**Григорьев В.М**

.

# Виртуальная лаборатория по компьютерным сетям

**Днепропетровск**
**2011**

## Введение

Сетевые маршрутизаторы работают под управлением операционных систем. В ряде случаев эти операционные системы можно запустить внутри виртуальных машин. Сетевые устройства в сети соединены каналами связи. Для организации виртуальной сетевой лаборатории надо соединить между собой виртуальные машины, в которых запущены операционные системы сетевых устройств.

На лабораторном занятии в такой виртуальной лаборатории по компьютерным сетям обычно присутствует десяток студентов, и каждый из них изучает сетевую топологию, состоящую из нескольких сетевых устройств. Одновременно работают десятки виртуальных машин. Возникает задача такого выбора операционной системы сетевого устройства и виртуальной машины, чтобы обеспечить студентам комфортную одновременную работу при условии ограниченности ресурсов хост-компьютера.

Идея виртуальных сетевых лабораторий не нова, и для их реализации существуют специализированные решения. Для моделирования устройств Cisco, работающих под управлением операционной системы IOS, используются программы Cisco Packet Tracer и Boson NetSim, имеющие удобный графический интерфейс, позволяющий быстро создавать достаточно сложные сетевые топологии. Однако встроенная в них урезанная версия IOS позволяет изучать лишь сетевые технологии начального уровня.

Больший интерес представляет использование операционных систем реальных сетевых устройств. Возникают следующие вопросы, касающиеся выбора операционной системы устройства и виртуальной машины для запуска этой операционной системы: сколько ресурсов хост-машины потребляет виртуальная машина; каково время запуска операционной системы внутри виртуальной машины; какие средства предоставляют виртуальные машины для соединения между собой запущенных внутри них операционных систем и как быстро создать сложную сетевую топологию из десятков устройств.

Для моделирования сетевых топологий широко используется контейнер виртуальных машин GNS3. Для создания сетевых топологий в GNS3 используется технология Drug-and-Drop: зацепил устройство мышью и перетащил его на рабочее поле. GNS3 поддерживает две виртуальные машины: Dynamips и Qemu. Выбор именно этих машин для включения в GNS3 обусловлен наличием в их составе развитых средств для соединения между собой операционных систем.

Виртуальная машина Dynamips позволяет запустить внутри себя реальную IOS для очень широкого класса устройств Cisco. Однако при работе с Dynamips следует подбирать параметры для уменьшения нагрузки на центральный процессор. Без должных настроек Dynamips использует все ресурсы компьютера уже для топологии из трёх маршрутизаторов.

Под Qemu работает весьма широкий класс сетевых, встроенных и мобильных операционных систем: Juniper JunOS, Vyatta, Openwrt, файерволы Cisco IDS, Google Android, Mikrotik RouterOs и др. Наш выбор был сделан в пользу Qemu в составе GNN3.

Нельзя не упомянуть виртуальную машину IOU для Cisco IOS. В ней можно запустить пару операционных систем Cisco IOS с весьма мощной функциональностью, и она не требует такой настройки, как Dynamips. К сожалению, IOU не обладает графическим интерфейсом.

Следовало определиться, в чём работать: в Windows или в Linux. GNS3 и Qemu задуманы, сделаны и развиваются в Linux. Qemu под Linux поддерживает аппаратную виртуализацию KVM. Qemu под Windows не поддерживает KVM и при запуске нескольких экземпляров

Qemu используется только одно ядро центрального процессора, что существенно замедляет работу с большими сетевыми топологиями.

Возникает вопрос выбора дистрибутива Linux. GNS3 написан на Python и требует библиотеки Qt4. После ряда экспериментов с различными дистрибутивами Linux по установке GNS3 из исходных кодов выбор пал на настольную 64-х разрядную версию Ubuntu.

Виртуальная машина Qemu в составе GNS3 под управлением Ubuntu оказалась лучшим выбором для организации виртуальной лаборатории.

Определим операционную систему сетевого устройства для запуска под Qemu. Если потребовать, чтобы устройство поддерживало сетевую технологию MPLS, то выбор сразу сократится: это либо операционная система JunOS фирмы Juniper, либо RouterOS фирмы Mikrotik.

По объёму потребляемых ресурсов JunOS существенно превосходит RouterOS. Например, на процессоре с частотой 2.4 ГГц время загрузки RouterOS версии 5.5 в Qemu под Ubuntu составляет 6 (шесть) секунд, а JunOS версии 10.1R1.8 грузится 80 секунд. При выключенной поддержке KVM время загрузки было в два раза больше. RouterOS требует минимум 64 Мб памяти, JunOS — 512 Мб. Образ диска RouterOS - 40 Мб, JunOS - 2.6 Гб.

Для обеспечения непрерывности процесса обучения студентов было решено организовать доступ к виртуальной лаборатории через Интернет. Использовался компьютер с доменным именем lib.dnu.dp.ua, работающий под управлением операционной системы Windows server 2008 R2 Sp1. На этом компьютере активирована роль виртуальных машин HYPER-V. В HYPER-V запущена виртуальная машина под управлением операционной системы Ubuntu-11.04-desktop-amd64. В Ubuntu установлены GNS3 0.7.4, Qemu 13.0. Доступ к Ubuntu из внешнего мира осуществляется через OpenVPN с использованием RSA-сертификатов.

Для доступа к окну GNS3 на удалённом рабочем столе Ubuntu используется технология фирмы Nomachine, основанная на X-протоколе. Для удалённого подключения к GNS3 можно использовать и обычный X-сервер для Windows, например Xming. Организована удалённая работа в лаборатории и по протоколу VNC.

Qemu в Ubuntu под управлением HYPER-V работает несколько медленнее, чем Qemu в Ubuntu на реальном компьютере. Это обусловлено, в частности, и тем, что аппаратный ускоритель KVM не работает на виртуальной аппаратуре HYPER-V. Так время загрузки RouterOS в Qemu под Ubuntu в HYPER-V составила 12 секунд против 6 секунд в Qemu под Ubuntu на реальном процессоре частотой 2.4 ГГц. Хост-компьютер HYPER-V имеет 8-ядерный процессор Intel Core i7 950 3066 МГц.

Многоядерность процессоров уменьшает время загрузки нескольких экземпляров RouterOs под управлением Qemu. Так время загрузки 10 экземпляров RouterOS на реальном 2-х ядерном процессоре частотой 2.4 ГГц. и виртуальном 4-х ядерном процессоре HYPER-V оказалось одинаковым и равным 30 секундам.

Число одновременно работающих экземпляров RouterOs под Qemu определяется свободной памятью хост-машины. Так выделение для Ubuntu под HYPER-V 8 Гб памяти позволило запустить сто экземпляров RouterOs. При 4-х виртуальных процессорах время загрузки составило 300 секунд.

Связь между хост-машиной Ubuntu и устройствами внутри виртуальной лаборатории осуществляется как через консоль по протоколу Telnet, так и с использованием tap-интерфейсов. Для этой цели в Ubuntu создано несколько сотен tap-интерфейсов. Каналы связи между сетевыми устройствами в сетевой топологии организованы с помощью UDP-протокола. Для избежания конфликтов при одновременном доступе к лаборатории

нескольких пользователей они получают непересекающиеся диапазоны UDP-портов и TCP-портов для консоли, а также разные наборы tap-интерфейсов.

Организована маршрутизация из локального компьютера в запущенные экземпляры RouterOs в Qemu. При плохом качестве связи к lib.dnu.dp.ua пользователь может не использовать удалённый рабочий стол Ubuntu и осуществлять настройку сетевых устройств напрямую с помощью локально запущенных утилит Winbox, которые удалённо подключаются к устройствам в топологии.

Операционная система RouterOs поддерживает практически все современные сетевые технологии. Это позволило разработать лабораторный практикум для изучения следующих сетевых технологий: Ethernet-мосты, DHCP, балансировка нагрузки, EoIP, NAT, маршрутизация RIP, OSPF и BGP, перераспределение маршрутов, PPP, PPTP, SSTP, L2TP, OpenVPN, виртуальные частные сети 2-го и 3-го уровня, IPSec, MPLS, VPLS, VRF. Представляется нереальной комплектация учебной лаборатории вуза реальным сетевым оборудованием, позволяющим практически освоить перечисленные сетевые технологии. С лабораторным практикумом можно ознакомиться на сайте http://lib.dnu.dp.ua.

Образ установочного CD операционной системы RouterOs находится на сайте фирмы Mikrotik в свободном доступе, но имеет одно ограничение – время непрерывной работы составляет одни сутки. Этих суток хватает пользователю на несколько лабораторных занятий. По истечении срока пользователи должны сохранить настройки операционных систем RouterOs и сетевую топологию выполняемой лабораторной работы, запустить сохранённую топологию без настроек и восстановить настройки RouterOs. Написаны скрипты, которые соединяются с помощью ssh с устройствами в топологии, посылают в устройства команды для создания или восстановления резервных копий и загружают или выгружают эти копии. Переустанавливать при этом операционную систему RouterOs не надо. Это объясняется тем, что GNS3 при первом старте каждого устройства в топологии создаёт для него копию образа диска заранее установленной операционной системы и не изменяет оригинал.

Для получения RSA-сертификата и тестового доступа к виртуальной сетевой лаборатории следует обратиться к автору и посетить сайт http://vmg.pp.ua.

Показано, что виртуальная учебная лаборатория по компьютерным сетям может быть создана с помощью виртуальных машин Qemu в оболочке GNS3 с использованием операционной системы RouterOS фирмы Mikrotik. Виртуальная учебная лаборатория позволяет вместо реальной аппаратуры применять соединённые между собой виртуальные машины, в которых запущены операционные системы сетевых устройств.

# 1. Настройка домашнего компьютера

**Выбор платформы.**
**Сетевая архитектура виртуальной лаборатории**
**Способы выполнения лаб и курсовой по сетям**
**Установка GNS3, Qemu и Winbox под Ubuntu на компьютере дома**
**Установка tap-интерфейсов под Ubuntu**
**Работа в Windows**
**Удалённая работа из Windows и Ubuntu**

## Выбор платформы.

Для самостоятельного глубокого изучения сетевых технологий совсем не обязательно иметь под рукой достаточного количества различных сетевых устройств -маршрутизаторов и коммутаторов. Вспомним, что виртуальные машины позволяют запускать внутри себя различные операционные системы (ОС) и, что маршрутизаторы работают под управлением своих ОС. Сетевые устройства в сети соединены каналами связи. Поэтому, для моделирования сетевой топологии надо соединить между собой операционные системы маршрутизаторов, запущенные под виртуальными машинами. Можно воспользоваться любой виртуальной машиной (Hyper-V, VirualBox, VmWare и т.д.), но для создания сложных сетевых топологий как нельзя лучше подходит виртуальные машины Qemu и Dynamips.

Существуют специализированные решения. Для моделирования устройств Cisco, работающих под управлением ОС IOS, используются программы Packettracer и Boson NetSim, которые поддерживают лишь достаточно урезанные варианты системы IOS. К достоинствам Packettracer и Boson относится удобный графический интерфейс, позволяющий быстро создавать достаточно сложные сетевые топологии.

Для моделирования сложных сетевых технологий широко используется контейнер виртуальных машин GNS3. GNS3 поддерживает технологию drugAndDrop для создания сетевых топологий: зацепил устройство мышью и перетащил его на рабочее поле. GNS3 поддерживает две виртуальные машины Dynamips и Qemu.

Виртуальная машина Dynamips позволяет запустить внутри себя реальную ОС IOS для очень широкого класса устройств Cisco. Однако при работе с Dynamips следует подбирать параметр idlepc для уменьшения нагрузки на центральный процессор. Этому подбору надо уделять особое внимание, что требует должной ответственности со стороны пользователя. У нерадивого пользователя Dynamips съест все ресурсы уже для топологии из трёх маршрутизаторов.

Под Qemu работает весма широкий класс сетевых, встроенных и мобильных операционных систем: Juniper JunOS, vyatta, openwrt, файерволы Cisco ASA, PIX, IDS, Google Android и др. Причём Qemu поддерживает не только архитектуру процессоров Интел.

Отдельно стоит виртуальная машина IOU для Cisco IOS. Хотя в ней можно запустит ряд IOS с весьма мощной функциональностью и она не требует такой тонкой настройки как Dynamips, она не обладает графическим интерфейсом, что является недостатком при обучении персон не желающих возиться с интерфейсом командной строки.

