ Протокол VRRP (а также аналогичный ему HSRP) следует использовать с осторожностью, поскольку он, по существу, переносит точку потенциального отказа с единственного маршрутизатора на коммутатор локальной сети, в который включены маршрутизаторы. Получается по сути то же самое, но за двойную и более цену. Последовательная стратегия дублирования устройств, портов и каналов связи приводит к необходимости построения двух самодостаточных сетей связи от одной точки входа трафика (прикладного хоста, маршрутизатора локальной сети и т.п.) до другой, параллельно друг другу. Желательно также иметь перемычки между этими сетями в промежуточных узлах и использовать протоколы динамической маршрутизации, чтобы сохранить работоспособность системы при отказах на разных сегментах разных сетей одновременно.

§4.4.6. Агент SNMP

В программное обеспечение устройств NSG Linux 2.0 входит встроенный агент SNMP. Для мониторинга и управления данными устройствами могут использоваться любые стандартные платформы сетевого управления на основе SNMP. Собственно агент SNMP обеспечивает доступ к устройству NSG как к объекту управления, а также определяет субъектов этого управления. Настройка SNMP-агента осуществляется в узле .services.snmp .

Агент SNMP поддерживает версии SNMP *v1*, *2c* и *3*. Версии существенно различаются механизмами доступа к областям дерева объектов SNMP и средствами безопасности.

— SNMP v1 обеспечивает простейший доступ к дереву объектов, начиная от корня или от одного заданного OID. Субъектами управления являются управляющие сообщества (*SNMP communities*), члены которых идентифицируются по IP-адресам.

— SNMP v2c также использует управление на основе *communities*, но добавляет модель доступа на основе *видов* (View-based Access Control Model, VACM), позволяющую комбинировать несколько ветвей дерева объектов, а также дополнительно вырезать более узкие ветви внутри них.

— SNMP v3 предусматривает управление на основе модели безопасной работы пользователей (User Security Model, USM). Она предусматривает аутентификацию каждого пользователя по имени и ключу, а также защиту передаваемых данных с помощью другого ключа. В зависимости от результатов аутентификации и наличия защиты данных, для пользователя выбираются виды, доступные с правами на чтение, на чтение и запись, или на уведомление.

Настройка SNMP-агента включает в себя несколько задач.

1. Установка идентификаторов данного узла: местонахождение, имя и контакты ответственного специалиста (переменные sysLocation и sysContact согласно RFC–1213), а также engineID для SNMP v3. Engine ID является одной из компонент для генерации ключей; из него, в совокупности с ключами, заданными в узле usm-model.user, создаются реальные ключи для аутентификации и защиты данных при использовании USM. Параметр опциональный; если он не установлен, то используется некоторое значение по умолчанию, которое определяется первым IP-адресом данного хоста.

2. Создание управляющих сообществ SNMP *v1* и *2*. Сообщество состоит из одной или более управляющих станций (*managers*), объединенных некоторым идентификатором — именем *community*. Менеджеров можно указывать как поодиночке (с маской /32), так и целыми сетями. Для каждого сообщества определяются права доступа и доступная ветвь дерева объектов SNMP *v1*. Если начальный OID не указан, то по умолчанию для членов данного сообщества доступно всё дерево, начиная с iso(1).

Настройка SNMP *v1*, *v2c* может быть также выполнена без явного указания *communities* средствами модели USM (см. ниже). В частности, более развитая система видов для SNMP *v2c* настраивается, в данной версии NSG Linux, в узле usm-model.view .

3. Настройка модели USM. Протокол SNMP *v3* предусматривает три основные категории объектов конфигурации:

— Виды (SNMP *views*) описывают некоторое подмножество SNMP-дерева, которое может быть сделано доступным для определённых пользователей.

— Каждый из пользователей (*users*) включается в определённую группу, и для него устанавливаются:
  — Модель безопасности (*security model*) — v1, v2c или usm. Первые два варианта интегрируют SNMP *v1* и *v2* в структуру SNMP *v3*, третья использует возможности SNMP *v3* по существу.
  — Для пользователей, которым установлена модель usm — алгоритмы для аутентификации и защиты данных. Защита может использоваться только при включённой аутентификации.
  — Ключи для аутентификации и защиты данных, если эти функции используются. Строго говоря, это не

полные ключи, а заготовки для них (*passphrases*). Реально используемые ключи генерируются автоматически из этих заготовок и *Engine ID* данного устройства.

— Группы пользователей (*groups*) связывают воедино пользователей и виды. Группа определяет, какие виды и в каком режиме (только чтение, чтение/запись, уведомление) будут доступны пользователям при использовании тех или иных моделей и уровней безопасности. Для каждого уровня безопасности, предусмотренного в системе, должен быть указан хотя бы один из трёх возможных видов.

Для настройки этих объектов используются, соответственно, узлы usm-model.view, usm-model.user и usm-model.group .

ВНИМАНИЕ Если какой-либо из видов, указанных в узле usm-model.group.*имя* , не определен, то соответствующая операция для данной группы недоступна.

Если *community* не определено, права пользователей настраиваются в узле usm-model, но при этом установлена модель *v1* или *v2c*, то работа SNMP-агента имеет следующие особенности:

— Менеджерам разрешён доступ с любых IP-адресов.
— Доступные области определяются командами group и view так же, как и при использовании SNMP *v3*.
— При обращении к устройству необходимо в SNMP-менеджере в качестве имени *community* указывать имя пользователя, например:

```
snmpwalk  -v 2c -c MYUSERNAME ...
```

§4.4.7. Агент Zabbix

Zabbix — комплексная многофункциональная система сетевого мониторинга и управления корпоративного уровня с открытым кодом, разработанная Алексеем Владышевым и распространяемая компанией Zabbix SIA.

Программное обеспечение Zabbix позволяет контролировать разнообразные параметры, характеризующие как работу сетей, так и состояние серверов. Zabbix имеет гибкий механизм уведомлений по электронной почте, развитые средства генерации отчётов и визуального представления данных.

Система Zabbix состоит из ряда функциональных компонент, включая сервер, прокси, агент(ы), хранилище данных и Web-интерфейс пользователя. Подробное описание системы и руководство пользователя представлены на web-сайте компании-разработчика

https://www.zabbix.com/documentation/

в связи с чем их изложение в данном документе представляется излишним. Данная версия NSG Linux включает в себя агента Zabbix (*zabbix_agentd*), работающий в режиме демона и осуществляющий сбор требуемой информации и её отправку на центральный сервер мониторинга. Параметры настройки в узле services.zabbix.agentd, в основном, идентичны параметрам конфигурационного файла собственно Zabbix-агента. Исключением является узел userparams, который представляет собой список ключей UserParameter=<key>,<shell script>; в NSG Linux он записывается в формате

```
userparams
: key = "shell script"
.....
```

Сверх оригинальной документации Zabbix, уместно привести несколько общих пояснений:

1. Система может работать как в активном, так и в пассивном режиме. Режимы именуются по отношению к агенту Zabbix, который работает на наблюдаемом устройстве, т.е.:
   — в пассивном режиме агент не инициирует обращений к серверу и только отвечает на его запросы (*polling*)
   — в активном режиме агент регулярно обновляет заданный список наблюдаемых данных и отправляет эти данные на сервер по достижении либо максимального интервала времени, либо максимального объёма буфера — в зависимости от того, что произойдёт раньше (*trapping*)
2. Для получения требуемых данных, в общем случае, используются те или иные скрипты (предопределённые в агенте или заданные пользователем). Кроме того, агент Zabbix позволяет выполнять команды, поступившие от удалённого сервера. Все скрипты и команды исполняются в командной оболочке собственно ОС Linux (ash) и не имеют отношения к работе оболочки nsgsh и Web-интерфейса NSG.
3. Для исполнения отдельных скриптов и команд могут потребоваться права пользователя root. В этом случае агент запускается от имени этого пользователя, что указывается специальным параметром в конфигурации. Если параметр установлен в значение false, то агент запускается с правами юзера обыкновенного.
4. Для приёма запросов от сервера в пассивном режиме агент открывает TCP-сокеты по заданному номеру порта. Для передачи на сервер в активном режиме используется TCP-соединение на заданный номер порта на сервере. По умолчанию, используются порты 10050 и 10052.

§4.4.8. Служба UPnP

Служба Universal Plug and Play предназначена для автоматической настройки правил Destination NAT (виртуальных серверов) на устройствах NSG. В этом случае хосты-клиенты UPnP, находящиеся во внутренней сети, посылают устройству NSG запросы на добавление и удаление соответствующих правил NAT для протоколов TCP и UDP в зависимости от приложений, работающих на них и подразумевающих доступ к ним из внешнего мира. Демон UPnP, работающий на устройстве NSG, обрабатывает эти запросы и генерирует/удаляет соответствующие правила в цепочках ip.nat.UPnP и ip.filter.UPnP . Обращения к этим цепочкам создаются, соответственно, в цепочках ip.nat.PREROUTING и ip.filter.FORWARD и имеют наивысший приоритет; таким образом, входящие транзитные пакеты сначала передаются на обработку в цепочки UPnP, а затем возвращаются в исходную цепочку.

Служба UPnP в NSG Linux 2.0 использует проект miniupnpd и поддерживает как спецификацию Internet Gateway Device (IGD), являющуюся составной частью общей спецификации UPnP, так и альтернативный протокол NAT Port Mapping Protocol (NAT–PMP). Настройка службы производится в узле .services.upnp .

Для работы службы необходимо указать имя и IP-адрес интерфейса, подключённого к внешней сети. IP-адрес указывается только для статически сконфигурированных интерфейсов. По существу, он необходим в случае, когда этот интерфейс имеет несколько адресов; если он не указан явно, то для настройки DNAT используется первый адрес из списка. Для интерфейсов, настраиваемых динамически, адрес указывать не следует — автоматически используется адрес, назначенный ему по PPP или DHCP, и предполагается, что этот адрес единственный.