Итак, выбор сделан: Qemu под GNS3. Сделаем следующий шаг. Определим, что будем запускать под Qemu, какую ОС и для сетевых устройств какого производителя. Если потребовать, чтобы устройство поддерживало VPN и MPLS, то выбор сразу сократиться: это либо ОС JunOS фирмы Juniper, либо ОС RouterOS литовской фирмы Mikrotik. По объёму потребляемых ресурсов эмуляция в Qemu ОС JunOS фирмы Juniper существенно

превосходит ОС RouterOS фирмы Mikrotik. Кроме того для RouterOS фирма Mikrotik разработала для конфигурации прекрасный десктоп- клиента Winbox.

Выбор сделан RouterOS под Qemu под GNS3.

Осталось определиться, в чём работать в Windows или в Linux. Следует помнить, что и GNS3, и Dynamips, и Qemu и IOU задуманы, сделаны и развиваются в Linux. Исходные коды GNS3, Dynamips, и Qemu имеются в свободном доступе (IOU – нет). Нашлись люди, которые на основании этих исходных кодов создали аналоги этих продуктов под Windows. Требовать или хотя бы ждать от таких приложений стабильной работы не приходится.

Итак имеем RouterOS под Qemu под GNS3 под Linux.

Линуксов много. Какой брать? GNS3 написан на Python и требует библиотеки Qt4. На Ubuntu (десктоп 32 разряда) GNS3 стал без проблем. Попытка приручить GNS3 на RedHat Fedora или Centos сразу не удалась и была оставлена. Было решено собирать Qemu из исходных кодов, что и было проделано на Ubuntu (десктоп 32 разряда).

Итак RouterOS под Qemu под GNS3 под Ubuntu (десктоп 32 разряда).

## Сетевая архитектура виртуальной лаборатории

Имеются две независимые лаборатории. Одна доступна только на кафедре по адресу 192.168.14.56. Вторая доступна в университете по адресу 192.168.10.11 и удалённо по адресу 212.3.125.93. Доменное имя vmg.pp.ua взаимно однозначно соответствует этому адресу 212.3.125.93/24 и он назначен некоторому устройству в сети ДНУ. Это устройство осуществляет для входящих пакетов преобразование этого адреса в адрес 192.168.10.11/24 для портов http, rdp, 8000-8004. Адрес 192.168.10.11/24 назначен на сетевую карточку компьютера LIB со следующими характеристиками. MB Asus P6X58D Premium, 8-ми ядерный процессор Intel Core i7 950 3066MG, RAM 12G, HDD 2xWD1-1002FAEX. LIB работает под управление Windows server 2008r2sp1.

В LIB активирована роль виртуальных машин HYPER-V. Под HYPER-V запущена виртуальная машина U1104 под управлением операционной системы Ubuntu 11.04 desktop i386. В рамках виртуальной сети HYPER-V настроено 3 адаптера коммутатора виртуальной сети Microsoft: один внешний с именем inet, подключенный к физическому сетевому адаптеру и два внутренних с именами NAT и oVPN.

Адаптер inet имеет адрес 192.168.10.11/24 и через него осуществляется выход в Интернет.

В LIB активирована роль маршрутизации и удалённого доступа и в ней организовано преобразование адресов из внутренней сети адаптера NAT во внешнюю сеть адаптера inet. На адаптер NAT назначен адрес 192.168.1.1/24.

На адаптер oVPN назначен адрес 192.168.0.3/24.

Эти три сетевых адаптера NAT, oVPN и inet подключены и к виртуальной машине U1104. В ней на адаптер NAT назначен адрес 192.168.1.2/24 и шлюз 192.168.1.1, который расположен в хост-машине LIB. Багодаря NAT в LIB, через этот интерфейс из виртуальной машины осуществляется доступ в Интернет, что необходимо для получения обновлений операционной системы Ubuntu.

На адаптер inet в Ubuntu назначен адрес 192.168.10.254/24, что позволяет компьютерам внутри университетской сети работать с виртуальной машиной без использования OPENVPN.

В Ubuntu в качестве менеджера виртуальных машин используется Qemu. Под Qemu запускается множество операционных систем RouterOS маршрутизаторов фирмы Mikrotik. Для связи виртуальных машин с внешним используются tap-интерфейсы. Для каждого

студента D в Ubuntu уже созданы tap-интерфейсы tapD00 tapD01 ... tapD07 tapD10 ... tapD19. Tap-интерфейсы помещены в мосты BD00 BD01 ... BD07 BD10 ... BD19. На мосты назначены адреса 10.D.0.2, 10.D.1.2 … 10.D.7.2, 10.D.10.2 … 10.D.19.2. Маска /24. Все адреса мостов для одного студента образуют сеть 10.D.0.0/16. Например для D=15 (student15) это сеть 10.15.0.0/24

На адаптер oVPN в Ubuntu назначен адрес 192.168.0.1/24. Именно по этому адресу осуществляется удалённый доступ к этой виртуальной машине через OPENVPN.

Служба OPENVPN-сервера запущена в LIB со следующей конфигурацией адресов и маршрутов

**port 8002** клиенты подключаются к этому порту
**proto tcp** по этому протоколу
**dev tun** в режиме маршрутизации
**ca ca.crt** для них осуществляется
**cert lib.crt** проверка
**key lib.key** сертификатов
**server 192.168.3.0 255.255.255.0** После соединения сервер получает адрес 192.168.3.1/24
**push "route 192.168.0.0 255.255.255.0"** Клиенты получают маршрут к сети 192.168.0.0/24 адаптера oVPN в LIB и виртуальной машине U1104, что позволяет внешним компьютерам подключаться к Ubuntu через OPENVPN, используя адрес 192.168.0.1.
**topology subnet** Позволяет избегать назначение адресов с маской /30
**tcp-nodelay** Отменить алгоритм Nagle (чуть быстрее)
**client-config-dir "\\OpenVPN22\\config\\ccd\\"** Клиент берёт из этой папки адрес и маршрут из файла с именем равным common name своего сертификата. Пример файла student15
**ifconfig-push 192.168.3.100 255.255.255.0** клиент получает адрес 192.168.3.115/24
**push "route 10.0.0.0 255.255.0.0"** и маршрут в сторону сети 10.15.0.0/16 для удалённого подключения к маршрутизаторам внутри Qemu.

В Ubuntu прописан статический маршрут на сеть OPENVPN 192.168.3.1/24 через шлюз 192.168.0.3 расположенный в LIB (адаптер oVPN). Это явлчется обратным маршрутом для подключения внешних компьютеров к Ubuntu через OPENVPN, используя адрес 192.168.0.1.

**Способы выполнения лаб и курсовой по сетям**

Лабы можно выполнять либо локально на своём компьютере либо путём удалённого доступа к компьютеру vmg.pp.ua либо путём удалённого доступа к компьютеру по адресу 192.168.14.56.

Самый быстрый старт дома локально на своём компьютере -это взять GNS3 под Windows с сайта gns3.net и установить. Qemu интегрирован в GNS3 под Windows. К сожалению, даже на хорошем компьютере, для топологии из четырёх маршрутизаторов будут тормоза. Из ядер процессора Qemu под Windows используется только одно. Второй вариант – взять Ubuntu с сайта производителя и установить под виртуальную машину и затем устанавливаем там Qemu и GNS3. В качестве виртуальной машины можно воспользоваться бесплатными продуктами от Vmware, например Vmware player. Работает более стабильно, но даже на хорошем компьютере для топологии из 5-6 маршрутизаторов будут тормоза.

Лучшее решение - это поставить Ubuntu на жёсткий диск. Надо минимум 2 раздела диска и не обязательно главных. Один раздел надо под корневую файловую систему и второй для свопинга. Для первого раздела будет достаточно 30-40G. Для свопинга рекомендуется размер равный удвоенному размеру памяти. Для создания новых разделов на

существующем диске следует его дефрагментировать и затем воспользоваться утилитой типа Partition Magic.

При удалённом доступе к vmg.pp.ua следует знать, что лабы развёрнуты на GNS3 на Ubuntu под виртуальной машиной Hyper-V в составе Windows server 2008 R2 на компьютере vmg.pp.ua. На компьютерепо адресу 192.168.14.56 Ubuntu установлен непосредственно.

Для удалённого доступа к десктопу в Ubuntu используется NoMachine NX Server.

Сетевой доступ к Ubuntu из Интернет организован через VPN с помощью Openvpn. Поэтому для удалённого доступа из дома следует выкачать с сайта производителя Openvpn и NoMachine NX Client for Windows и установить их на домашнем компьютере. Подробности подключения по Openvpn и настройки NX Client изложены ниже.

Openvpn не применяется для удалённого доступа по адресам 192.168.14.56 и 192.168.10.254 с кафедральных компьютеров или из ноутбука через WiFi.

Находясь на кафедре, используется преимущественно локальный адрес 192.168.14.56. По окончании работы сделайте резервную копию своей сетевой топологии (см. ниже) и сохраните её с помощью FTP по адресу 192.168.10.254 другого компьютера с виртуальной лабораторией. Дома восстановитесь из резервной копии и продолжайте работать с топологией в vmg.pp.ua через Openvpn. Сделайте дома резервную копию. Находясь на кафедре, перепешите резервную копию с 192.168.10.254 на 192.168.14.56. Продолжайте работать на кафедре и.т.д. В итоге должны бвть две одинаковые работающие топологии по адресам 192.168.10.254 и 192.168.14.56.

Чтобы попасть из дому на кафедральный компьютер с адресом 192.168.14.56 нужно зайти удалённо на vmg.pp.ua и с vmg.pp.ua зайти с помощью NX-клиента по адресу 192.168.10.10. Вы попадёте на кафедральный компьютер с адресом 192.168.14.56. Естественно, что находясь в vmg.pp.ua вы можете зайти и в другую сетевую лабораторию, которая располагается там же внутри Ubuntu под Hyper-V. Для этого есть 3 адреса. См. Выше. Удалённый доступ к vmg.pp.ua ограничен.

## Установка GNS3, Qemu и Winbox под Ubuntu на компьютере дома

Взять последнюю Ubuntu с сайта ubuntu.com (32 разряда, десктоп). Если памяти Польше 4Г берите десктоп 64 разряда. Установить либо на локальный диск, либо на виртуальную машину.

Помните, что под виртуалкой вы сможете работать только с небольшими сетевыми топологиями до 5 устройств и работа будет медленной. Поэтому рекомендуется установить Ubuntu на жесткий диск. Для этого надо 2 партиции. Достаточно будет одну взять 40-80Г и вторую равной двум объёмам памяти. Партиции можно либо создать на новом диске либо подвинуть существующие партиции на имеющемся диски и создать 2 новые.

Переключите сценарий входа на классический (мышью вверху слева, ищем system и там login screen). Добавим терминал на панель (apps-accessories-terminal-правая кнопка мыши). Нажать на верхней панели правую кнопку мыши, выбрать add to panel и добавить system monitor и Windows selector .

Используйте возможности copy-paste терминала и не забывайте, что в нём можно создавать табы (file-open tab) .

Установка софта производится через sys adm synaptic package manager или из командной строки . Нужен хороший интернет (8мбит) . Установитеефайловый менеджер mc.

Проверьте поддерживает ли ваш процесор аппаратную виртуализацию

**egrep -c '(vmx|svm)' /proc/cpuinfo**

Если вы увидите число 0 и болеее, то установите kvm (kernrl virtual machine-виртуальную машину на уровне ядра)

**sudo apt-get install qemu-kvm.**

Теперь Qemu будет выполняться в KVM, что увеличит быстродействие.

В последующем, при желании установите последнюю версию главного модуля kvm-kmod для KVM с сайта www.linux-kvm.org. Для этого распакуйте архив и в нём выполните

**./configure**
**make**
**sudo make install**

Проверьте установку командой **lsmod** и **modinfo kvm**. Проверьте появление нового устройства kvm в папке /dev.

Берём с http://gns3.net/download последний GN3, например GNS3-0.7.4-src.tar.gz. Открываем графический менеджер файлов (places на верхней панели) и распаковываем (правая кнопка мыши). Набираем в терминале mc. Через mc входим в папку GNS3-0.7.4-src. Выбираем gns3 и жмём enter. Убираем панели mc (CtrlO) и видим

**~/GNS3-0.7.4-src$ ./gns3**
**Can't import Qt modules, PyQt is probably not installed ...**

Заходим в synaptic package manager, ищем Python-Qt4 и ставим его. Теперь gns3 запустится. Закроем его и поместим ссылку на панель.

Нам надо последняя версия Qemu с поддержкой туннелей UDP. На http://gns3.net/download видим, что патчи есть для 13 версии Qemu:

**Patches for Qemu 0.13.0 to support multicast and UDP tunnels**

Скачиваем патчи с gns3.net и соответствующую версию Qemu с сайта http://wiki.Qemu.org/Download

Распаковываем Qemu и патчи. Патч Qemu-0.13.0-mcast-udp.patch помещаем рядом с папкой Qemu-0.13.0. Патчим, набирая в терминале

**patch -p0 < Qemu-0.13.0-mcast-udp.patch**

Содержимое папки net (udp.c udp.h) из папки **Qemu-0.13.0.patched** помещаем в папку **net** папки **Qemu-0.13.0**. Входим в папку **Qemu-0.13.0** (смена папки cd ) и, желая включить kvm, запускаем

**./configure –enable-kvm --target-list=i386-softmmu**

Видим

**Error: zlib check failed**
**Make sure to have the zlib libs and headers installed.**

Заходим в synaptic package manager ищем zlib1g-dev и ставим. Снова запускаем

**./configure --enable-kvm --target-list=i386-softmmu**

Видим

**...**
**KVM support yes**
**...**

Строим

**make**

Устанавливаем

**sudo make install**

Проверяем установку, запуская qemu

**Qemu**

Видим

**VNC server running on `127.0.0.1:5900'**

Виртуальная машина Qemu общается с внешним миром через протокол **VNC-** Virtual Network Computing. Запускаем VNC –клиент **(**applications-internet-remote desktop viewer-connect-protocol VNC-host 127.0.0.1:5900-connect). Видим, что нет загрузочного диска (boot disk) опрерационной системы. Его мы ещё не сделали.

## Установка tap-интерфейсов под Ubuntu на компьютере дома

Для связи маршрутизаторов Mikrotik, работающих под Qemu с внешним миром необходимо установить в Ubuntu tap-интерфейсы. tap-интерфейсы есть в Openvpn . Установим Openvpn и bridge-utils . Затем меняем /etc/rc.local

**sudo gedit /etc/rc.local**

```
#!/bin/sh -e
ifs="0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19"
for i in $ifs; do
Openvpn --mktun --dev tap$i
brctl addbr br$i
brctl addif br$s$i tap$i
ifconfig tap$i up
ifconfig br$i 10.0.$i.2 netmask 255.255.255.0 up
done
exit 0
```

Выполняем

**sudo /etc/rc.local**

Проверяем

**ifconfig|more**

Создались интерфейсы tap* в мостах br* с адресами 10.0.*.2 здесь * это 0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 ..

Наберите в консоли ubuntu

**brctl show**

**brctl show |grep br0**

Видим, что tap-интерфейсы ubuntu `tap0` лежат в сетевом мосту br0. Наберите в консоли ubuntu

**ifconfig br0**

Видим, что они имеют адреса 10.0.0.2.

Обеспечим проборску IP-пакетов, то есть превратим Ubuntu в маршрутизатор

**sudo sysctl -w net.ipv4.ip_forward=1**

Пойдёт без перегрузки. Или раскоментируйте строку

**#net.ipv4.ip_forward=1**

в файле /etc/sysctl.conf и перегрузитесь

Ставим wine1.3 для запуска исполняемых файлов Windows в Linux. Скачиваем с mikrotik.com Winbox.exe, делаем на него launcher и запускаем. Это очень удобное средство для настройки routeros mikrotik . В отличие от варианта с Windows работает copy paste для Winbox .

## Работа в Windows

Можно работать и в Windows, выкачав инсталляцию GNS3 с сайта gns3.net. Но работа будет нестабильной и крайне медленной. Не работает ускоритель KVM. На многоядерных процессорах используется одно ядро. Число допустимых интерфейсов у маршрутизатора меньше, чем в Ubuntu - только 4.

## Удалённая работа из дому для Windows и Ubuntu

Следует проверить доступность адресов (через cmd.exe)

**ping vmg.pp.ua**
**ping 212.3.125.93**

Если **vmg.pp.ua** виден в интернет, а пинги не идут, то обратитесь к Интернет-повайдеру. Проверим наличие службы Openvpn на удалённом хосте
**telnet 212.3.125.93 8002**

Пример отсутствия службы
C:\Users\Administrator>telnet vmg.pp.ua 8002
Connecting To vmg.pp.ua...Could not open connection to the host, on port 8002: Connect failed

При наличии службы вы увидите чистый чёрный экран. При неудаче - проверьте сетевые настройки и правила фаервола.

Загрузите Openvpn из Интеренета. Установите его как администратор. Отправляем мне письмо с темой **Запрос сертификата от Фамилии.** В ответном письме получаете в архиве конфигурацию

client.ovpn
корневой сертификат ca.crt
свой сертификат studentN.crt
и ключ к нему studentN.key
Здесь N-ваш номер

Эти 4 файла кладём в папку Openvpn\config и меняем в конфигурации client.ovpn 2 сроки
cert ***.crt
key ***.key
на
cert studentN.crt
key studentN.key

Если вы до этого установили в Openvpn tap-интерфейсы для работы с GNS3, то следует явно в файле client.ovpn указать в строке **dev-node** какой tap-интерфейс вы используете для Openvpn. Пример файла конфигурации client.ovpn

**client**
**dev tun**
**dev-node tap**
**proto tcp**

**remote 212.3.125.93 8002**
**resolv-retry infinite**
**nobind**
**persist-key**
**persist-tun**
**ca ca.crt**
**cert vmg.crt**
**key vmg.key**
**ns-cert-type server**
**comp-lzo**
**log openvpn.log**
**status openvpn-status.log**
**verb 4**

Помещаем client.ovpn в папку Openvpn\config. Нажимаем правой кнопкой на client.ovpn и выбираем пункт “запустить с помощью Openvpn”. Если всё сделали правильно, то увидите в чёрном консольном окне что-то типа

…
Wed May 04 16:37:53 2011 TLS: Initial packet from 212.3.125.93:8002, sid=c727c0b
a 4215f15a
Wed May 04 16:38:06 2011 VERIFY OK: depth=1, /C=UA/ST=UA/L=Dnepropetrovsk/O=DNU/
OU=FFECS/CN=lib/name=keyName/emailAddress=mail@host.domain
Wed May 04 16:38:06 2011 VERIFY OK: nsCertType=SERVER
Wed May 04 16:38:06 2011 **VERIFY OK:** depth=0, /C=UA/ST=UA/L=Dnepropetrovsk/O=DNU/
OU=FFECS/CN=lib/name=keyName/emailAddress=**mail@host.domain**
…
ifconfig **192.168.3.2** 255.255.255.0'
…
Wed May 04 16:38:17 2011 **Initialization Sequence Completed**

Ваш адрес от Openvpn **192.168.3.2.**

Другой конец Openvpn-соединения находится в Windows server 2008 R2 и имеет адрес 192.168.3.1. Проверим

**ping 192.168.3.1**

Ubuntu запущена внутри Hyper-V и имеет адрес **192.168.0.1.**

Проверим доступность удалённой системы Ubuntu

**ping 192.168.0.1**

Если пинги не идут, проверим маршруты

Route print

Должна быть строка

192.168.0.1 255.255.255.0 … 192.168.3.1

Этот маршрут даёт вам Openvpn сервер. Если её нет, то добавьте маршрут

Route add 192.168.0.1 mask 255.255.255.0 192.168.3.1

Делаем службу. Запускаем cmd.exe. С помощью команды смены папки cd идём в папку Openvpn\bin и запускаем команду **runas /user:администратор "Openvpnserv -install"**

Вводим пароль. Открываем администрирование-службы, запускаем службу openvp в режиме автозапуска. Теперь при включении компьютера вы автоматически соединитесь с Ubuntu.

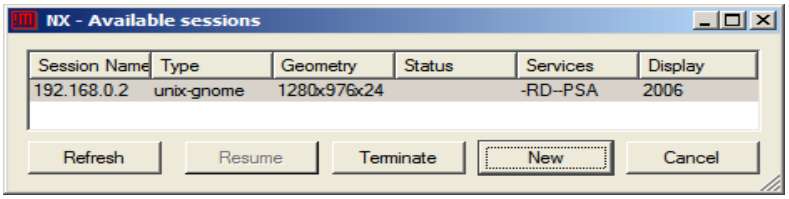
Ubuntu доступна по FTP из командной строки или через броузер по адресу ftp://192.168.0.1. Это относится и к компьютеру 192.168.14.56. Передавать файлы следует из командной стоки FTP с помощью команды put или mput.

Если у вас есть ноутбук и вы находитесь на кафедре, то назначьте на WiFi-адаптер адрес из сети 192. 168.14.0/24, шлюз 192.168.14.1 и можно подключится с помощью NoMachine через адрес 192.168 .14.56 или через адрес 192.168.10.254 виртуальной машины Ubuntu в vmg.pp.ua.

Для входа в удалённый Ubuntu через графический десктоп установим (от имени администратора) X-сервер nomachine nxclient-3.4.0-10.exe из архива http://vmg.pp.ua:8001/l/upload/nomachine.zip. Возможно понадобятся шрифты. Запустим мастер подключения nxclient в котором в поле session введём произвольное имя сессии, в поле host введём адрес 192.168.0.1, в следующем поле можно установить ADSL, в следующем окне установим менеджер окон GNOME и создадим ярлык. Запустим сессию с помощью ярлыка NX client for windows. Введём имя и пароль, например для номера D=1 login: student1 и password: student1. Нажимаем Login. Ждём соединения с Ubuntu. Последовательно в появившемся окошке видим сообщения:

Setup the environment -установка окружения
Connecting to 192.168.0.2 - соединение с Ubuntu
Connected to 192.168.0.2 - соединились с Ubuntu
Waiting authentication - ждём проверку подлинности
Authentication complete - проверку прошли
Или

Authentication failed for user student1-это если вы неправильно ввели имя или пароль. Если вы в прошлый раз некорректно завершили работу и не вышли из Ubuntu , то появится окно, извещающее о незакрытых сессиях




Если недоступна кнопка Resume, то нажимайте Terminate и затем New.
Иначе вы увидите сообщения
Download session information - загрузка информации о сессии
Negotiation the link parameters - соглашение о параметрах связи
Esteblish a display connection - установка соединения к дисплею
Появится окно во весь экран с большим изображения символа Nomachine **!m**

Возможно придётся ждать несколько минут и вы в Ubuntu.

Можно подключаться к vmg.pp.ua и из домашнего Ubuntu или другого Linux. Для подключения по Openvpn, полученные файлы следует положит в папку /etc/Openvpn . Файл client.ovpn переименуйте в client.conf и закомментируйте строку

**dev-node tap**

Старт-стоп Openvpn командой

**/etc/init.d/openvpn start|stop**

Далее скачать nxclient из интернета и установить его

**dpkg -i файл.deb**

Или воспользуйтесь в Ubuntu штатной програмой qtnx. Заметим, что qtnx даже при сжатии jpeg, медленне прорисовывает экран, чем nxclient.

Если вы просто хотите получить доступ только к командной строке, воспользуйтесь безопасным шелом ssh, например

**ssh studentD@192.168.0.1**

Для Windows есть уникальная утилита putty, выполняющая и функции Ssh-клиента. Putty входит в состав Gns3. Укажите в putty имя пользователя, адрес 192.168.0.1 и протокол ssh.

Войдя в Ubuntu дома, через Openvpn, по адресу 192.168.10.254 или 192.168.14.56, выполните в теминале команду top, показывающую загрузку компьютера. Не стартуйте топологию, если компьютер занят более чем на 50% или у него свободной памяти меньше,чем 500Мб. Если компьютер не ваш, то утилита top покажет, кто занимает компьютер. Свяжитесь с ним.

Для завершения своей удалённой работы в Ubuntu никогда не закрывайте в Windows окно NX client for windows. Делайте в Ubuntu System-Logout. Будут проблемы.

Эта работа считается сданной, если

Студент получит сертификат

Студент поместит по FTP на свой рабочий стол на компьютеры vmg.pp.ua и 192.168.14.56 свою фотографию с подписью в виде своей фамилии.