Остальные параметры для настройки DNAT содержатся в конкретном запросе от конкретного клиента: протокол (TCP либо UDP), порт назначения на внешнем интерфейсе, новые IP-адрес и порт назначения во внутренней сети. Обработка этих запросов регулируется следующими параметрами:

— Списком внутренних сетей, из которых служба UPnP готова принимать запросы. В простейшем случае это целиком одна или несколько подсетей IP, непосредственно подключённых к внутренним интерфейсам маршрутизатора.

— Фильтром запросов — узел permissions. В этом узле можно создать несколько правил, разрешающих или запрещающих исполнение запросов, параметры которых попадают в заданную совокупность значений: диапазон портов назначения на внешнем интерфейсе, внутреннюю подсеть IP и диапазон новых портов назначения в этой сети. Каждое правило действует одновременно для обоих протоколов, обрабатываемых данной службой — TCP и UDP.

  Правила анализируются последовательно, в порядке их нумерации. Если для принятого запроса найдено подходящее правило, то выполняется действие, предписанное этим правилом: выполнить запрос или игнорировать его. Последующие правила не рассматриваются. Если для запроса не найдено ни одного явно заданного правила, то выполняется действие, заданное по умолчанию.

— Безопасным режимом. В обычном режиме работы UPnP не анализирует связи между IP-адресом, с которого поступил запрос, и внутренним IP-адресом, указанным в теле запроса. Таким образом, некорректно работающий или контролируемый злоумышленником хост может создавать и удалять правила DNAT, указывающие на некоторый другой хост во внутренней сети. В безопасном режиме накладывается дополнительное ограничение: каждый клиент UPnP может создавать только правила, относящиеся к нему самому. Это обеспечивает некоторую степень защиты от подставных хостов и других злоупотреблений.

В целом, относительно безопасности UPnP необходимо заметить, что эта технология вообще не предусматривает никаких средств аутентификации и авторизации и исходит из предположения, что все хосты во внутренних подсетях априори заслуживают доверия.

Параметр natpmp определяет протокол, по которому устройство NSG принимает запросы от клиентов. Он включает поддержку протокола NAT–PMP, в дополнение к UPnP IGD.

ПРИМЕЧАНИЕ Для работы IGD на устройстве должны быть открыты для входящих и исходящих пакетов (цепочки ip.filter.INPUT, ip.filter.OUTPUT) порты UDP 1900 и TCP 2869 на IP-интерфейсах локальной сети. Для работы NAT–PMP должен быть открыт порт UDP 5351.

Оставшиеся два параметра определяют взаимодействие службы UPnP с клиентами. Для начала, маршрутизатор, поддерживающий UPnP, оповещает клиентов во внутренней сети о своём присутствии, возможностях и предлагаемых услугах с помощью широковещательных уведомлений. Интервал рассылки этих уведомлений определяется параметром notify-interval .

Далее, когда клиент UPnP обнаруживает в сети устройство с функциями сервера, он, как правило, выводит в своём интерфейсе пользователя ссылку на некоторую информационную Web-страницу (создаёт иконку на рабочем столе, и т.п.). Предполагается, что на этой странице пользователь может ознакомиться с возможностями данного устройства, предоставляемыми им сервисами и т.п. Это может быть также любая другая страница — вход на корпоративный Web-портал, в информационную службу, реклама и т.п. Непосредственно на работу службы UPnP данный параметр не влияет. По умолчанию, используется страница входа в Web-интерфейс устройства NSG.

§4.5. Инструменты сетевого управления

§4.5.1. Общие сведения об инструментах сетевого управления

Инструменты и утилиты, предназначенные для ручной отладки и мониторинга процессов, происходящих в сети, содержатся в узле .tools корневого меню. Элементы этого узла являются интерфейсом для запуска соответствующих утилит и клиентских программ ОС Linux и содержат набор наиболее употребительных параметров для их вызова, которые передаются в командную строку. Структура данных узлов, как и остальных элементов меню, унифицирована для Web-интерфейса и консольной оболочки nsgsh.

Непосредственно запуск программы осуществляется командой launch. При работе в Web-интерфейсе в этом случае открывается терминальное окно для ввода-вывода программы. (Поскольку Web-эмулятор терминала работает довольно медленно, рекомендуется использовать его только для коротких и небольших операций; для длительной работы, такой как настройка удалённого устройства по сети, рекомендуется использовать консольный режим доступа на устройство NSG).

ПРИМЕЧАНИЕ Все параметры, установленные пользователем для исполнения данных команд, сохраняются до завершения текущего сеанса работы пользователя.

Набор параметров для каждой программы может быть дополнен в последующих версиях NSG Linux 2.0 по мере необходимости.

Полный доступ ко всем инструментам сетевого управления в обычном режиме командной строки возможен в консольном режиме из командной оболочки ОС Linux, доступной только пользователю root. Для просмотра всех поддерживаемых параметров следует ввести требуемую команду с ключом --help или ознакомиться с *man pages* по данной программе.

Подробно об основах работы в ОС Linux для администраторов устройств NSG см. Часть 7.

§4.5.2. Клиенты Telnet и SSH

Клиенты Telnet и SSH предназначены для доступа с устройства NSG на другие устройства сети в простейшем (небезопасном) и в безопасном режиме, соответственно. При работе в сетях общего пользования настоятельно рекомендуется применять исключительно SSH.

Клиент Telnet использует стандартный набор управляющих символов и команд (CTRL-] для выхода в командный режим, и т.п.).

Служба SSH предусматривает передачу данных в бинарном режиме. Помимо ручного доступа на удалённое устройство, она может быть использована и для других целей, в частности, для копирования произвольных бинарных файлов с устройства на устройство. Подробно об этой возможности использования SSH в ОС Linux см. Часть 7.

Помимо аутентификации клиента на удалённом хосте при помощи пароля, механизм SSH предусматривает и другой способ — при помощи публичного ключа клиента, который должен быть заранее сгенерирован (в паре с приватным) на неё самом (либо на стороннем удостоверяющём центре) и помещён в список доверенных ключей на сервере. В NSG Linux 2.0 эти действия могут быть выполнены с помощью разовых команд keygen и append в меню клиента SSH. Первая генерирует пару ключей; поскольку это довольно ресурсоёмкая операция, для её выполнения требуется подтверждение: либо повторный ввод команды (в консольной оболочке nsgsh), либо ввод yes в поле значения (в Web-интерфейсе). Вторая команда соединяется с заданным хостом, аутентифицируется на нём вручную при помощи пароля и передаёт на него файл с ключом.

ВНИМАНИЕ Для генерации ключей SSH-клиента необходимо войти в систему (*nsgsh* или Web-интерфейс) под именем root . Это общее правило, вытекающее из разграничения прав пользователей. Ключи предназначены для доступа на удалённую машину под именем root и должны храниться в домашней директории пользователя root, к которой имеет доступ только он один.

После того, как ключ передан на удалённый хост, дальнейший доступ на него с локальной машины не требует ручного ввода пароля. Данный способ особенно удобен для автоматизированного выполнения различных операций: копирования конфигураций, мониторинга всей системы *ui*TCP с одного сервера, управления удалённым устройством из скриптов, выполняемых локально (в т.ч., например, трансляции локальных событий в удалённый обработчик) и т.п. Подробнее о различных операциях данного типа см. §4.8.

ПРИМЕЧАНИЕ Для генерации ключей SSH, их передачи на удалённое устройство и синхронизации конфигураций используется утилита nsgconfsync, которую можно использовать непосредственно из командной строки Linux. Подробное описание данной утилиты см. в Части 7.

§4.5.3. Утилиты *ping* и *traceroute*

Утилиты ping и traceroute — стандартные инструменты для проверки связности сети.

При использовании данных утилит следует обращать внимание на IP-адрес источника, с которым будут отправляться тестовые пакеты. По умолчанию, в это поле подставляется IP-адрес интерфейса, через который отправляются пакеты в соответствии с текущей таблицей маршрутизации. В частности, при работе с туннелями IPsec необходимо принудительно указывать в качестве source адрес устройства NSG в приватной (защищаемой) сети; это гарантирует, что пакет, адресованный на другую сторону туннеля, будет действительно отобран для передачи по туннелю и отправлен в этот туннель. В противном случае пакет будет отправлен с публичного интерфейса во внешний мир, минуя туннель.

§4.5.4. Утилита *tcpdump*

tcpdump — мощный инструмент для трассировки и мониторинга трафика, передаваемого через заданный сетевой интерфейс. Использование tcpdump и анализатора пакетов WireShark настоятельно рекомендуется не только для отладки конкретных сетевых решений, но и с целью самообразования.

Основным параметром для вызова tcpdump является имя интерфейса, на котором требуется выполнять мониторинг. Если имя не указано, то, по умолчанию, программа запускается на интерфейсе eth0.

Наиболее важным из дополнительных параметров является так называемое *выражение* (*expression*). Оно по существу представляет собой фильтр и позволяет отбирать в трассу только пакеты, интересующие пользователя. Примеры простейших выражений (примитивов):

host X.X.X.X	tcpdump захватывает только пакеты, содержащие заданный адрес (не важно, в качестве источника или назначения). Вместо адреса может использоваться также имя хоста, если на устройстве включён клиент DNS.
dst host X.X.X.X	Только пакеты с заданным адресом назначения.
src host X.X.X.X	Только пакеты с заданным адресом источника.
net XXXX	Сеть с подразумеваемой по умолчанию маской; пишется только ненулевая часть адреса сети, например, 10, 172.16, или 192.168.1. Аналогично для src net и dst net.
net X.X.X.X mask Y.Y.Y.Y	Сеть с явно указанными адресом и маской. Аналогично для src net … mask и dst net … mask.
ip proto PPP	Протокол 4 уровня. Указывается номером или названием. Названия tcp, udp, icmp необходимо предварять обратным слэшем (\), поскольку они совпадают с соответствующими ключевыми словами.
tcp, udp, icmp	Сокращённые обозначения для ip proto \tcp, ip proto \udp, ip proto \icmp, соответственно.
port N	Номер порта TCP или UDP. Аналогично для src port и dst port.
portrange M-N	Диапазон номеров портов TCP или UDP. Аналогично для src portrange и dst portrange.