# 2. Работа с Mikrotik RouterOS в Qemu и GNS3

**Начальная подготовка Ubuntu.**
**Установка и работа с операционной системы RouterOS.**
**Установка и настройка GNS3**
**Работа с RouterOS из коммандной строки**
**Импорт и экспорт топологий**
**Доступ к удалённому маршрутизатору из домашнено компьютера**

## Начальная подготовка Ubuntu.

Добавим терминал на панель (Applications_>Accessories->terminal -правая кнопка мыши). Нажать на верхней панели правую кнопку мыши, выбрать add to panel и добавить system monitor и Windows selector … menu .

В период загрузки сетевой топологии маршрутизаторы сильно грузят процессоры. Загружать топологию может любой студент. Стартовать свою топологию нет смысла, пока загружены процессоры. Поэтому за правило надо взять наблюдение за окном системного монитора. С его помощью можно узнать, когда загрузка маршрутизаторов у вас или других студентов завершилась. За загрузкой системы можно наблюдать и с помощью консольной команды top.

При работе вы можете открыть массу окон, чтобы в них не запутаться используйте закреплённый на панели Windows selector.

При желании для добавления русского system->preferences->keyboard->layouts->add byLanguage russian russia. Должны увидеть русские буквы на изображении клавиатуры, если нет доставьте шрифты Add. Далее нажимаем Options и выбираес клавиши для переключения раскладки.(!! Один выбор там уже сделан). Close. Applay system wide. Close. Если при следующем входе русский пропадет выполните в в терминале команду setxkbmap

Мы часто используем терминал (консоль). Не запускайте много экземпляров. Используйте табы в терминале (file-open tab) для открытия в одном окне множества консолей.

В качестве менеджера файлов мы используем mc. В нём для поиска используйте F9-c-f. Для скрытия-показа панелей ctrlO.

## Установка и работа с операционной системы RouterOS.

**1.** Используем последнюю версию образа RouterOS mikrotik для интел-платформы. В папке /home/4all возьмём в свою папку CD-образ операционной системы, например mikrotik-5.2.iso. Создадим пустой образ виртуального диска формата qcow2 размером 111mb:

**Qemu -img create -f qcow2 mikrotik-5.2.img 111M**

В виртуальной машине Qemu из образа CD mikrotik-5.2.iso установим RouterOS на образ виртуального диска mikrotik-5.2:

**qemu mikrotik-5.2.img -cdrom mikrotik-5.2.iso -boot d**

видим

[1] 11424
VNC server running on `127.0.0.1:5900'

Нам нужен VNC-клиент для общения с виртуальной машиной. Выбираем applications - internet - remote desktop viewer. Далее connect - protocol VNC -host 127.0.0.1:5900 и нажимаем connect. Видим начальное окно установки RouterOS. Выбираем

модули для установки system, ppp, dhcp, advanced-tools, ipv6, mpls, ntp, routerboard, routing, security, synchronous, user-manager. Нажимая i, начинаем установку. На вопросы отвечаем по умолчанию. После завершения установки закрываем qemu (и RouterOS) , нажав CtrlC (в терминале, а не в VNC-клиенте).

Проверяем установку:

**qemu mikrotik-5.2.img**

Запускаем VNC –клиент. Вводим Login: admin, Password: оставляем пустым. Для остановки RouterOS используйте команду RouterOS system sghutsown. При обилии маршрутизаторов ввод этой команды для каждого маршрутизатора утомляет. Поэтому иногда будем практиковать неправильную остановку RouterOS, нажав CtrlC.

Важно помнить, что Qemu в VNC –клиенте захватывает мышь - освобождение ctrlAlt. Qemu под VNC работает в двух режимах: системы и монитора. Пепеключение ctrl+alt 1 и ctrl+alt 2. Перейдя в монитор посмотрите версию Qemu командой info version. Полный список команд с описанием смотри на сайте Qemu .org.

Можно стартовать RouterOS и так

**qemu mikrotik-5.2.img&**

**Жмём комбинацию клавишCtrlZ**

Знак & в конце команды важен: Qemu не захватит терминал.

Для неправильной остановки RouterOS нам надо убить процесс **Qemu** . Находим его номер

**ps ax|grep qemu**

видим

11989 pts/3 S 0:11 qemu -hda mikrotik-5.2.img -cdrom mikrotik-5.2.iso -boot d

12020 pts/3 S+ 0:00 grep --color=auto qemu

Убиваем процесс

**kill 11989**

или

**kill -KILL 11989**

проверяем

**ps ax|grep qemu**

видим

12020 pts/3 S+ 0:00 grep --color=auto qemu

Убить все процессы qemu можно так

**killall qemu**

Всегда проверяйте наличие нежеланных версий процесса Qemu командой **ps ax|grep qemu.** Набирая

**ps ax|grep qemu|grep student**

вы узнаете, кто сейчас и с каким номером D вам мешает. Чужой процесс вы не убъёте - звоните студенту D, чтобы он убил неправильные qemu или qemuwrapper и правильно назначил порты или обратитесь к администратору.

Узнать, кто сейчас занимает процессорное время можно с помощью консольной утилиты top.

**2.** В VNC –клиенте не работает copy-paste в окне Qemu . Добъёмся этого другими средствами. Реальные железные маршрутизаторы не имеют клавиатуры и монитора и настройка поизводится через через консоль через последовательный порт с помощью

терминальных программ типа HyperTerminsl в XP. Qemu позволяет перенаправить консоль. Наберём в терминале команду

**qemu mikrotik-5.2.img -serial telnet:127.0.0.1:3000,server,nowait**

В другом табе окна терминала введём

**telnet 127.0.0.1 3000**

Мы в консоли RouterOS. Copy-paste работает. Проверьте. Заметим, что Copy-paste только для мыши. Клавиатура не работает.

Сделайте копию R0.img для **mikrotik-5.2.img**.

**cp mikrotik-5.0rc8.img R0.img**

и пока работайте с копией. Чистая версия нам понадобится для GNS3.

**3.** RouterOS внутри Qemu можно связать с внешним миром многими способами (см. документацию на сайте wiki.Qemu.org). Мы для связи с хост машиной (Ubuntu) используем tap-интерфейсы, а для связи между RouterOS **—** UDP.

Для каждого студента D в Ubuntu уже созданы tap-интерфейсы tapD00 tapD01 ... tapD07 tapD10 ... tapD19. Интерфейсов tapD08 и tapD09 нет по техническим причинам. Tap-интерфейсы помещены в мосты BD00 BD01 ... BD07 BD10 ... BD19. На мосты назначены адреса 10.D.0.2, 10.D.1.2 … 10.D.7.2, 10.D.10.2 … 10.D.19.2. Маска /24.

Соединим маршрутизатор **R0** с внешним миром для студента 0. (Вы работаете со своим номером). В дальнейшем старайтесь придерживаться соглашения - номер в имени маршрутизатора равен номеру tap-интерфейса.

**qemu R0.img -serial telnet:127.0.0.1:3000,server,nowait -net nic –net **
**tap,script=no,downscript=no,ifname=tap000**
**VNC server running on `127.0.0.1:5900'**

\ - перенос команды на другую строку. Использовать по необходимости. Запускаем в другом табе окна терминала

**ip link show tap000**

3: tap0: <**BROADCAST**,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 100
link/ether b2:cf:b1:1e:89:43 brd ff:ff:ff:ff:ff:ff

Для сравнения звапустим

**ip link show tap001**

5: tap1: <**NO-CARRIER**,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 100 .

Tap-интерфейс **tap000** Ubuntu перешёл в активную стадию, а **tap001**-нет. Запускаем VNC –клиент к 127.0.0.1:5900. Переключаемся в монитора ctrl+alt 2. Вводим

**info network**

и видим что Qemu увидела для Routeros tap-интерфейс как эсернет-карточку модели e1000 с мак-адресом 525400123456.

В третьем табе окна терминала введём

**telnet 127.0.0.1 3000**

Мы в консоли RouterOS. Смотрим интерфейсы

[admin@MikroTik] > **interface ethernet pr**
Flags: X - disabled, R - running, S - slave
# NAME MTU MAC-ADDRESS ARP
0 R ether1 1500 52:54:00:12:34:56 enabled

Видим, что наша эсернет-карточку модели e1000 с мак-адресом 525400123456 называется в RouterOS как ether1. Со стороны Ubuntu tap-интерфейс для студента 0 имеет адрес 10.0.0.2/24. Поэтому в RouterOS назначим другой адрес из сети 10.0.0.0/24 например

[admin@MikroTik] > **ip address add address=10.0.0.1/24 interface=ether1**

[admin@MikroTik] > **ping 10.0.0.2**

```
HOST                    SIZE  TTL TIME  STATUS
10.0.0.2                56    64  10ms
10.0.0.2                56    64  1ms
  sent=2 received=2 packet-loss=0% min-rtt=1ms avg-rtt=5ms max-rtt=10ms
```

В четвёртом табе окна терминала введём

**ping 10.0.0.1**

**telnet 10.0.0.1**

Мы опять в консоли RouterOS.

Берём себе из папки /home/4all утилиту winbox.exe. Делаем для неё ярлык для запуска. (desktop - create launcher) Помещаем его в верхню панель. Запускаем. Вводим адрес 10.0.0.1 и попадаем в маршрутизатор. Смените имя маршрутизатора (system identity). Остановим маршрутизатор из winbox или командной строкой sys shut.

**4.** Сделайте ещё одну копию R1.img образа диска маршрутизатора mikrotik-5.2.img. Соединим два маршрутизатора, используя UDP.

**qemu R0.img -net nic -net udp,sport=33333,dport=22222,daddr=127.0.0.1&**

```
Warning: vlan 0 is not connected to host network
VNC server running on `127.0.0.1:5900'
```

**qemu R1.img -net nic -net udp,sport=22222,dport=33333,daddr=127.0.0.1&**

```
Warning: vlan 0 is not connected to host network
VNC server running on `127.0.0.1:5901'
```

Запускаем VNC –клиент и подключаемся к 127.0.0.1:5900 и 127.0.0.1:5901. Пепеключаемся в монитор ctrl+alt 2. Вводим

**info network**

и видим, для R0


для R1


Т.е. эсернет-адаптерам поставлены в соответствие UDP-канал. Возвращаемся в RouterOS (ctrl+alt 2) и назначаем адреса для R0

**ip address add address=1.1.1.1/24 interface=ether1**

для R1

**ip address add address=1.1.1.2/24 interface=ether1**

Пингуем из R1 R0. Связь есть.

Все пакеты из эсернет интерфейса R0 поступают на UDP порт 22222 хост-машины ubuntu и есернет интерфейса R0 принимает все пакеты из UDP порта 33333 хост-машины. Для R1 наоборот. Все пакеты из эсернет интерфейса R1 поступают на UDP порт 33333 хост-машины ubuntu и есернет интерфейса R1 принимает все пакеты из UDP порта 22222 хост-машины. Таким образом сетевой ethernet кабель моделируется двунаправленным UDP

каналом с использованием двух портов.

Попытаться из командной строки организовать сложную топрологию – дело громоздкое и чревато ошибками. Тут на помощь приходит контейнер виртуальных машин GNS3

## Установка и настройка GNS3

GNS надо только для быстрого сбора топологий. Он генерирует командную строку для Qemu , которую можно увидеть, запустив перед запуском GNS программу Qemuwraper.

Скачиваем GNS с сайта gns3.net или берём архив из папки /home/4all. Распаковываем архив в свою папку. Запускаем mc. Находим в папке GNS3 файл gns3 и запускаем его (жмём на нём Enter). Появится графическая оболочка GNS3. Если не появилась, то убираем панели mc (CtrlO) и смотрим ошибки. Если это не помогло, то перед запуском gns3 стартуйте до старта gns3 в консоли програму
**Qemuwrapper/Qemuwrapper/py -p порт,**
где порт тот же, что и в настройках GNS для Qemuwrapper. Игнорируйте ошибку. Попытайтесь устранить ошибку, анализируя консоль Qemuwrapper. На начальном этапе Qemuwrapper может помочь диагностировать ошибки почему не запускается тот или иной маршрутизатор. Держите консоль Qemuwrapper открытой.

Создадим ярлык для запуска GNS3. Создадим в **Вашей Папке (**studentD) папки **mkdir tmp**

**mkdir projects**

Запустим GNS3. Закроем окошко создания проекта. Выберем для настройки edit **—** preferences. Устанавливаем путь к своей папке проектов projects и путь к папке, где лежит образ диска **mikrotik-5.2.img**(image directory).