Выражение может содержать несколько примитивов, соединённых логическими операторами && (И), || (ИЛИ) и ! (НЕ). В этом случае его необходимо взять в кавычки, чтобы эти операторы не были восприняты как спецсимволы самой командной оболочки, например:

```
"host 1.2.3.4 && tcp port 80"
```

Подробно о выражениях и других опциях вызова tcpdump см. *man pages* по данной программе.

tcpdump может выводить захваченные пакеты в их неизменном бинарном виде, либо выполнять простейший анализ пакетов и показывать информацию о типе пакетов в удобочитаемой форме, например:

```
07:34:24.717514 arp who-has 192.168.1.1 tell 192.168.1.250
07:34:24.739441 arp reply 192.168.1.1 is-at 00:09:56:20:00:2b
```

По умолчанию, в консольном окне показывается трасса в человеко-читаемой форме. Параметр write-to-file позволяет направить вывод программы во временный файл в формате, совместимом с большинством программ для анализа пакетов. Позже этот файл может быть выгружен на ПК любым доступным способом (TFTP, FTP, NFS, USB-носитель) и исследован с помощью соответствующих программ, например, WireShark (http://www.wireshark.org).

§4.5.5. Утилита *mail*

В состав NSG Linux 2.0 входит утилита nsgsmtp, вызываемая из командной строки ОС Linux и предназначенная, в первую очередь, для оправки уведомлений по электронной почте из различных скриптов и обработчиков событий. Узел меню .tools.mail предоставляет дружественный интерфейс к этой утилите, с помощью которого можно вручную отправить с устройства NSG почтовое сообщение.

Для отправки писем необходимо, в первую очередь, указать имя или IP-адрес почтового сервера SMTP. Если сервер требует аутентификации пользователей (это обычная мера защиты против злоупотреблений, таких как рассылка спама), то дополнительно необходимо задать имя и пароль пользователя.

Обязательными компонентами собственно для письма являются адрес получателя и адрес отправителя. По существу обязательны также тема и тело письма, но если они не указаны, письмо отправляется с заголовком "No subject" и/или текстом "No message", соответственно.

Дополнительно может потребоваться явное указание номера порта TCP, используемого почтовым сервером — если он отличается от стандартного 25. При очень строгих настройках почтового сервера дополнительно может потребоваться указание домена отправителя. Это имя указывается в поле helo протокола SMTP и должно соответствовать IP-адресу интерфейса, через который отправляются пакеты на сервер SMTP.

Если почтовый сервер работает только в безопасном режиме (SMTP-over-SSL), то необходимо включить опцию ssl . Номер порта TCP по умолчанию в этом случае — 465.

После того, как письмо составлено, отправка производится разовой командой send . Все параметры отправки почты действуют только в пределах одного сеанса работы пользователя.

Подробно об утилите nsgsmtp см. Часть 7 данного руководства.

§4.6. Другие прикладные сервисы

§4.6.1. Сервер печати

Сервер сетевой печати (принт-сервер) обслуживает принтер, подключённый к устройству NSG через порт USB, и делает его доступным для других компьютеров в сети. Данная служба совместима с принтерами, поддерживающими технологию HP JetDirect (другое название — Raw Socket Printing). Служба может поддерживать до 3 принтеров, доступных по портам TCP 9100 (стандартый), 9101, 9102, соответственно.

ПРИМЕЧАНИЕ Перед приобретением устройства и принтера для совместной эксплуатации следует уточнить, поддерживает ли данная модель принтера технологию JetDirect. Информацию о совместимости можно получить на специализированных Web-ресурсах и, в отдельных случаях, в службах технической поддержки фирм-производителей.
Список проверенных моделей принтеров, а также адаптеров USB–LPT, доступен на Web-сайте NSG (http://www.nsg.ru) в разделах FAQ, Форум и Документация/Справочные материалы. Компания NSG будет признательна пользователям за любую (положительную или отрицательную) информацию о совместимости других моделей принтеров.

Помимо номера порта TCP, принтер может иметь два дополнительных параметра: привязку к определённому IP-адресу (если нет, то он доступен по всем IP-адресам, присвоенным данному устройству NSG), и режим одно/двустороннего обмена. Большинство современных моделей принтеров работает в двустороннем режиме, обеспечивающем расширенный обмен диагностической информацией с соответствующими драйверами и утилитами.

Для работы с принт-сервером на клиентских ПК должен быть установлен драйвер принтера и сделана привязка этого драйвера не к локальному физическому порту, а к виртуальному порту TCP/IP.

ПРИМЕЧАНИЕ Сетевой принтер, работающий на основе Raw Socket Printing, не имеет ничего общего с понятием "общего доступа к принтерам и файлам" в сети Майкрософт. В сети Майкрософт такой принтер настраивается как локальный, подключённый к виртуальному порту TCP/IP.

Конфигурация принт-сервера содержит небольшое число параметров и полностью производится в меню порта, имеющего тип printer.

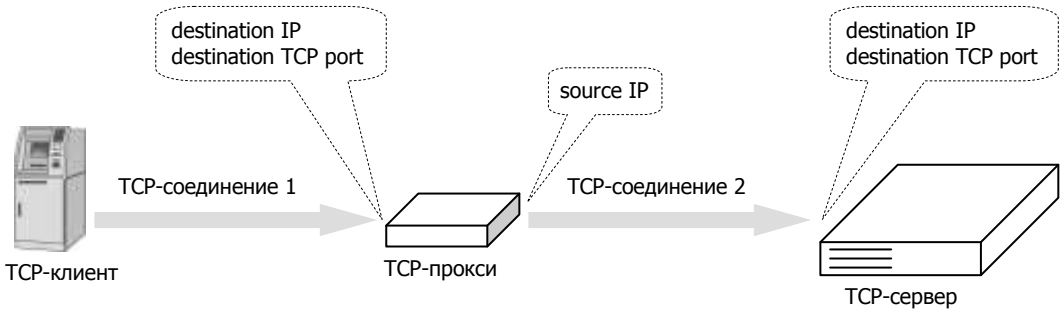
Примеры настройки клиентских ПК в ОС Windows и Linux приведены в документе NSG:

Настройка клиентов сетевой печати TCP/IP

§4.6.2. TCP-прокси

Служба TCP-прокси состоит из одного или нескольких демонов, предназначенных для работы в качестве промежуточного агента между некоторыми клиентом и сервером. Клиент инициирует соединение с сервером по заданному IP-адресу и номеру порта TCP, сервер ожидает входящие соединения на этом порту. TCP-прокси позволяет ретранслировать соединения между клиентом и сервером таким образом, чтобы для сервера они исходили с адреса устройства NSG. Типичным примером является задача, когда процессинговый центр выделяет банку один IP-адрес, с которого должны исходить все соединения. В этом случае многочисленные банкоматы устанавливают соединения со своих IP-адресов не напрямую с процессинговым центром, а с банком; отсюда они все ретранслируются в процессинговый центр уже с единственного адреса банка. (Данная задача может быть решена и с помощью NAT, но в некоторых случаях TCP-прокси удобнее.)

Каждый из демонов принимает входящие соединения со стороны на заданном IP-адресе и порту TCP. Входящие пакеты разбираются вплоть до уровня TCP включительно, пакеты вышестоящего протокола (HTTP, FTP, NDC, DDC и т.п.) остаются в неизменном виде. От прокси устанавливается новое соединение с сервером, полезная нагрузка заново инкапсулируется в TCP, IP и т.д. и отправляется по этому соединению.



Обязательными параметрами каждого демона TCP-прокси являются номер порта TCP для входящих соединений, IP-адрес и номер порта TCP для исходящих соединений. При этом два TCP-порта не обязательно должны совпадать, т.е. прокси дополнительно осуществляет трансляцию портов. Дополнительно можно указать IP-адрес, на котором устройство принимает входящие соединения со стороны клиента (по умолчанию — на всех адресах, назначенных устройству) и IP-адрес, который будет указываться в качестве адреса источника в пакетах, адресованных серверу (по умолчанию в пакет подставляется адрес того IP-интерфейса, через который он отправляется). Оба адреса должны принадлежать каким-либо интерфейсам устройства NSG — если не физическим, то, например, lo или dummy .

Помимо этого, с любой из сторон прокси можно использовать механизм TCP keepalive, который решает две задачи. Во-первых, он контролирует соединение с удалённым хостом; если подряд на несколько пакетов *keepalive*, посланных через заданные промежутки времени, не получено ни одного подтверждения, то соединение считается умершим, и соединение с другой стороны прокси также разрывается. Во-вторых, он позволяет эмулировать активность одной из сторон (клиента или сервера, или взаимно), если другая сторона следит за ней и при отсутствии активности может разорвать соединение. Такая ситуация встречается, в частности, в некоторых типах программного обеспечения для процессинговых центров.

ПРИМЕЧАНИЕ IP-адрес сервера и номера портов назначения на нём могут быть одинаковыми для всех клиентов, в зависимости от особенностей работы ПО сервера. В любом случае, для каждого TCP-соединения между прокси и сервером всегда остаётся, как минимум, один уникальный параметр — номер порта TCP источника (который выбирается сетевым стеком прокси-машины автоматически), и по этому параметру сервер может различать соединения от разных клиентов даже в том случае, если остальные три параметра (порт назначения и оба IP-адреса) совпадают.