Настроим терминал: Qemu edit - preferences - General - Terminal setting. Возьмём lдля Preconfigurated terminal commands — Konsole (Linux KDE), для Terminal command - gnome-terminal -t %d -e 'telnet %h %p' >/dev/null 2>&1 & и жмём OK.

Выберем для настройки Qemu edit - preferences - Qemu - general setting. Устанавливаем путь к своему файлу Qemuwraper.py и путь к своей рабочей папке tmp.

Важно, чтобы у всех одновременно работающих **не было общих окрытых портов**. Возьмём порты

10525+D - Qemuwraper

20000+D*1000 -UDP

3000+D*100 - консоль

D - ваш номер. Например, если вы student7, то вписываем порты

10532 - Qemu wraper

27000 -UDP

3700 – консоль

Удалите внешний Qemuwraper или дайте ему тот же порт.

Хотя заметим, что порт консоли поменять не удаётся и его придётся менять постоянно

для каждого маршрутизатора. Нажимаем применить. Должен пройти Test. Жмём OK.

**Периодически проверяйте целостность GNS, нажимая Test. При проблемах- распакуйте архив и настройте снова.**

Занятые порты TCP и соответствующие программы можно посмотреть командой

**netsat -a -t -n -p**

Занятые порты UDP и соответствующие программы можно посмотреть командой

**netsat -a -u -n –p**

Определите, кто забрал ваш порт и обратитесь к администратору.

Перейдём edit - preferences - Qemu - Qemu host и добавим в **binary image** путь к образу созданного маршрутизатора **mikrotik-5.2.img.** Галочки kQemu , kvm не ставим. Галочку kvm нужно поставить у себя дома, если вы работаете не под виртуалкой и у вас установлена поддержка kvm. Даём имя. Save. Apply. ok.

Изменим иконку Qemu host на левой панели GNS. Выберем Edit-Symbol Manager. Выбираем слева иконку (например router) и нажимая > переносим её в правую часть. Дважды щёлкаем на нём в правой части. Появится
имя: router
тип: декоративный узел
Меняем
имя: **Mikrotik**
тип:выбираем из списка Qemu host

Нажимаем применить. Имя в правой части изменится на **Mikrotik**. Нажимаем OK. Слева в GNS в типах узлов появится новая иконка **Mikrotik** - синоним Qemu host.

Перед запуском топологии запустим Qemuwrapper. Когда вы наберётесь опыта, то это можно не делать.

Создадим первую топологию first. Запускаем GNS. В появившемся окошке вводим имя нового проекта first и ставим галочку Save nvrams … Ok. Если проект уже создан, то выбираем Open. Для создания проекта можно открыть и пункт меню File-NewBlankProject. Переташим два устройства Qemu host с левой панели на центральную и меняем у них порт консоли согласно своему номеру и имена на R0 и R1 (правая кнопка-change console port и change the hostname). Соединим интерфейс e0 маршрутизатора R0 с интерфейсом e1 маршрутизатора R1 (значок «add a link» на на верхней панели, пункт manual). После завершения соединения снова нажмите значок «add a link». Порт консоли маршрутизатора можно узнать наведя на него в GNS мышь, а сменить — правой кнопкой (до запуска). Обязательно смените порты консоли.

Заметим, о соотношении названий интерфейсов в GNS и RouterOS

| GNS | RouterOS |
|---|---|
| e0 | ether1 |
| e1 | ether2 |
| ... | |

По умолчанию GNS3 назначает маршрутизатору шесть сетевых интерфейсов. Менять не будем. Добавляем в маршрутизаторы tap-интерфейсы. Пусть у вас номер 7. Начнём.

Правая кнопка на R0, configure, R0, Qemu options

**-net nic,vlan=6 -net tap,script=no,downscript=no,vlan=6,ifname=tap700**

Ok

Правая кнопка на R1, configure, R1, Qemu options

**-net nic,vlan=6 -net tap,script=no,downscript=no,vlan=6,ifname=tap701**

Ok

Это приводит к созданию внутри маршрутизатора сетевой карточки ether7 которая связывается с сетевым tap-интерфейсом Ubuntu с именем tap700 (tap701). Tap-интерфейса в GNS не видно.

Сохраните проект. Улучшив момент, когда система не занята (system monitor), стартуйте все маршрутизаторы (зелёный треугольник вверху). Старт-стоп отдельного маршрутизатора производится из контекстного меню. Не используйте рестарт – не всегда работает. Дождитесь окончания загрузки.

Откройте в папке проектов папку проекта first. Откройте двойным щелчком (программой gedit) файл topology.net. Изучите его. Поняв его устройство можно менять топологию без GNS. Перед запуском топологии, сделанной на другой машине обязательно проверьте и исправьте этот файл
autostart = False

[Qemu 127.0.0.1:10537] **-согласно номера студента 7**

workingdir = working **-папка-обязательная строка**

udp = 30700 **-согласно номера студента 7**

[[QemuDevice]]

image = /home/student7/mikrotik-5.5.img **-измените при переносе с другой машины**

netcard = e1000

kvm = True **- эта строка есть, если только работаете на чистом железе с установленным kvm**

[[QEMU R0]]

console=3700**-согласно номера студента 7**

e0 = R1 e1 **-связь**

[[QEMU R1]]

console=3701**-согласно номера студента 7**

e1 = R0 e0 **-связь**

[GNS3-DATA]

workdir = working **-папка-обязательная строка**

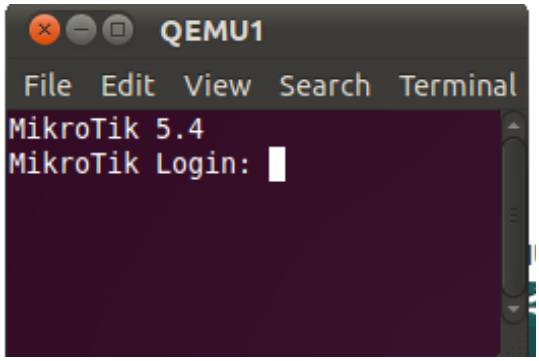
В папке first вы увидите папку working, а в ней папки R0,R1, а в этих папках два образа диска FLASH и SWAP. Если это не так, то остановитесь и проверьте настройки,так как

в дальнейшем возникнут трудности.

Осуществлять конфигурацию маршрутизатора через консоль с помощью интерфейса командной строки можно тремя способами: либо через VNC-клиент, либо через консоль маршрутизатора в GNS либо через telnet. В первом случае надо знать порт VNC-сервера. Порт VNC можно узнать в консоли Qemuwrapper, если запускать маршрутизаторы по одому. CopyPaste не поддерживается Во втором случае консоль маршрутизатора в GNS вызывается двойным щелчком мыши. Во третьем случае клиент telnet вызывается через консоль Ubuntu telnet 127.0.0.1 порт-консоли. Второй случай эквивалентен третьему и также приводит к вызову команды telnet 127.0.0.1 порт-консоли.

В третьем случае работать удобнее всего так как работает copy-paste. Если консоль маршрутизатора не запустилась, то анализируйте вывод в консоли Qemuwrapper и занятые порты (netstat). Если там всё нормально, то останавливаем маршрутизатор, убиваем папку маршрутизатора и снова стартуем маршрутизатор.

Стартуем консоли маршрутизаторов, два раза щёлкнув на их иконках в GNS3. Видим




Назначьте имена

**system identity set name=R0**

**system identity set name=R1**

или коротко

sy i s n=R0

sy i s n=R1

Используйте CopyPaste.

Каждый маршрутизатор шлёт соседям широковещательную информацию о себе и обрабатывает такую же информацию, принятую от сосоедей. Результат можно посмотреть командой **/ip neighbor discovery print и /ip neighbor discovery print detail.** Мы увидим соседей, достижимых по ethernet-прооколу в том числе через мосты.

[admin@R0] > **ip neighbor print**

| # | INTERFACE | ADDRESS | MAC-ADDRESS | IDENTITY | VERSION | BOARD |
|---|---|---|---|---|---|---|
| 0 | ether1 | | **00:AA:00:28:C5:01** | **R1** | 5.2 | x86 |

[admin@R0] > **ip neighbor print detail**

0 interface=**ether1** mac-address=**00:AA:00:28:C5:01** identity="**R1**"
platform="MikroTik" version="5.2" unpack=none age=17s uptime=10m5s
software-id="1ZGH-GE5Q" board="x86" ipv6=yes interface-name="**ether2**"

R0 через interface=**ether1** обнаружил соседа identity="**R1**". Сосед R1 послал эту информацию через свой интерфейс interface-name="**ether2**" у которого mac-address=**00:AA:00:28:C5:01**

Проверим MAC-адрес

[admin@R1] > **interface ethernet print where name=ether2**

| # | NAME | MTU | MAC-ADDRESS | ARP |
|---|---|---|---|---|
| 1 R | ether2 | **1500** | **00:AA:00:28:C5:01** | enabled |

[admin@R1] > **ip neighbor pr int**

| # | INTERFACE | ADDRESS | MAC-ADDRESS | IDENTITY | VERSION | BOARD |
|---|---|---|---|---|---|---|
| 0 | ether2 | | 00:AA:00:F4:68:00 | **R0** | 5.2 | x86 |

[admin@R1] > **ip neighbor print detail**

0 interface=ether2 mac-address=**00:AA:00:F4:68:00** identity="R0"
platform="MikroTik" version="5.2" unpack=none age=1m uptime=10m6s
software-id="1ZGH-GE5Q" board="x86" ipv6=yes interface-name="**ether1**"

R1 через interface=ether2 обнаружил соседа identity="R0". Сосед послал эту информацию через свой интерфейс interface-name="**ether1**" у которого mac-address= **00:AA:00:F4:68:00**

Посмотрим MAC-адрес

[admin@R0] > **interface ethernet print where name=ether1**

| # | NAME | MTU | MAC-ADDRESS | ARP |
|---|---|---|---|---|
| 0 R | ether1 | **1500** | **00:AA:00:F4:68:00** | enabled |

Возьмите за привычку давать маршрутизворам имена и набирать эту команду обнаружения соседей. Если вдруг маршрутизатор не видят соседи, то сохраните его конфигурацию, остановите его, убейте его папку и стартуйте заново.

Так как ethet7 сам подсоединён к бриджу Ubuntu, то для чистоты следует использовать фильтр
**ip neighbor print** where ((interface!=ether7)&&( interface-name!=ether7))

**ip neighbor print detail** where (interface!=ether7)&&( interface-name!=ether7)

Назначаем адреса и проверяем

На R0

**ip ad ad**

**address:10.7.0.1/24**

**interface: ether7**

**ping  10.7.0.2**

**ip ad ad**

**address:1.1.1.1/24**

**interface: ether1**

На R1

**ip ad ad**

**address:10.7.1.1/24**

**interface: ether7**

**ping  10.7.1.2**

**ip ad ad**

**address:1.1.1.2/24**

**interface: ether1**

Должен пойти

[admin@R1] >**ping  1.1.1.1**

Наличие tap-интерфейсов с настроенными внутри RoutrOS адресами даёт четвёртый способ доступа к консоли RoutrOS из Ubuntu

**telnet 10.7.0.2**

**telnet 10.7.1.2**

Запускаем winbox. Вводим адрес 10.7.0.2, имя admin. Сохраняемся и соединяемся. Дедаем то же  для адреса 10.7.0.2**.**  Совершите обзор возможностей маршрутизатора. Они поражают, если посмотреть на размер виртуального диска

**ls –l r*img**

Более того маршрутизатор имеет веб интерфейс, например можно ввести в firefox http://10.7.0.1 и маршрутизатор который и доступен через ftp 10.7.0.1.

В winbox есть пункт меню NewTermial, что даёт пятый способ доступа к консоли RoutrOS из Ubuntu.

Помните, что если в GNS3 назначить маршрутизатору один сетевой интерфейс, то tap-интерфейс в RouterOS будет называться ether2 и  настройки адресов для tap интерфейса

ether7 будут относится к несуществующему интерфейсу. О том как это поправить узнаете ниже.

Смотрим в окно консоли предварительно запущенного Qemuwrapper. Берём содержимое консоли в карман и отформатируем согласно правилам записи командной строки Qemu http://wiki.Qemu .org/Manual. MAC-адреса мы не указываем и уберём несоединённые интерфейсы. Получим

**qemu -name R0 -m 256 /home/student7/projects/first/working/R0/FLASH -hdb /home/student7/projects/first/working/R0/SWAP -net nic,vlan=0 –net udp,vlan=0,sport=27002,dport=27003,daddr=127.0.0.1 -serial telnet:127.0.0.1:3700,server,nowait**
**-net nic,vlan=6 -net nic -net tap,script=no,downscript=no,vlan=6,ifname=tap700**

**qemu -name R1 -m 256 /home/tudent7/projects/first/working/R1/FLASH -hdb /home/p/.gns3/first/working/R1/SWAP -net nic,vlan=1 -net udp,vlan=1,sport=27003,dport=27002,daddr=127.0.0.1 -serial telnet:127.0.0.1:3701,server,nowait -net nic,vlan=6 -net nic -net tap,script=no,downscript=no,vlan=6,ifname=tap701**

Заметим, что здесь мы не увидели образ диска RouterOS mikrotik-5.2.img. GNS делает для него копию FLASH для каждого маршрутизатора в топологии. SWAP – это образ диска для свопинга.

Если ввести эти команды в консоли ubuntu, то получим тот же эффект, что и в gns. Делать это не обязательно. Если решитесь, то лучше для этого создайте командный файл.

Строки

**-serial telnet:127.0.0.1:3700,server,nowait**

**-serial telnet:127.0.0.1:3701,server,nowait**

отвечают за консоль

Что отвечает за соединение между собой интерфейсов маршрутизаторов R0 и R1? Об этом уже говорилось. Для R0 строка

**-net nic,vlan=0 -net udp,vlan=0,sport=27002,dport=27003,daddr=127.0.0.1**

говорит, что все пакеты из e0 поступают на UDP порт 27002 хост-машины ubuntu и e0 принимает все пакеты из UDP порта 27002 хост-машины.

Для R1 наоборот. Строка

**-net nic,vlan=1 -net udp,vlan=1,sport=27003,dport=27002,daddr=127.0.0.1**

говорит, что все пакеты из e0 поступают на UDP порт 27003 хост-машины ubuntu и e0 принимает все пакеты из UDP порта 27001 хост-машины.

И, наконец, строки

**-net nic,vlan=6 -net nic -net tap,script=no,downscript=no,vlan=6,ifname=tap700**

**-net nic,vlan=6 -net nic -net tap,script=no,downscript=no,vlan=6,ifname=tap701**

приводят к созданию внутри маршрутизатора сетевой карточки ether7 которая связывается с

сетевым tap-интерфейсом с именем tap700 (tap701)

Остановите все маршрутизаторы (красный квадрат). Сохраните топологию. Вывод. Можно обойтись без GNS. Для сложных топологий легко запутаться поэтому и используют gns. Найдите, к стати, что в файле topology.net отвечает за соединение маршрутизаторов.

Из-за невозможности зафиксировать для себя порт консоли в GNS3 создайте топологию-шаблон, состоящую из несвязаных между собой маршрутизаторов R0,R1 ...R7,R10,R11… в которых настроены IP-адреса для связи с хост-машиной Ubuntu через tap-интерфейсы. Это позволит не использовать консоль в GNS3. Конфигурация маршрутизаторов будет проходить из Ubuntu через telnet или winbox путём соединения к IP-адресу tap-интерфейса маршрутизатора. Дайте маршрутизаторам имена внутри его операционной системы RouterOS . Далее при выполнении очередной лабораторной работы следует лишь создать копию папки с топологией шаблона. В дальнейшем приработе с топологией из N маршрутизаторов используйте N тапов в консоли Ubuntu и в этих тарах запускайте команды telnet 10.7.0.1, telnet 10.7.1.1.

Лицензионных суток работы RouterOS вполне достаточно для освоения любой сетевой технологии!!! (не хватит - переставьте RouterOS в Qemu , предварительно сделав бекап конфигурации )

## Работа с RouterOS из коммандной строки

Запустите топологию first. Обратимя к R0 через телнет. Всё время смотрите, через сколько закончится лицензия. Если остался час-два, то сделайте backup (об этом позже), остановите маршрутизатор, убейте его папку, заапустите его, назначьте адрес на tap интерфейс и восстановите конфигурацию из backup (об этом позже),

**telnet 10.7.0.1**

Список команд - ? Синие пункты — это меню, а остальное — команды. Перейдём в меню interface. Команда идентифицируется по первым набранным символам, если они уникальны, то текст зеленеет. Поэтому достаточно набрать in. При желани видеть всю команду нажмите табуляцию. Команда print выведет интерфейсы. Команда print detail выведет более детальную информацию. Вернёмся уровнем выше «..». Переход в главное меню /. Тоже самое можно увидет не заходя в подменю

**int pr**

**int pr det**

Зайдём в меню

**[admin@MikroTik] /ip address> pr**

**Flags: X - disabled, I - invalid, D - dynamic**

| # | ADDRESS | NETWORK | INTERFACE |
|---|---|---|---|
| 0 | 10.0.0.1/24 | 10.0.0.0 | ether7 |
| 1 | 1.1.1.1/24 | 1.1.1.0 | ether1 |

Команды работают в режиме полного ввода

**[admin@MikroTik] /ip address> add address=2.2.2.2/24 interface=ether2**

или далога

**[admin@MikroTik] /ip address> add**

**address: 3.3.3.3/24**

**interface: ether3**

Всегда помогает табуляция и знак вопроса. Установки можно поменять. Например для адреса 2.2.2.2/24 поменяем интерфейс на ether4. В начале узнаем номер

**[admin@MikroTik] /ip address> pr**

**Flags: X - disabled, I - invalid, D - dynamic**

| | # | ADDRESS | NETWORK | INTERFACE |
|---|---|---|---|---|
| **0** | | **10.0.0.1/24** | **10.0.0.0** | **ether7** |
| **1** | | **1.1.1.1/24** | **1.1.1.0** | **ether1** |
| **2** | | **2.2.2.2/24** | **2.2.2.0** | **ether2** |
| **3** | **3.3.3.3/24** | **3.3.3.0** | **ether3** | |

Это №2

Меняем

**[admin@MikroTik] /ip address> set 2 interface=ether4**

**[admin@MikroTik] /ip address> pr**

**Flags: X - disabled, I - invalid, D - dynamic**

| | # | ADDRESS | NETWORK | INTERFACE |
|---|---|---|---|---|
| **0** | | **10.0.0.1/24** | **10.0.0.0** | **ether7** |
| **1** | | **1.1.1.1/24** | **1.1.1.0** | **ether1** |
| **2** | | **2.2.2.2/24** | **2.2.2.0** | **ether4** |
| **3** | **3.3.3.3/24** | **3.3.3.0** | **ether3** | |

Убиваем созданное

**[admin@MikroTik] /ip address> rem 2,3**

**[admin@MikroTik] /ip address> pr**

**Flags: X - disabled, I - invalid, D - dynamic**

| | # | ADDRESS | NETWORK | INTERFACE |
|---|---|---|---|---|
| **0** | | **10.0.0.1/24** | **10.0.0.0** | **ether7** |
| **1** | **1.1.1.1/24** | **1.1.1.0** | **ether1** | |

Это основной принцип. Интерфейс чрезвычайно мощный. Например, фильтр

**/ip address print where interface=ether1**

Кроме того поддерживаются скрипты. Вот пример скрипта, который наряду с IP_адресами выводит и MAC-адреса

**foreach i in=[/ip address find] do={**

**:local intr [/ip address get $i interface];**

**:local addr [/ip address get $i address];**

**:local MAC  [/int eth get [/int eth  find name=$intr] mac-address] ;**

**:put ($addr." ".$intr." ".$MAC);**

**}**

При желании изучите скриптинг самостоятельно wiki.mikrotik.com

Покажем лишь как добавить и запустить скрипт

**[admin@MikroTik] > sys scr**

**[admin@MikroTik] /system script> add**

**[admin@MikroTik] /system script> pr**

**Flags: I - invalid**

**0 I name="script1" owner="admin" policy=ftp,reboot,read,write,policy,test,winbox,password,sniff,sensitive,api run-count=0 source=""**

Поместим скрипт в карман и вставим его в редакторе скриптов  RouterOS

**edit 0 source**

Сохраняемя в редакторе Ctrl o и выполняем

**[admin@MikroTik] /system script> run 0**

**10.0.0.1/24 ether7 52:54:00:12:34:5C**

**1.1.1.1/24 ether1 00:00:AB:18:47:00**

## Импорт и экспорт топологий

Рассмотрим скрипт /home/4all/makeGetBackup

```
#!/bin/bash
for ((i=$1;i<=$2;i++)) ;do
ssh admin@10.D.$i.1 "system backup save name=R$i"
sftp admin@10.D.$i.1:/R$i*
done
```

ssh (secure shell)  даёт команду маршрутизатору Mikrotik с адресом **10.0.$i.1** на сохранение конфигурации в маршрутизаторе в файл **R$i** , а sftp забирает  файл из  Mikrotik  в локальный файл н Ubuntu.

Заберите скрипт в свою папку projects, адаптируйте его под свой вариант D. Сделайте скрипт исполняемым

**chmod a+x makeGetBackup**

Если  в вашей топологии есть маршрутизаторы у которых tap-интерфейсы имеют адреса  10.D.1.1 10.D.2.1 ... 10.D.6.1 , то чтобы забрать конфигурацию в Ubuntu надо выполнить команду

**makeGetBackup 1 6.**

В локальной папке появятся файлы бекапов R1.backup , R2. backup ... R6. backup. Поместите файлы вместе с топологией в архив first и, воспользовавшись ftp, заберите его

домой.

Для восстановления конфигураций операционных систем маршрутизаторов предлагается скрипт /home/4all/Restore

```
#!/bin/bash
for ((i=$1;i<=$2;i++)) ;do
scp R$i.backup admin@10.0.$i.1:
ssh admin@10.0.$i.1 "system backup load name=R$i"
done
```

Здесь команда scp безопасно копирует по сети файл бекапа в Mikrotik. ssh удалённо выполняет команду восстановления.

Для восстановления топологии следует вначале разархивировать архив в папку, создать в папке папку working и отредактировать файл топологии. Запустить топологию. Назначить адреса tap-интерфейсам согласно принятым соглашениям. Далее выполнить скрипт

**../Restore 0 6**

Отвечайте yes. Маршрутизаторы подхватят конфигурацию и перегрузятся. В случае проблем удалите файл known_hosts в папке .ssh.

Если конфигурация RouterOS содержала сертификаты, то восстановлении сертификаты не восстанавливаются. Следует сохранить файлы сертификатов и вручную их импортировать до восстановления всей конфигурации. Далее обязательно надо проверить сетевые настройки в которых фигурируют сертификаты например протоколы SSTP и OPENVPN.

## Работа в Windows

Можно работать и в Windows, выкачав инсталяцию с gns3.net. Но работа будет нестабильной и крайне медленной. Не работает ускоритель KVM. На многоядерных процессорах используется одно ядро. Число допустимых интерфейсов у маршрутизатора меньше - только 4. Для связи с Winbox необходимо создать с помощью Openvpn (раздел утилиты) несколько сетевых интерфейсов tap и переименовываем их в tap0, tap1, tap2 ...tapi. Строка подключения маршрутизатора к интерфейсу tapi в Qemu

**-net nic,vlan=7 -net tap,vlan=7,ifname=tapi**

Далее надо угадать в маршрутизаторе какой это для него интерфейс. Если без угадать, то.

В виндовом Qemu при старте маршрутизатора стартует и VNC –клиент (что надоедает, если маршрутизаторов много). В VNC –клиенте нажимаем ctrl alt 2 и вводим

**info network**

Запоминаем мак адрес tap-интерфейса. Нажимаем ctrl alt 1 - возвращаемся в консоль маршрутизатора. Набираем в маршрутизаторе

**int eth print**

И смотрим по мак адресу, кто из эсернет интерфейсов у нас соответствует tap-интерфейс. . VNC –клиент захватывает мышь - освобождение ctrl alt

Может пропасть возможность Openvpn-соединения с vmg.pp.ua. Следует явно в файле client.ovpn указать в команде devnode какой tap-интерфейс вы используете для Openvpn.

## Доступ к удалённому маршрутизатору из домашнено компьютера

И для Windows и для Ubuntu можно из локального компьютера подключаться к удалённому маршрутизатору на vmg.pp.ua. При плохой сети работа с удалённым Linux

десктопом через NX клиента может быть некомфортной. Поэтому можно ограничить роль NX только сбором и запуском топологии.

А далее ...