§4.6.3. Служба Netflow в пространстве пользователя

Программное обеспечение NSG Linux 2.0 поддерживает сбор статистики трафика в формате Netflow и её отправку на заданные сервера-коллекторы статистики. Данная функция реализована двумя независимыми средствами: службой Netflow, работающей в пространстве пользователя, и механизмом Netflow, встроенным в ядро ОС Linux (см. след. параграф).

Служба Netflow в пространстве пользователя реализована на основе утилиты *fprobe*. Она имеет гибкий набор настроек и схему конфигурации, типичную для многих моделей маршрутизаторов. Недостатками её являются медленная работа и высокая вычислительная нагрузка на процессор. Настройка данной службы производится в узле .services.netflow в виде *шаблонов*. Обязательными параметрами шаблона, по существу задачи, являются список серверов, которым будет отсылаться статистика (должен содержать хотя бы один хост) и административный статус данной службы — up. Далее каждому сетевому интерфейсу можно назначить один из созданных шаблонов Netflow и учитывать статистику с его помощью; назначение шаблонов производится в узле настройки требуемого порта или псевдо-интерфейса.

Именем шаблона, в данной версии NSG Linux, является число от 1 до 255. Следует подчеркнуть, что список шаблонов является именованным и не упорядочивается автоматически.

В качестве необязательных параметров, для каждого шаблона можно задать режим захвата пакетов (избирательный по MAC-адресу интерфейса, или неизбирательный) и выражение-фильтр по протоколам, адресам, номерам портов и другим параметрам. Эти параметры влияют на отбор пакетов для статистики и позволяют сузить круг обрабатываемых пакетов до того, что представляет интерес для пользователя по существу. Формат выражения является стандартом де-факто, используется в большинстве инструментов для захвата и анализа трафика, таких как *tcpdump, pcap, Wireshark*, и подробно описан в руководствах по этим программам.

С точки зрения работы с серверами Netflow, дополнительно можно выбрать версию протокола (поддерживаются версии 1, 5 и 7), а также указать явным образом IP-адрес, который будет подставляться в исходящие пакеты Netflow в качестве адреса источника.

§4.6.4. Служба Netflow в ядре Linux

Альтернативная реализация Netflow встроена в ядро ОС Linux, а именно, в модуль *netfilter*, осуществляющий отбор пакетов по заданному набору критериев и выполнение заданных действий с этими пакетами. Управление этим модулем производится с помощью утилиты *iptables*; в командном дереве NSG Linux 2.0 оно находится в узле .ip.filter и никак не связано с реализацией Netflow в пространстве пользователя (см. пред. параграф). Данная реализация поддерживает NetFlow v5 и v9.

Любому из фильтров, созданных пользователем в данном узле, может быть назначено действие (target) NETFLOW; в этом случае пакеты, отобранные данным фильтром, учитываются в статистике Netflow. Данный механизм отличается высокой скоростью работы и невысокой вычислительной нагрузкой на процессор, но имеет свой набор параметров и настроек.

Общие параметры работы Netfilter в ядре настраиваются в узле .ip.filter.netflow и относятся к отправке статистики на сервер(ы)-коллектор(ы). Обязательным параметром по существу задачи является список серверов, которым будет отсылаться статистика (должен содержать хотя бы один хост). Опциональные параметры устанавливают время отсылки статистики на сервер и условия агрегации потоков.

Статистика по каждому заданному потоку трафика накапливается в течение максимального времени активности, при условии, что поток остаётся непрерывным (см. ниже). По достижении этого времени (active-timeout) статистика отсылается на сервер независимо от других условий.

Если интервал между двумя последовательными пакетами превышает некоторый установленный предел (inactive-timeout), то поток считается завершённым (временно или навсегда). В этом случае статистика по нему, накопленная с момента последней отправки, отправляется на сервер как есть на данный момент.

Агрегация потоков позволяет получить более наглядную картину трафика, игнорируя малосущественные детали. Агрегация может выполняться по IP-подсетям и/или по портам TCP и UDP, например:

192.168.0.0/8=16	Учитывать все сети, захваченные фильтрами вида 192.168.x.0/8, как одну сеть 192.168.0.0/16.
80-89=80	Учитывать все номера портов в диапазоне от 80 до 89 как одно целое с портом 80. Результирующий порт может принадлежать или не принадлежать этому диапазону, например, запись 3128=80 означает, что следует учитывать трафик *http-proxy* вместе с трафиком *http*.

Несколько правил могут быть перечислены через запятую, например:

```
192.0.0.0/8=16,10.0.0.0/8=16,80-89=80,3128=80
```

Правила каждого типа применяются к каждому пакету в последовательности их перечисления до тех пор, пока не найдётся первое, которому данный пакет соответствует. Правила агрегации для сетей и для портов анализируются отдельно.

Подробно о формате и назначении данных параметров см. документацию по пакетам *ipt-netflow* (устаревшая реализация в виде отдельного модуля), *netfilter* и *iptables* (в современной редакции).

§4.6.5. Операции с файлами на устройстве

NSG Linux 2.0 содержит несколько механизмов для приёма, передачи и манипулирования файлами на устройстве, в особенности, для операций с ключами и сертификатами, получения журналов, трасс и т.п.

Сервер SFTP (Secure, или SSH File Transfer Protocol) является составной частью сервера SSH, запускается и конфигурируется одновременно с ним. Если на устройстве включёна служба SSH (см. §4.4.2), то на него также возможен доступ по SFTP. Для работы с устройством можно использовать любой стандартный клиент SFTP, в частности, файловые менеджеры для ПК с возможностью монтирования удалённой файловой системы по SFTP: WinSCP, FAR (ОС Windows), Krusader (ОС Linux), FireFTP (надстройка для броузера Firefox) и т.п.

ПРИМЕЧАНИЕ Не следует путать SFTP с передачей FTP поверх туннеля SSL (т.н. FTPS). SFTP представляет собой самостоятельный протокол, описываемый отдельным документом IETF.

Протокол FISH (FIles transfer over SH) представляет собой простейшую надстройку для манипулирования файлами посредством SSH, реализованную на стороне клиента. На стороне сервера при этом не требуется ничего, кроме поддержки стандартных возможностей SSH. Из числа распространённых файловых менеджеров, поддержка FISH имеется в Midnight Commander и Krusader (ОС Linux).

Возможна также непосредственная передача файлов с/на устройство при помощи SSH (см. Часть 7).

Поскольку все вышеперечисленные механизмы являются расширениями SSH, они обеспечивают безопасную передачу файлов по сетям общего пользования без использования дополнительных средств защиты.

§4.6.6. Сервер TFTP

Сервер TFTP предназначен для приёма и передачи файлов, в первую очередь, для модернизации программного обеспечения сторонних устройств при помощи устройства NSG. Сервер работает по стандартному порту UDP 69. Для хранения файлов может быть выбрана либо встроенная флэш-память устройства (с учётом её доступного размера), либо внешний USB-накопитель. Подробно о подключении USB-накопителей и о манипуляциях с файлами на них см. Часть 2.

В данной версии NSG Linux 2.0 функциональность TFTP-сервера не поддерживается.

§4.7. Обработчик событий

§4.7.1. Общее описание

Обработчик событий представляет собой встроенный механизм, позволяющий контролировать наступление определённых *событий* в системе и выполнять по этим событиям заданные *действия*. События представляют собой переходы некоторых *виртуальных датчиков* из одного *состояния* в другое. Виртуальным датчиком, или источником событий, могут быть разнообразные объекты в системе NSG Linux: сетевые интерфейсы, физические датчики ("сухие контакты", датчики напряжения, температуры, влажности и т.п.), тестовые скрипты типа *netping* и другие.

Состояния датчиков, например, up/down, on/off, cold/normal/hot могут быть определены в системе заранее либо заданы пользователем, в зависимости от типа датчика. Некоторые объекты (в частности, сетевые интерфейсы) всегда являются виртуальными датчиками с заранее заданным набором состояний. Другие имеют в своём меню узел event-generator, в котором описывается набор состояний, интересующих пользователя. Для ряда объектов, в которых этот узел отсутствует в данной версии NSG Linux, можно в качестве временной меры написать скрипт, опрашивающий их состояние (например, *netping* со специфической процедурой test-script вместо *ping*).

В качестве действия обработчик событий может, например, включить/выключить программируемый светодиодный индикатор (при наличии таковых на устройстве), замкнуть/разомкнуть заданную выходную цепь, послать уведомление по SMS или электронной почте, модифицировать таблицу маршрутов и т.п.

§4.7.2. Генераторы событий

Генераторы событий, определённые в системе, имеют иерархическую структуру имён, позволяющую упорядоченно описать большое число разнообразных датчиков. В данной версии NSG Linux предусмотрены следующие типы источников событий:

ifstate.*имя* Сетевой интерфейс с указанным именем. Возможные состояния для данных источников: up, down и no (интерфейс не существует в системе, например, интерфейс PPP или туннель в момент рестарта). В качестве имени может указываться любой из сетевых интерфейсов устройства, например, gre1 или br1.103.11 . Для интерфейсов, имеющих синонимы (*aliases*), например, edge и ppp1, каждый раз генерируются состояния с каждым из имён, так что в обработчике событий можно использовать любое из них.

ПРИМЕЧАНИЕ Интерфейсы Ethernet в Linux всегда считаются находящимися в состоянии UP, независимо от состояния их физического и канального уровней. Таким образом, для них генератор ifstate не имеет смысла. Но, с другой стороны, физический уровень для них также редко бывает информативным, поскольку проблемы возникают с бóльшей вероятностью в последующих сегментах сети Ethernet. По этим причинам для портов Ethernet следует вместо ifstate пользоваться процедурой *netping* и встроенным в неё генератором событий.

устройство.цепь

Входы физических датчиков, подключённых к порту 1–Wire (встроенному, сменному, либо внешнему адаптеру). Датчики, подключённые к шине 1–Wire, не имеют встроенного интеллекта и работают по принципу опроса со стороны контроллера шины (устройства NSG). Поэтому общим параметром для всех датчиков данного типа является интервал опроса.
Имя виртуального датчика составляется из идентификатора самого датчика и имени входа на нём. То и другое могут быть заданы в полях description таким образом, чтобы описать реальный смысл измеряемых величин:

— Если для входной цепи датчика задано непустое имя, например, "Температура в подвале" или "Питание сервера", то оно и служит именем виртуального датчика.

— Если для конкретной входной цепи имя не задано, но задано имя для всего датчика, то виртуальные датчики получают имена вида *имяДатчика*.A, *имяДатчика*.B, ... или *имяДатчика*.1, *имяДатчика*2, … в зависимости от того, как технически именуются входы для данного типа датчиков.

— Если не задано ни имя датчика, ни имя входа, то имена датчиков составляются из идентификатора устройства, под которым оно фигурирует в списке devices, и технического имени входа, например, swt2-3A6E0E0100000062.B.

На одном и том же датчике часть входов может иметь осмысленные названия, а часть — обходиться техническими.

порт.параметр

Генератор событий, встроенный в некоторые типы портов. Позволяет контролировать отдельные параметры физического уровня, такие как уровень сигнала в беспроводном интерфейсе. Для порта и для датчика могут быть заданы содержательные имена (префикс и суффикс, соответственно), например, 3G-port.signal-level . Если они не заданы, то используются техническое имя порта и название параметра, соответственно — например, m1.CSQ .

netping.*имя*

Факт срабатывания заданного *netping*. Если *ping* (или заданный test-script) перестаёт выполняться успешно, то в обработчик событий посылается состояние failure (по умолчанию), если начинает выполняться вновь — состояние restore . Названия обоих состояний могут быть изменены на более удобные. Для самого датчика также может быть назначено содержательное имя; если оно не задано, то имя составляется из префикса netping. и имени конкретной пинговалки.

uitcp.*клиент*.status

Состояние заданного туннеля *ui*TCP. Возможные состояния датчика: normal, suspended, dead.

uitcp.*клиент*.link

Имя канала связи, по которому в данный момент работает заданный туннель *ui*TCP. Возможные состояния — имена каналов, определённые на данном клиенте.

uitcp.*клиент*.backup

Приоритетность текущего канала связи для заданного туннеля *ui*TCP. Возможные состояния: 0 — туннель работает по основному каналу, или приоритетность каналов не определена; 1 — туннель работает по резервному каналу.

mercury230.*параметр*

Параметры, измеряемые трёхфазным электрическим счётчиком Меркурий 230:

COS-*n*	Косинус угла между током и напряжением для *n*-ой фазы
COS-t	Суммарный косинус угла между током и напряжением
E	Суммарная потреблённая энергия, нарастающим итогом
F	Частота в электрической сети
FU-*mn*	Угол между *m*-ой и *n*-ой фазами, при нормальной работе должен составлять по 120°
I-*n*	Ток *n*-ой фазы
P-*n*	Активная мощность *n*-ой фазы
P-t	Суммарная активная мощность
Q-*n*	Реактивная мощность *n*-ой фазы
Q-t	Суммарная реактивная мощность
S-*n*	Полная мощность *n*-ой фазы
S-t	Суммарная полная мощность
U-*n*	Напряжение *n*-ой фазы

По существу, электросчётчик представляет собой совокупность виртуальных датчиков — на каждом из параметров. Полное имя каждого виртуального датчика образуется из префикса "mercury230." (при необходимости может быть изменён) и специфичного имени (sensor-name), удобного для восприятия. Именно это имя фигурирует в настройке обработчика событий, а также в выводимых сообщениях и Web-мониторинге.

ms6.*датчик* Физические датчики, входящие в состав интегрированного датчика NSG MS–6. Имена датчиков:

HUMI	Датчик относительной влажности воздуха, в процентах
TEMP	Датчик температуры, в градусах Цельсия
LUMI	Датчик освещённости
PYRO	Датчик освещённости
MAGN	Датчик магнитного поля
ACCL	Датчик ускорения по 3 осям, в условных единицах (*g*/64)

Имена состояний для данных датчиков — произвольные, все состояния настраиваются пользователем в меню порта, к которому подключён датчик MS–6. Мультидатчик сообщает, строго говоря, не о состояниях, а о событиях — наступлении состояния, отличного от предыдущего. Например, пока температура постоянна, он молчит и генерирует событие только в том случае, если она изменилась на 0,1°C или более.

По существу, данное устройство представляет собой совокупность виртуальных датчиков — на каждой из измеряемых физических величин. Полное имя каждого виртуального датчика образуется из префикса "ms6." (при необходимости может быть изменён) и специфичного имени (sensor-name), удобного для восприятия.

Другие типы виртуальных датчиков могут быть реализованы в последующих версиях NSG Linux.

§4.7.3. Настройка пользовательских состояний датчиков

Для датчиков с дискретными входами, имеющих конечное число состояний, каждое состояние входа может иметь собственное имя, например:

```
device
: swt2-3A6E0E0100000062
: : event-generator
: : : on-event      = "включен"
: : : off-event     = "выключен"
```

Если имя состояния не задано, то оно будет on/off или open/short, соответственно, в зависимости от типа датчика. Для датчиков с несколькими входами названия событий, в данной версии NSG Linux, одинаковы на всех входах.

Для датчиков USB и 1–Wire с аналоговыми входами, измеряющими температуру, напряжение и т.п., необходимо создать одно или несколько состояний в узле event-generator соответствующего порта, устройства 1–Wire, или каждого из входов комплексного датчика. Список состояний по каждому параметру — нумерованный и может содержать произвольное число записей. Каждая запись имеет до 3 параметров: название состояния, минимальное и максимальное значения (включительно). Любое из значений может быть опущено, в этом случае состояние определяется только по другому параметру: больше from или меньше to, соответственно.

Состояния анализируются последовательно в порядке нумерации, пока не найдётся первое, подходящее к текущему значению датчика. Таким образом, последовательно расписывать все возможные диапазоны и следить, чтобы они не пересекались, нет необходимости (хотя для человека это делает конфигурацию более понятной). Например, для датчика температуры, работающего с точностью 0,1°C, следующие две конфигурации равносильны:

```
port
: usb1
: : type              = "multisensor"
: : event-generator
: : : enable          = true
: : : TEMP-states
: : : : 1
: : : : : from        = "50"
: : : : : event-name  = "hot"
: : : : 2
: : : : : from        = "30"
: : : : : event-name  = "warm"
: : : : 3
: : : : : from        = "5"
: : : : : event-name  = "normal"
: : : : 4
: : : : : from        = "-5"
: : : : : event-name  = "shilly"
: : : : 5
: : : : : to          = "-5.1"
: : : : : event-name  = "frozen"
```

```
port
: usb1
: : type              = "multisensor"
: : event-generator
: : : enable          = true
: : : TEMP-states
: : : : 1
: : : : : from        = "50"
: : : : : event-name  = "hot"
: : : : 2
: : : : : from        = "30"
: : : : : to          = "49.9"
: : : : : event-name  = "warm"
: : : : 3
: : : : : from        = "5"
: : : : : to          = "29.9"
: : : : : event-name  = "normal"
: : : : 4
: : : : : from        = "-5"
: : : : : to          = "4.9"
: : : : : event-name  = "shilly"
: : : : 5
: : : : : to          = "-5.1"
: : : : : event-name  = "frozen"
```

Как только найдено состояние, соответствующее текущему значению переменной, дальнейший анализ прекращается и датчик передаёт обработчику событий (event-handler) это состояние. Если просмотрен весь список и ни одного подходящего состояния не найдено (например, между описанными состояниями имеются разрывы, и результат измерения попал в один из них), то передаётся состояние UNKNOWN.

По умолчанию, для каждого из состояний в качестве имени используется конструкция $(VAL), означающая, что в обработчик событий будет посылаться значение датчика "как есть". При необходимости можно его заменить полностью (например, hot и cold), или модифицировать для удобства восприятия (в т.ч. допускаются спецсимволы HTML, например, T=$(VAL)°C). При использовании такого формата обработчик событий имеет на входе не бинарный факт срабатывания датчика, а точное показание, и может использовать более развитый алгоритм для его обработки (см. след. параграф).

ПРИМЕЧАНИЕ Не следует использовать для пользовательских состояний имя nil. Это специальное значение, зарезервированное для особых случаев (в частности, для обозначения предыдущего состояния датчика в момент первого обращения к нему).

§4.7.4. Настройка обработчика событий

Обработчик событий настраивается в узле .services.event-handler и содержит в себе список событий, которые пользователь желает контролировать и предпринимать по ним какие-либо действия. Анализируются и обрабатываются все события в списке, независимо от порядка их следования. Каждая запись списка содержит следующие параметры.

Имя виртуального датчика (virt-sensor) вводится в виде произвольной текстовой строки, например, ifstate.eth0 . Если указанный датчик не существует в системе, то никакого автоматического слежения за событиями, естественно, не будет, однако несуществующие события несуществующих датчиков можно использовать для запуска действий внешними скриптами (см. ниже).

Начальное и конечное состояния датчика. Событие, на которое реагирует обработчик — это переход датчика из заданного начальное в заданное конечное состояние. При этом одни и те же состояния одного датчика могут фигурировать в нескольких событиях. Для датчиков, имеющих фиксированный набор состояний, значения обоих параметров выбираются из списка; для датчиков, настраиваемых пользователем, названия состояний вводятся как текст. Помимо состояний, определённых на датчике по существу, эти два параметра могут принимать специальные значения:

other — Любое состояние датчика, отличное от указанного в другом из этих двух параметров. В этом случае обработчик будет реагировать на переход в некоторое состояние из любого другого или наоборот, соответственно.

any — Любое состояние датчика. В этом случае событием будет, по существу, любое нахождение датчика в состоянии, которое указано другим параметром.