Все адреса tap-интерфейсов маршрутизаторов mikrotik внутри Qemu для студента D лежат в этой сети 10.D.0.0 /24/. D – ваш номер. В конфигурации Openvpn сервера на vmg.pp.ua для пользователя studentD есть строка **push "route 10.D.0.0 255.255.0.0"** Что нам это даёт? Выполняем на своём компьютере команду (после Openvpn соединения)

**netstat -r**

и видим строку типа (для Windows)

| NetworkDest | Netmask | Gateway | Interface | Metric |
|---|---|---|---|---|
| | | ... | | |
| 10.D.0.0 | 255.255.0.0 | 192.168.3.1 | 192.168.3.2 | 30 |

Есть маршрут на сеть 10.D.0.0/16. Адрес 192.168.3.2 нам дал Openvpn сервер удалённой системы vmg.pp.ua. 192.168.3.1-адрес tap-интерфейса на vmg.pp.ua.

Внутри маршрутизатора Mikrotik надо установить маршрут на домашний компьютер. Если tap интерфейс маршрутизатора Mikrotik имеет адрес 10.D.I.1, то в консоле маршрутизатора надо дать команду

**ip route add dst-address=192.168.3.2/32 gateway=10.D.I.2.**

10.D.I.2- адрес tap интерфейса в удалённой Ubuntu. 192.168.3.2— ваш адрес. Тогда мы видим этот адрес со своего домашнего компьютера.

**Ping 10.D.I.1**

Мы можем кофигурировать этот маршрутизатор, запустив на своей машине Winbox или telnet (в виндовс через putty, который в gns3) или набрав в броузере

**http:// 10.D.I.1**

или, наконец, забрать домой по ftp backup конфигурации

Если связь совсем плохая и не удаётся войти на удалённый рабочий стол, то можно стартовать топологию через командную строку, после соединения по ssh (см. Лабу 1)

**ПутьКGns3/gns3 topology.net&**

изменив в файле topology.net режим запуска на автостарт autostart = True. Соединяётесь с консолями роутеров через телнет в putty и настраивайте их.

Для сдачи работв необходимо

1. Довести до автоматизма процесс создания топологии first, сохранения резервной копии и восстановления из неё топологии first. При этом следует продемонстрировать ввод команд в консоли RouterOS всеми пятью способами.
2. Предоставить доказательство в виде скриншота того, что вы со своего домашнего компьютера зашли через winbox в маршрутизатор Mikrotik под qemu под Ubunu на vmg.pp.ua.
3. Запустить топологию first с помошщью ssh без использования NX-клиента. Запустить локально утилиты winbox и консоли роутеров.
4. Настройте Gns3 под Windows. Соберите и запустите в Gns3 под Windows торологию first. Сравните время загрузки с Ubuntu.

# 3. Маршрутизация

**Статическая маршрутизация**
**Маска /32**
**Динамическая маршрутизация.**
**RIP**
**OSPF**
**Перераспределение маршрутов и BGP**

В сетевом устройстве IP-пакеты бывают входящие и выходящие. Для входящих TCP/UDP-пакетов в устройстве должна быть запущена программа, которая их ждёт и возвращает обратно. На входящие IP-пакеты ICMP-протокола (пинги) отвечает система устройства. Выходящие пакеты могут быть как проходящими (пробрасывемыми), так и порождаемые процессами самого устройства. Для того чтобы сетевое устройство пробрасывало пакеты, на нём должен работать процесс маршрутизации. Тогда устройство становится маршрутизатором или роутером.

Для того, чтобы пакет ушел из устройства для пакета должен быть прописан маршрут. Должна быть строка в таблице маршрутов. Маршрутизация осуществляется только по адресу назначения. Сетевые интерфейсы не понимают IP. Они понимают пакеты 2-го уровня, например ethernet или PPP. Для проброски или отправки IP-пакет должен быть помещён в правильный пакет 2-го уровня. Это делается на основании таблиц маршрутизации плюс ( в случае ethernet) и протокола ARP. Остановимся на ethernet.

## Статическая маршрутизация

Соберём топологию routing. Воспользуйтесь шаблоном template. Для этого сделаем копию routing папки template. Откроем routing в GNS. Убъем роутеры R7,R10,R11. Сохраним проект. Закроем GNS. В папке routing/working убъём папки R7,R10,R11. Снова откроем routing в GNS. Переименуйте устройства (номер не трогайте -это номер тап-интерфейса), измените их иконки и соедините интерфейсы согласно рисунку. Соединить, а затем переименовать не удасться. **Запомните это.** Запустите топологию, назначьте имена и адреса согласно рисунку. Воспользуйтесь телнетом к тап-интерфейсам в разных табах одного окна консоли. Помним, что e0 это ether1, e1это ether2. Обязательно проверьте соседей.

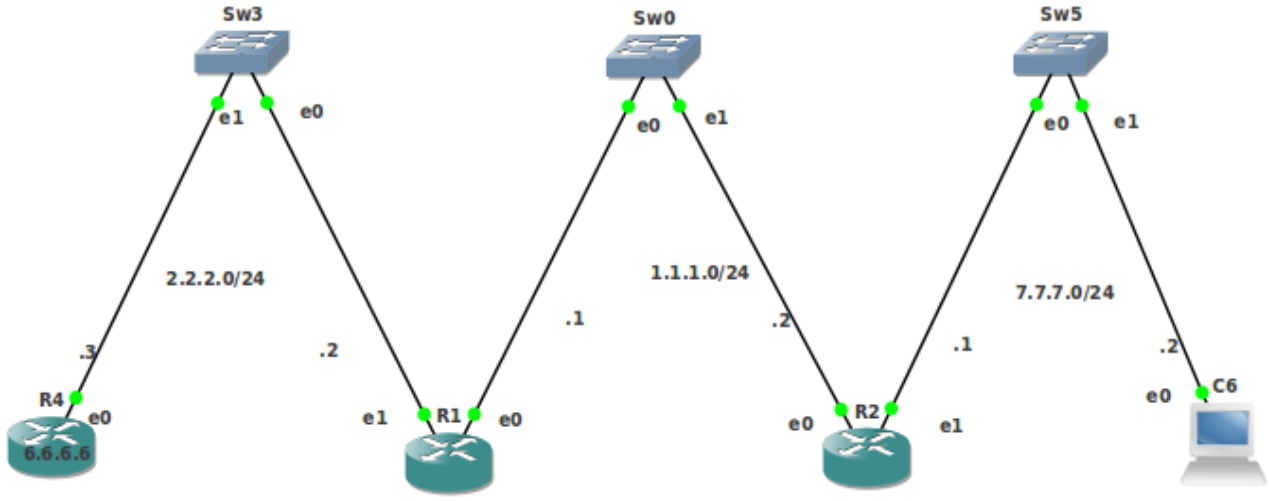



**Рис.1 Топология routing**

Превратим устройства Sw0, Sw3, Sw5 в коммутаторы второго уровня (свичи), обединив интерфейсы e0, e1в мост bridge1

**interface bridge add-добавится момст с именем по умолчанию bridge1**
**interface bridge port add bridge= bridge1 interface=ether1**
**interface bridge port add bridge= bridge1 interface=ether2**

К свичам sw0 sw3 sw5 можно подсоединить ещё много сетевых устройств с адресами из указанных на рисунке IP-сетей. Для сути дела хватит то, что изображено.

Как только на сетевую карточку e0(ether1) R1 назначается адрес, например 1.1.1.1/24, система по маске определяет IP-сеть для этого адреса (1.1.1.0/24) и в таблицу маршрутов добавляется строка, говорящая, что пакеты на 1.1.1.0/24 отправлять на ether1

[admin@R1] > **ip route print**
**Flags: X - disabled, A - active, D - dynamic,**
**C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,**
**B - blackhole, U - unreachable, P - prohibit**

| # | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE |
|---|---|---|---|---|
| 0 | ADC 1.1.1.0/24 | 1.1.1.1 | ether1 | 0 |
| 1 | ADC 2.2.2.0/24 | 2.2.2.2 | ether1 | 0 |
| 2 | ADC 10.0.1.0/24 | 10.0.1.1 | ether7 | 0 |

Здесь 24-число единиц в маске и в столбце REF-SRC указан адрес источника . В других обозначениях это маска 255.255.255.0 . Далее система сможет обработать любой выходящий или проходящий пакет с адресом назначения, лежащим в этой сети 1.1.1.0/24. Пусть это будет пакет с адресом назначения 1.1.1.2. Система видит, что пакет подпадает под строку таблицы маршрутизации. И, согласно этой строке, видит, что путь к адресу 1.1.1.2 лежит через интерфейс ether1. Для определения ethernet адреса, соответствующего адресу 1.1.1.2, система обращается в ARP-таблицу состветсвия MAC-адрес-IP-адрес (**ip arp print**). Если такого соответствия нет, то система шлёт через интерфейс ether1 широковещательный эсернет-пакет с ARP-запросом "Устройство имеющее адрес 1.1.1.2, сообщи свой MAC-адрес". Устройство R2 с адресом 1.1.1.2 в есернет-пакете ARP-ответа сообшает MAC-адрес **M** своей карточки e0 (ether1). IP-Пакет на R1 в сторону 1.1.1.2 помещается в ethernet пакет с MAC-адресом назначения **M** и отправляется из сетевой карточки e0(ether1) R1 на сетевую карточки e0(ether1) R2.

На устройстве R2 с адресом 1.1.1.2 в свою очередь должен быть прописан маршрут в обратную сторону. В нашем случае он добавляется автоматически системой

[admin@R2] > **ip route print**
**Flags: X - disabled, A - active, D - dynamic,**
**C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,**
**B - blackhole, U - unreachable, P - prohibit**

| # | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE |
|---|---|---|---|---|
| 0 | ADC 1.1.1.0/24 | 1.1.1.2 | ether1 | 0 |
| 1 | ADC 7.7.7.0/24 | 7.7.7.1 | ether2 | 0 |
| 2 | ADC 10.0.2.0/24 | 10.0.2.1 | ether7 | 0 |

То есть маршрутизатор безо всяких настроек осуществляет проброску пришедших пакетов из одной непосредственно присоединённой IP-сети в другую. Если надо пробросить или отправить IP-пакет, предназначенный для IP-сети, которая не присоединена непосредственно к маршрутизатору, необходимо добавить соответствующую строку в таблицу маршрутов. Это автоматически делают протоколы динамической маршрутизации,

согласно заданным в них правилам. Это можно сделать и с помощью команд статической маршрутизации. Например, команда для R1

**ip route  add dst-address=7.7.7.0/24 gateway=1.1.1.2**

добавляет в таблицу маршрутов строку

[admin@R1] > **ip route print**

**2 A S  7.7.7.0/24  1.1.1.2  1**

о том, что  пакеты на 7.7.7.0/24 отправлять на 1.1.1.2 . Здесь dst-address - сеть назначения (7.7.7.0/24). В качестве адреса шлюза gateway надо брать ближайший (но не свой) адрес в направлении к сети назначения. Далее система сможет обработать любой выходящий или проходящий пакет с адресом назначения, лежащим в сети  7.7.7.0/24. Она их будет направлять по адресу 1.1.1.2. Или точнее. Этот адрес лежит в сети 1.1.1.0/24, для которой уже существует строка в таблице маршрутизации. Согласно этой строке, пакеты  для 1.1.1.2 направляются на интерфейс ether1. Они помещаются в ethernet-пакеты  с адресом назначения равным  ethernet-адресу, соответствующему  IP-адресу 1.1.1.2.

Вся IP-сеть обозначается как 0.0.0.0/0. Маршрут на неё называется маршрутом по умолчанию. Для R1 его можно задать так

[admin@R1] > **ip route  add gateway=2.2.2.3**

Важно, что адрес 2.2.2.3 должен лежать в присоединённой к R1 сети 2.2.2.0/24, то есть в начале на R1 рекомендуется пропинговать шлюз

[admin@R1] > **ping 2.2.2.3**

Следует помнить, что для прохождения пинга из исходной точки в точку назначения необходимо, чтобы каждом хосте, куда попадёт пакет пинга  должен быть маршрут в точку конечного назначения; и при возврате этого пакета из точки назначения в исходную точку на каждом хосте, куда попадёт этот пакет пинга  должен быть маршрут в исходную точку.