Использовать значение any следует с осторожностью, поскольку в этом случае каждый опрос датчиков (например, раз в секунду) будет генерировать непрерывный поток событий. Целесообразно применять его только для датчиков, которые не присылают обработчику регулярные сообщения, а самостоятельно определяют факт произошедшего события. В частности, к таким датчикам относятся:

— Интегрированный мультидатчик MS–6. Сообщения от каждого из его датчиков выдаются только в том случае, если результат очередного измерения (раз в секунду) отличен от предыдущего.

— Сетевые интерфейсы. Сообщения выдаются при каждом событии, происходящем на интерфейсе, и не выдаются в остальное время. При этом начальное и конечное наблюдаемые состояния интерфейса могут и совпадать (например, интерфейсу назначен дополнительный IP-адрес, при этом он был в UP и остался в UP).

Если установить для одного из состояний any, а для другого other, то обработчик событий будет реагировать на любое изменение состояния датчика (т.е. начальное состояние не равно конечному). Такой же результат будет иметь пара other-other . Пара any-any будет срабатывать при любом сообщении от датчика.

ПРИМЕЧАНИЕ При первом срабатывании датчика после включения его предыдущим состоянием всегда будет nil — специальное значение, отличное от любого другого (в т.ч. и от any). Эту особенность необходимо учитывать при построении алгоритма работы.

Датчик *netping* имеет особенность: он самостоятельно определяет факт перехода из состояния "не работает" в состояние "работает" и обратно, т.е. событие. Поэтому его сообщения restore или failure означают не текущее состояние, а именно событие. Для него необходимо и достаточно указывать только значение state; параметр prev-state смысла не имеет и по существу его значение всегда будет other .

Для описания состояний допускаются также выражения сравнения, например, ">223", "<-17", "=36.6". Если датчик передаёт алфавитно-цифровые строки, например, "T = 36.6°" , то из них в этом случае выделяется первая максимальная подстрока в формате, похожем на число (т.е. состоящая из цифр, десятичной точки, и, возможно, знака "–" в начале). Таким образом, анализ показаний можно выполнять как на самом датчике (event-generator), так и в обработчике событий (event-handler).

Действие, которое следует выполнить при обнаружении данного события. Действия могут описываться в двух полях: action либо script. В поле script можно писать произвольный скрипт командной среды Linux (ash), в т.ч. вызов nsgsh в пакетном режиме (с учётом ограничений по правам доступа и с возможностью их обхода, см. Часть 1). Заданный скрипт выполняется в режиме демона.

В поле action используются внутренние скрипты *nsgconfd*, язык которых представлен ниже. В качестве действия можно также задать отправку уведомления по TCP на заданный сервер (см. §4.7.6) в следующем формате:

```
action = "#tcpsender.имя"
```

где имя — имя конкретного демона, предназначенного для обработки данного события.

Также допускается использовать в этом поле скрипты Linux; эта возможность сохранена для совместимости с предыдущими версиями и не рекомендуется для новых конфигураций. Различение производится по первому

символу в строке скрипта: если этот символ — точка, то скрипт интерпретируется как внутренний скрипт *nsgconfd*; любой другой — скрипт исполняется в командной среде Linux. Категорически запрещается использовать в данном поле скрипт Linux, вызывающий вложенную оболочку nsgsh в пакетном режиме для выполнения каких-либо действий — это приведёт к зацикливанию системы.

Дополнительно можно указать:

— Время неактивности датчика после каждого срабатывания. Это позволяет исключить "дребезг", т.е. многократные срабатывания датчика, когда наблюдаемая физическая величина колеблется, на фоне долговременной тенденции, около граничного значения. Например, предположим, что температура в помещении может изменяться, в среднем, с скоростью порядка 0,1 градуса в час; но 0,1 градуса — это цена деления термодатчика, и в показаниях возможен случайный разброс в пределах этой величины. Если установить для данного датчика lock=3600, то в следующий раз его показания будут анализироваться только через час, когда температура либо уйдёт уже уверенно в новый диапазон, либо вернётся в исходные пределы.

— Максимальное разрешённое время исполнения скрипта Linux, указанного в параметре script. Если скрипт не закончит свою работу за это время, он будет завершён принудительно. Данный параметр не влияет на работу внутренних скриптов (параметр action), исполняемых в контексте демона конфигурации NSG (*nsgconfd*).

Внутренние скрипты nsgconfd. Язык внутренних скриптов весьма прост и по существу описывает выполнение команд в дереве конфигурации. Синтаксис языка определяется следующими правилами:

— Скрипт может содержать одну или несколько команд. Если команд несколько, то в качестве разделителя может использоваться пробел или точка с запятой.

— Команда представляет собой путь от корня дерева меню NSG, начиная с одного из узлов верхнего уровня, до требуемого листа дерева. Начальная точка, обозначающая корень дерева — обязательна, поскольку именно по ней различаются внутренний скрипт nsgconfd и обычный скрипт Linux.

— Чтобы установить значение параметру, необходимо составить команду в формате путь.параметр=значение . Если значение текстовое и имеет какие-то особенности, например, должно содержать пробелы или знак равенства, или это цифры, которые необходимо интерпретировать как текст, то его следует заключить в двойные кавычки. (После ввода кавычки автоматически заменяются на \").

— Исполнение встроенных команд nsgsh, начинающихся с подчерка (_apply, _new, _remove и т.п.) невозможно, поскольку скрипт не обрабатывается предварительно nsgsh, а исполняется непосредственно в nsgconfd .

— Сокращение команд и параметров не допускается по этой же причине.

— Если требуется выполнить подряд несколько команд внутри одного узла, то общая часть пути может быть вынесена за круглые скобки, последняя общая точка при этом опускается, например:
.tools.mail(server=my.mail.provider;from=NSG700;to=ivan@izba.ru;subject=BEDA;body="ATM is dead";send)
В данном случае сначала несколькими параметрами компонуется оповещение по электронной почте, а затем оно отправляется командой send . Все действия производятся внутри узла tools.mail .

Пример. Светодиодный индикатор устройства NSG–700 отображает состояние соответствующего сетевого интерфейса (например, сменного модуля GPRS/3G или CDMA). Для наглядности использованы различные варианты синтаксиса.

```
services
: event-handler
: : 1
: : : virt-sensor       = "ifstate.s1"
: : : prev-state        = "any"
: : : state             = "up"
: : : action            = ".tools.led.l1(red=off;green=on)"
: : 2
: : : virt-sensor       = "ifstate.s1"
: : : prev-state        = "up"
: : : state             = "other"
: : : action            = ".tools.led.l1.red.on .tools.led.l1.green.off"
```

Помимо четырёх вышеописанных параметров, узел каждого события содержит отладочную команду test-event . С её помощью можно эмулировать наступление указанного события и проверить, как в этом случае будет выполняться желаемое действие.

Журнал обработчика событий содержит все сообщения о состояниях, поступившие в обработчик от датчиков — как те, которые он квалифицировал в качестве заданных событий и исполнил соответствующие действия, так и те, которые он проигнорировал. Если сообщение квалифицировано как событие (или несколько событий одновременно), то номер этого события выводится в этой же строке в квадратных скобках. Помимо известных состояний датчиков, в журнале может указываться состояние nil — состояние неизвестно. Это может

происходить, например, в первом опросе после рестарта системы (когда предыдущего состояния нет по определению), или если внешний датчик отключён от устройства или административно выключен.

Команда simulate-event позволяет эмулировать в обработчике сигнал о некотором состоянии некоторого датчика. В качестве аргумента для неё следует ввести имя.датчика:состояние . В отличие от команды test-event в меню конкретного события, данная команда позволяет проверить не только исполнение заданных действий, но и то, насколько корректно обработчик устанавливает факт наступления событий, представляющих интерес.

Более того, в отличие от test-event, данная команда позволяет эмулировать также события датчиков, не существующих в системе. В частности, пользователь может определить в event-handler следующее событие:

```
services
: event-handler
: : 1
: : : virt-sensor       = "my.zopa"
: : : prev-state        = "any"
: : : state             = "nexopoiiiee"
: : : action            = ".tools.led.red.on"
```

и написать собственную программу или скрипт, исполняемый в командной среде Linux. Если эта программа вызовет nsgsh следующим образом:

```
nsgsh -q .services.event-handler.simulate-event.my.zopa:nexopoiiiee
```

то обработчик интерпретирует это как переход датчика my.zopa в состояние nexopoiiiee и исполняет соответствующее действие, которое отобразит информацию от этого чувствительного датчика.

§4.7.5. Таймеры, планировщик заданий и пользовательские генераторы событий

Помимо событий, относящихся к различным частным объектам в системе, NSG Linux 2.0 имеет несколько типов программных генераторов событий, относящихся к системе в целом.

Таймеры позволяют генерировать заданные состояния через заданные интервалы времени. Эти состояния передаются в обработчик событий в виде timer:*состояние* (т.е. именем виртуального датчика является timer). Таймеры работают в относительном времени, начиная с момента их запуска (командой _apply или при старте системы).

Планировщик заданий генерирует заданные состояния в заданное время суток, дату и день недели. Они передаются обработчику событий в виде scheduler:*состояние* . В отличие от таймеров, планировщик работает в терминах абсолютного календарного времени и даты, поэтому для его корректной работы необходимо, чтобы в системе было выставлено правильное время (вручную с помощью команды .system.clock-set или, предпочтительно, автоматически с помощью клиента NTP — см. §4.2.3).

Следует подчеркнуть, что и планировщик, и таймеры при каждом срабатывании выдают обработчику событий некоторое *состояние*. Для того, чтобы это состояние интерпретировалось обработчиком как *событие*, следует, как правило, установить в обработчике предыдущее состояние prev-state = "any" (в противоположность тому, что обычно требуется для реальных датчиков, имеющих несколько взаимоисключающих состояний).

Синтаксис параметров планировщика для описания времени, в основном, аналогичен синтаксису утилиты cron . Значение каждого из пяти нижеперечисленных параметров — это, в общем случае, список (через запятую), который может содержать следующие элементы:

— Одиночные значения, например, 1,3,6
— Диапазоны значений, например, 1..5; в отличие от cron, диапазон обозначается двумя последовательными точками (..), а не тире
— Диапазоны значений с заданным шагом, например, 1..59/2 — каждая нечётная минута
— Звёздочку (*), обозначающую все возможные значения для данного параметра
— Звёздочку и шаг, например, */2 означает каждый второй элемент, начиная с первого (для минут, часов — 0,2,4 и т.д., для дней, месяцев и дней недели — 1,3,5 и т.п.

Допустимые значения для каждого из параметров:

— Минуты — от 0 до 59
— Часы — от 0 до 23
— Даты — от 1 до 31
— Месяцы — от 1 до 12
— Дни недели — от 1 (понедельник) до 7 (воскресенье); в отличие от cron, воскресенье обозначается только цифрой 7, значение 0 не допускается.

ПРИМЕЧАНИЕ Значения, лежащие вне допустимого диапазона, при анализе отсекаются в первую очередь, и только после этого вычисляются "прореживающие" циклы. Например, если указать

day = "0..10/2"

то планировщик сработает 1, 3, 5, 7 и 9 числа месяца.

Даты, выходящие за число дней в конкретном месяце (например, 31 февраля) также отсекаются автоматически. При этом, в некоторых случаях, неизбежен сбой регулярности, скажем, day = "1..31/2" сработает подряд 31 января и 1 февраля.

ВНИМАНИЕ По умолчанию, для всех пяти параметров установлено значение *, т.е. состояния планировщика передаются каждую минуту. Для получения содержательных настроек требуется, как правило, указать явным образом большинство параметров.

Пример. Следующий пример позволит вам поселить в устройстве NSG забавного зверька — тамагочи, который будет время от времени развлекать вас SMS-ками:

```
services
: event-handler
: : event-generators
: : : scheduler
: : : : enable              = "true"
: : : : 1
: : : : : event-name       = "hungry"
: : : : : min              = "30"
: : : : : hour             = "7,18"
: : : : 2
: : : : : event-name       = "thirsty"
: : : : : min              = "0"
: : : : : hour             = "8..20/6"
: : : : 3
: : : : : event-name       = "nasty"
: : : : : min              = "15"
: : : : : hour             = "3"
: : : : : wday             = "2,4,7"
: : 1
: : : virt-sensor          = "scheduler"
: : : prev-state           = "any"
: : : state                = "hungry"
: : : action               = "at2 sms --tcpport 50000 +79876543210 \"I want to eat! Your $HOSTNAME\""
: : 2
: : : virt-sensor          = "scheduler"
: : : prev-state           = "any"
: : : state                = "thursty"
: : : action               = "at2 sms --tcpport 50000 +79876543210 \"I want to drink! Your $HOSTNAME\""
: : 3
: : : virt-sensor          = "scheduler"
: : : prev-state           = "any"
: : : state                = "nasty"
: : : action               = "at2 sms --tcpport 50000 +79876543210 \"I want sex NOW!!! Your $HOSTNAME\""
```

Заключительный тип общесистемных виртуальных датчиков, предусмотренный в NSG Linux 2.0 — это пользовательские скрипты и программы. Пользователь может написать и запустить в среде Linux любой исполняемый файл, который будет записывать в стандартный поток вывода строки вида:

датчик:состояние или
датчик<TAB>состояние

Эти строки рассматриваются обработчиком событий как указанное состояние указанного виртуального датчика. Основным параметром данного типа генераторов является строка для запуска скрипта; как правило, это путь к исполняемому файлу от корня системы, например, /etc/myscripts/crow.counter . Скрипт запускается в командной оболочке Linux (ash) в качестве демона, поэтому обрамление #!/bin/sh (...)& не требуется (оно добавляется автоматически).

§4.7.6. Служба TCP-уведомлений

Отправка уведомлений по TCP — одна из возможных реакций на срабатывание датчиков. Служба уведомлений состоит из одного или нескольких демонов, каждый из которых может обрабатывать события от одного или нескольких генераторов. Обязательными параметрами демона являются IP-адрес и порт TCP сервера, на которые будут посылаться уведомления. Демон устанавливает TCP-соединение с этим сервером и в случае возникновения события, с которым он ассоциирован, посылает по этому соединению текст в формате:

имя_датчика,предыдущее_состояние,текущее_состояние

Дальнейшие действия определяются алгоритмами работы и настройками программного обеспечения сервера: вывести аварийное сообщение на экран дежурного оператора, записать в журнал или в базу данных, и т.п. Серверное ПО представляет собой отдельный программный продукт, разрабатываемый NSG или сторонними производителями, под различные операционные системы.

Для удобства восприятия каждому демону можно назначить имя (description), удобное для восприятия, например, "Банкомат на просп.Ленина, 1". Это имя выводится на сервере мониторинга NSG в качестве имени наблюдаемого объекта. Если имя не указано, то вместо него выводится имя самого демона.

В отсутствие наблюдаемых событий демон регулярно шлёт пакеты TCP *keepalive*. По этим пакетам сервер видит, что устройство NSG работает, связь с ним есть, но событий не происходит. Отсутствие пакетов *keepalive*, в большинстве случаев, по сути своей должно расцениваться сервером как нештатная ситуация на клиенте — так же, как и срабатывание аварийных датчиков.

ПРИМЕЧАНИЕ Параметры keepalive и retry демона TCP-уведомлений относятся только к нему самому, как к клиенту, и определяют, с каким интервалом будут посылаться пакеты и через сколько потерянных пакетов подряд клиент будет считать соединение умершим. Логика действий на серверной стороне определяется настройками сервера.

Для проверки работы демона и сервера в совокупности предусмотрена команда разовой отправки уведомления в ручном режиме. Текст уведомления, который должен был бы быть послан в случае реального срабатывания датчика, вводится в поле ввода команды.

§4.7.7. Сервер TCP-мониторинга

Ответной частью TCP-мониторинга является сервер, который может исполняться как в составе NSG Linux 2.0, так и в качестве отдельного приложения на компьютере под управлением ОС Linux или Windows. Сервер принимает сообщения от удалённых клиентов по заданному порту TCP и выводит состояние всех наблюдаемых клиентов и их датчиков в виде отдельной Web-страницы, также доступной по определённому порту TCP. Внешний вид и содержимое этой страницы могут меняться в зависимости от установленных на ней же режимов просмотра.

Цвет и атрибуты записей на странице мониторинга отражают состояние датчиков и, соответственно, клиентов, на которых эти датчики расположены. Кодовый цвет каждого состояния указывается в описании датчика (см. след. параграф). Мигание записи означает, что связь с клиентом отсутствует.

В состав данной версии NSG Linux 2.0 входит экспериментальная реализация сервера, с минимальным набором возможностей. В перспективе предполагается дальнейшее развитие механизма мониторинга, основанное на запросах и потребностей пользователя.

Настройка сервера мониторинга выполняется в узле .services.event-server . Основными параметрами сервера являются его административное состояние, номер порта TCP для входящих сообщений от клиентов, и номер порта TCP для просмотра Web-броузером.

В частности, сервер мониторинга должен (по форме и по существу) быть запущен для того, чтобы показания датчиков, посылаемые в виде TCP-уведомлений, выводились в мониторинге *ui*TCP (см. Часть 6).

§4.7.8. Формат отображения событий на сервере

В описания событий можно включать текущее численное значение, полученное от датчика. Значение указывается в параметре event-name в формате $(VAL). Возможно также использовать стандартные последовательности HTML для вывода спецсимволов, например, ° для знака "градус":

```
event-name = "too hot: T=$(VAL)&deg;C"
```

Для наглядного отображения состояния датчиков в Web-интерфейсе мониторинга можно использовать цветовые метки. Текущая версия сервера предусматривает использование 5 цветов, кодируемых числами:

+2 или 2	Красный
+1 или 1	Жёлтый
0	Зелёный
–1	Голубой
–2	Фиолетовый

Номер цвета, соответствующий каждому состоянию, указывается в описании этого состояния (см. §4.7.3, §4.7.5) через последовательность &&, например:

```
port
: usb1
: : type                = "multisensor"
: : event-generator
: : : enable            = true
: : : TEMP-states
: : : : 1
: : : : : from          = "50"
: : : : : event-name    = "hot&&2"
: : : : 2
: : : : : from          = "30"
: : : : : event-name    = "warm&&1"
: : : : 3
: : : : : from          = "5"
: : : : : event-name    = "normal&&1"
: : : : 4
: : : : : from          = "-5"
: : : : : event-name    = "shilly&&-1"
: : : : 5
: : : : : to            = "-5"
: : : : : event-name    = "frozen&&-2"
```

В данном случае температура, измеренная датчиком, будет показываться красным, желтым, зелёным, голубым и фиолетовым цветом, соответственно, от недопустимо высокой до недопустимо низкой.

Суффикс &&цвет используется только при передаче состояния датчика по TCP серверу мониторинга NSG. В других случаях он игнорируется, и указывать его не следует — в частности, в обработчике событий.

§4.8. Автоматизация операций между несколькими устройствами NSG

§4.8.1. Общие сведения

Для автоматизации действий, выполняемых с одного устройства NSG на другом, используется клиент SSH. Аутентификация при этом производится по ключу, что позволяет обойтись без ручного ввода пароля и выполнять всю процедуру в полностью автоматическом режиме. Подробнее о генерации ключей клиента SSH и их экспорте на удалённую машину см. §4.5.2. На удалённой стороне, естественно, должен быть запущен сервер SSH; никаких дополнительных настроек, сверх этого, не требуется.

После того, как клиент SSH настроен для автоматического входа на удалённую машину, он может быть запущен в автоматическом режиме скриптом следующего вида:

```
ssh user@host "команда"
```

Данный механизм используется командами и процедурами, встроенными в командную оболочку NSG Linux 2.0, а также может использоваться явным образом в пользовательских скриптах.

Примеры автоматизированных операций приведены в следующих параграфах.

§4.8.2. Ручное копирование конфигурации

NSG Linux 2.0 предусматривает возможность переноса конфигурации с удалённого устройства на локальное. Данная функция наиболее полезна для резервирования центральных серверов VPN, которые обслуживают большое число клиентов, и список клиентов постепенно изменяется со временем. Благодаря ей, администратор может выполнять изменения только на одном основном сервере, а затем копировать их на резервный(-е) одной командой вручную или автоматически по расписанию.

Настройка собственно копирования производится в узле system.confsync на резервном устройстве. Обязательными параметрами являются IP-адрес основного устройства и имя для входа на него (в данном случае — только root). Список import содержит узлы конфигурации, которые необходимо скопировать, а список exclude — возможные исключения из этих узлов для более тонкой настройки. Конфигурация, принятая с основного устройства согласно этим спискам, полностью замещает соответствующие узлы на резервном устройстве.

ПРИМЕЧАНИЕ Строго говоря, процедура построения конфигурации состоит из двух этапов:
- — Конфигурация, принятая с основного устройства согласно списку import, полностью замещает соответствующие узлы на резервном устройстве;
- — Узлы, перечисленные в списке exclude, восстанавливают исходные значения.

После настроек, описанных выше, непосредственно процедура переноса конфигурации выполняется разовой командой syncronize . Результат её выполнения зависит, в первую очередь, от параметра save:

save = false Процедура выполняется фиктивно, предполагаемые изменения отражаются только в журнале, но никакие реальные изменения ни в текущей конфигурации устройства, ни в конфигурации, сохранённой в энергонезависимой памяти, не производятся. Этот режим можно использовать для проверки того, что должно получиться в результате синхронизации, и убедиться, что при этом не возникает конфликтов или нежелательных изменений. (В этом случае не следует использовать quiet = true.)

save = true Полученная конфигурация сохраняется в энергонезависимой памяти. Будет ли она применена немедленно — зависит от параметров apply .

Узел apply — это список, который может тождественным списку import, а может и не совпадать с ним. В простейшем случае можно написать apply.1 = ".", и устройство будет применять конфигурацию полностью от корня дерева. По существу, такая процедура может быть (в зависимости от набора изменяемых узлов) эквивалентна полному рестарту устройства, что не всегда желательно. Поэтому списком apply можно отдельно перечислить узлы, которые необходимо автоматически применять после импорта конфигурации.

Ситуация окажется сложнее, если на данной машине в это время работает другой администратор; тогда скрипт импорта конфигурации получит только права пользователя, и в результате процедуры применения и сохранения конфигурации завершатся с ошибкой. Параметр force позволяет разрешить этот конфликт в пользу скрипта: сессия другого администратора принудительно завершается, и скрипт получает права администратора. После завершения его работы можно снова войти в систему, при этом в пользовательском интерфейсе будет выведена уже сохранённая конфигурация. Эту особенность необходимо учитывать при последующей работе с данным устройством.

Не следует использовать параметр force = true , если синхронизация выполняется разово вручную, чтобы не потерять свою собственную сессию со сделанными в ней изменениями. Если force = false, то скрипт проверяет, нет ли в текущей сессии неприменённых и несохранённых изменений. Если они есть, скрипт предлагает применить и сохранить их; если нет — то запускается от имени текущего пользователя и по завершении работы возвращает управление в эту же пользовательскую сессию.

Помимо собственно дерева конфигурации (файла /etc/nsgconfig), в процедуру можно включить любые другие файлы и директории. В частности, это могут быть файлы сертификатов X.509 для различных служб, или директории, в которых они хранятся (*/etc/uitcp/certs, /etc/ipsec.d* и др.). Из директорий можно исключить отдельные файлы и поддиректории. В последнем элементе пути (собственно имени файла или директории) допускается использование подстановочного символа *.

ПРИМЕЧАНИЕ Если на локальном устройстве существуют файлы с такими же путями/именами, как на мастер-устройстве, то они будут заменены. Если в локальных директориях есть файлы, которых нет на мастер-устройстве, то они останутся как есть и удалены не будут.

После синхронизации может потребоваться выполнение некоторых дополнительных действий, например, после синхронизации сертификатов *ui*TCP следует заново сгенерить хэш-файлы сертификатов. Скрипты для этого записываются в параметр script. Данный скрипт исполняется только при установленном параметре save=true и успешном завершении всех предыдущих операций (копирование, объединение, применение конфигураций).

Отладочный параметр quiet позволяет уменьшить объём информации, выводимой в журнал.

Пример. Резервный сервер *ui*TCP скачивает конфигурацию с основного. При этом он не изменяет конфигурацию портов и другие узлы, уникальные для каждой машины. В узле uitcp из импорта исключается узел tunnelListener.host, поскольку он также содержит уникальный адрес публичного интерфейса той или другой машины:

```
system
: confsync
: : host              = "192.168.12.34"
: : user              = " root"
: : import
: : : 1               = ".tunnel.uitcp.configure"
: : exclude
: : : 1               = ".tunnel.uitcp.configure.tunnelListener.host"
: : file
: : : 1               = "/etc/uitcp/certs"
: : exclude-file
: : : 1               = "/etc/uitcp/certs/_*"
: : : 2               = "/etc/uitcp/certs/server*"
: : : 3               = "/etc/uitcp/certs/capath"
: : script            = "rehash"
```

§4.8.3. Автоматическое копирование конфигурации

Для автоматического переноса конфигурации можно использовать механизм обработчика событий, а в качестве генератора событий — таймер (через заданные интервалы времени) или планировщик задач (в заданное время суток, число месяца и т.п.). Подробнее о данных инструментах см. §4.7. Пример настройки для выполнения синхронизации клиентов *ui*TCP ежедневно в 02:45:

```
services
: event-handler
: : event-generators
: : : scheduler
: : : : enable          = "true"
: : : : 1
: : : : : event-name    = "sync-conf"
: : : : : min           = "45"
: : : : : hour          = "2"
: : 1
: : : virt-sensor       = "scheduler"
: : : prev-state        = "any"
: : : state             = "sync-conf"
: : : script            = "nsgconfsync root@192.168.12.34 -i=.tunnel.uitcp.configure.clients ↵
                          --apply=. tunnel.uitcp.configure.clients --file=/etc/uitcp/certs ↵
                          --excludefile=/etc/uitcp/certs/_* --excludefile=/etc/uitcp/certs/server* ↵
                          --excludefile=/etc/uitcp/certs/capath --write --force --script=rehash"
```

Для генерации ключей SSH, их передачи на удалённое устройство и синхронизации конфигураций используется утилита nsgconfsync, исполняемая непосредственно из командной строки Linux. В данном примере использован именно такой вызов, чтобы избежать лишней прослойки в виде вызова *nsgsh* в пакетном режиме и связанных с этим возможных проблем с правами текущей сессии. Параметры командной строки nsgconfsync аналогичны параметрам узла .system.confsync . Подробное описание данной утилиты см. в Части 7.

§4.8.4. Мониторинг *ui*TCP

Механизм доступа к удалённым устройствам NSG по SSH используется также во встроенной функции мониторинга нескольких серверов *ui*TCP с одного. Поскольку заранее неизвестно, на каком сервере окажется какой клиент в некоторый момент времени, то мониторинг на одном сервере не может дать полной картины: клиент, отсутствующий в данный момент на данном сервере, может работать через какой-то другой сервер, а может быть недоступен вообще. Данный механизм позволяет контролировать состояние всей системы *ui*TCP с одного сервера, в одном окне Web-броузера. Подробно о его настройке см. Часть 6.

§4.8.5. Исполнение скриптов на удалённом устройстве

Механизм обработчика событий, встроенный в NSG Linux 2.0, позволяет задать в качестве условия исполнения скрипта разнообразные события, происходящие в системе — как на самом устройстве NSG, так и на присоединённых к нему датчиках. В качестве реакции на это событие может быть задан скрипт, исполняющий по SSH заданную команду на удалённом устройстве. В приведённом ниже примере производится трансляция состояния "сухих контактов" с локальной машины на удалённую по адресу 123.45.67.89:

```
port
: 1-wire
: : device
: : : swt2-3AC10D0100000054
: : : : type            = "IC-2dio"
services
: event-handler
: : 1
: : : virt-sensor       = "3AC10D0100000054.circuit-A"
: : : prev-state        = "other"
: : : state             = "on"
: : : action            = "ssh root@123.45.67.89 \"nsgsh -q .port.1-wire.swt-3AC10D0100000067.circuit.A.close\""
: : 2
: : : virt-sensor       = "3AC10D0100000054.circuit-A"
: : : prev-state        = "other"
: : : state             = "off"
: : : action            = "ssh root@123.45.67.89 \"nsgsh -q .port.1-wire.swt-3AC10D0100000067.circuit.A.open\""
```

Значение prev-state="other" в данном примере использовано для того, чтобы под требуемое событие подпадало также начальное показание датчика. Например, событию "включение" должны соответствовать переход nil—>on при включении датчика и переходы off—>on при последующих опросах. Тривиальные переходы on—>on и off—>off, сооответствующие неизменному состоянию входной цепи, как можно видеть, при этом не считаются событиями и не вызывают никаких действий.