Весь мир представит интерфейс-петля bridge1 на R4 с произвольным адресом 6.6.6.6/32

[admin@R4] > **interface bridge add**

[admin@R4] > **ip address add  address=6.6.6.6/32 interface=bridge1**

Теперь в нашей таблице маршрутов на R1следующие сроки

| | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE |
|---|---|---|---|---|
| 0 | A S  0.0.0.0/0 | | 2.2.2.3 | 1 |
| 1 | ADC  1.1.1.0/24 | 1.1.1.1 | ether1 | 0 |
| 2 | ADC  2.2.2.0/24 | 2.2.2.2 | ether2 | 0 |
| 3 | A S  7.7.7.0/24 | | 1.1.1.2 | 1 |
| 4 | ADC  10.0.1.0/24 | 10.0.1.1 | ether7 | 0 |

Один и тот же пакет может удовлетворять нескольким строкам таблицы. Так пакеты в сторону сети 1.1.1.0/24 удовлетворяют и правилу 0 и правилу 1. Система всегда использует более точное правило, или правило для сети у которой маска имеет больше единиц. В нашем случае это правило 1.

Добавим маршруты на остальных устройствах. Компьютер C6 видит весь мир через адрес 7.7.7.1

admin@ C6] > **ping 7.7.7.1**

| HOST | SIZE | TTL | TIME | STATUS |
|---|---|---|---|---|
| 7.7.7.1 | 56 | 64 | 4ms | |

Поэтому

admin@ C6 ] > **ip route  add gateway=7.7.7.1**

Роутер R4 видит весь мир через 2.2.2.2

[admin@R4] > **ping 2.2.2.2**

| HOST | SIZE | TTL | TIME | STATUS |
|---|---|---|---|---|
| 2.2.2.2 | 56 | 64 | 4ms | |

Поэтому

[admin@R4] > **ip route add gateway=2.2.2.2**

Роутер R2 видит весь мир через 1.1.1.1. Поэтому

[admin@R2] > **ip route add gateway=1.1.1.1**

Маршруты прописаны. Проверим, для самого тяжёлого случая: пинг из внешнего мира в сторону C6

[admin@R4] **> ping 7.7.7.2 src-address=6.6.6.6**

| **HOST** | **SIZE** | **TTL** | **TIME** | **STATUS** |
|---|---|---|---|---|
| **7.7.7.2** | **56** | **62** | **1ms** | |
| **7.7.7.2** | **56** | **62** | **2ms** | |

Посмотрим ARP-таблицы. Например

[admin@R2] > ip arp pr

Flags: X - disabled, I - invalid, H - DHCP, D - dynamic

| # | ADDRESS | MAC-ADDRESS | INTERFACE |
|---|---|---|---|
| 0 | D 1.1.1.1 | 00:AA:00:A3:7E:00 | ether1 |
| 1 | D 7.7.7.2 | 00:AA:00:4C:4F:00 | ether2 |

Найдите эти MAC-адреса эсернет интерфейсов у соседних устройств R1 и C6 (команда **int eth pr**).

Если два устройства соединены через свич, то с точки зрения этих устройств в рамках сетевого уровня этот свич можно убрать и соединить устройства напрямую. Обратно. Если два устройства соединены связью с адресами на концах из какой-то сети, то это эквивалентно соединению этих устройств через свич. Причем к свичу можно подсоединить устройства с адресами из этой сети. Эти устройства и исходные два устройства будут видеть друг друга по протоколу IP. Т.е. при рассмотрении маршрутизации можно вообще не использовать свичи.

Остановите топологию. Сделайте копию папки routing. Переименуйте её в routingBezSw. Откройте routingBezSw. Измените топологию согласно рисунку Рис.2, не перепутав интерфейсы

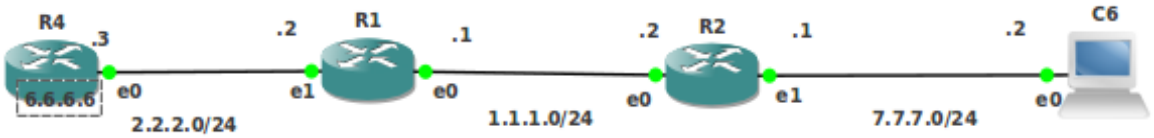



**Рис.2 Топология routingBezSw**

Проверим, для самого тяжёлого случая: пинг из внешнего мира в сторону C6

[admin@R4] > **ping 7.7.7.2 src-address=6.6.6.6**

В рамках одной ethernet-сети можно оргагизовать более, чем одну IP-сеть.

**Задание.** Организуйте ethernet-сеть из пяти устройств (0 1 2 3 4), путём их подключения к одному свичу. Топологию назовите Zadanie.Устройства 0 1 2 поместите в одну IP-сеть, а Устройства 0 3 4 поместите в другую IP-сеть. Устройство 0 будет маршрутизатором. У него на один ethernet интерфейс назначьте IP-адреса из разных IP-сетей. Путём назначения статических маршрутов для устройств 1 2 3 4 добейтесь, чтобы все устройства пигновали друг друга по кругу.

Младший адрес в IP-сети определяет саму сеть, а старший предназначен для широковещания. Поэтому реальное число адресов в сети равно 2^d-2, где d-число нулей в маске.

Для сетей с маской /24 это 2^8-2=254. Для сетей с маской /30 это 2^8-2=2. Маска /31 -смысла не имеет

## Маска /32

Интерес представляют адреса с маской /32, используемые при связи точка-точка. Соберём новую топологию M32 из двух устройств R1 и R2, соединив их интефейсы ether1. Стандартно при назначении адреса интерфейс помещается в сеть, которая определяется этим адресом. Назначим адреса особым образом, заменив сеть, в которую помещается интерфейс на адрес противоположной стороны

[admin@ R1]>**ip address add address=2.2.2.2/32 network=3.3.3.3 interface=ether1**
[admin@ R2]>**ip address add address=3.3.3.3/32 network=2.2.2.2 interface=ether1**

Посмотрим таблицы маршрутов

[admin@ R1]>**ip route print detail**
**dst-address=3.3.3.3/32 gateway=ether1**
[admin@ R2]>**ip route print detail**
**dst-address=2.2.2.2/32 gateway=ether1**

R1 пингует R2 и наоборот

**1.** Изучим маршрутизацию для адресов с маской /32. Соберите топологию PtPRouting, изображённую на рисунке Рис.3. Назначим адреса. Образуем соединение точка-точка между R2 и R3.

[admin@ R2]>**ip address add address=1.1.1.1/32 network=2.2.2.2 interface=ether1**
[admin@ R3]>**ip address add address=2.2.2.2/32 network= 1.1.1.1 interface=ether1**

Образуем соединение точка-точка между R3 и R4.

[admin@ R3]>**ip address add address=3.3.3.3/32 network=4.4.4.4 interface=ether1**
[admin@ R4]>**ip address add address=4.4.4.4/32 network=3.3.3.3 interface=ether1**

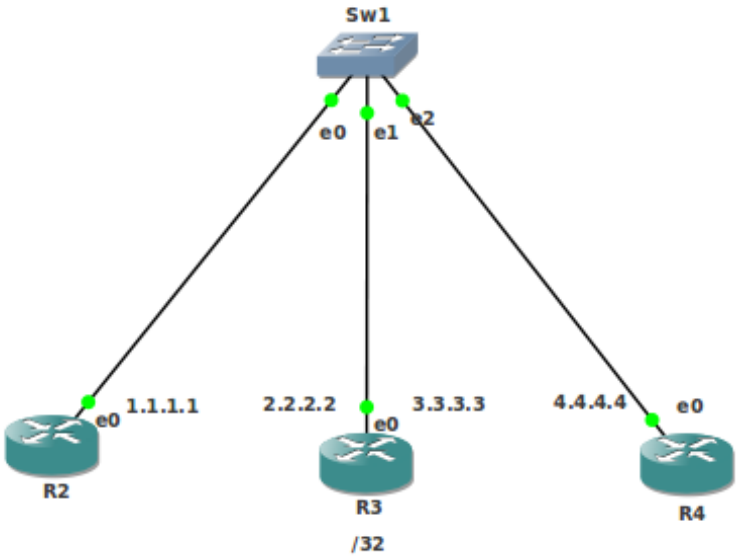


**Рис.3 Топология PtPRouting**

Две “точки” в R3 слились в одну на интерфейсе e0. Назначим маршруты
[admin@ R2]>**ip route add dst-address=4.4.4.4/32 gateway=2.2.2.2**
[admin@ R3]>**ip route add dst-address=1.1.1.1/32 gateway=3.3.3.3**

R2 пингует R4 и наоборот. Уберём свич, согласно Рис.4, образуя топологию PtPRoutingBezSw

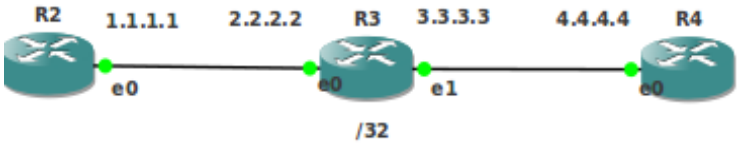



**Рис.4 Топологии** PtPRoutingBezSw routingRIPPtP routingRIPPtP1 routingOSPFPtP

На R3 сменим интерфейс для адреса 3.3.3.3
[admin@R3] > **ip ad pr**
Flags: X - disabled, I - invalid, D - dynamic
```
 #   ADDRESS          NETWORK        INTERFACE
 0   10.0.3.1/24      10.0.3.0       ether7
 1   2.2.2.2/32       1.1.1.1        ether1
 2   3.3.3.3/32       4.4.4.4        ether1
```
[admin@R3] > **ip ad set 2 interface=ether2**

Крайние роутеры опять пингуют друг друга

**2.** Для соединия двух IP-сетей обычно используют третью IP-сеть между ними. Можно избежать привлечения третьей сети, если использовать так называемую ненумерованную (unnumbered) адресацию. Имеем две эсеернет сети QEMU1,QEMU3 и QEMU2,QEMU4. В них настроены IP-сети 1.1.1.0/24 и 1.1.2.0/24. Рисунок Рис.5 отражает упрощённую топологию unnumbered: между QEMU1 и QEMU3 ( QEMU2 и QEMU4) можно поместить свич и к нему присоединить устойства с адресами из сети 1.1.1.0/24 ( 1.1.2.0/24 ). Соединим эсернет-сети, соединив QEMU1 и QEMU2 образуя единую эсернет-сеть

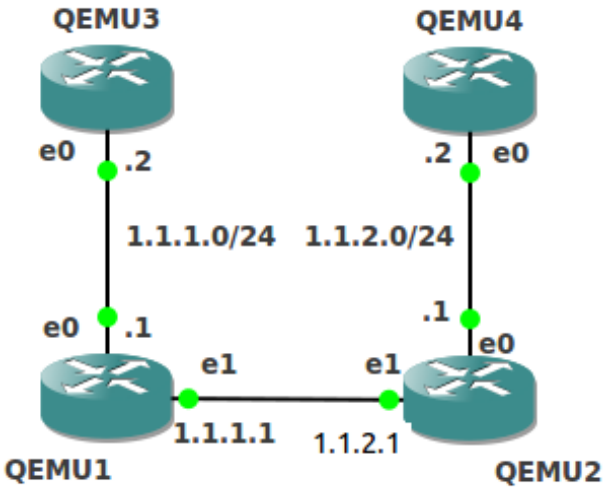


**Рис.5 Топология** unnumbered

.

Для соединия двух IP-сетей 1.1.1.0/24 и 1.1.2.0/24 обычно используют третью IP-сеть, скажем1.1.3.0/30, настроенную на роутерах QEMU1 и QEMU2 . Можно избежать привлечения третьей сети, если использовать так называемую ненумерованную (unnumbered) адресацию.

Настроим первую IP-сеть

QEMU1>**ip address add address=1.1.1.1/24 interface=ether1**

Назначим ненумерованный адрес

QEMU1>**ip address add address=1.1.1.1/32 network=1.1.2.1 interface=ether2**

Адрес 1.1.1.1 назначен на разные интерфейсы и сразными масками

QEMU3>**ip address add address=1.1.1.2/24 nterface=ether1**

QEMU3>**ip route add gateway= 1.1.1.1**

Настроим вторую IP-сеть

QEMU2>**ip address add address=1.1.2.1/24 interface=ether1**

Назначим ненумерованный адрес

QEMU2>**ip address add address=1.1.2.1/32 network=1.1.1.1 interface=ether2**

Адрес 1.1.2.1 назначен на разные интерфейсы и с разными масками

QEMU4>**ip address add address=1.1.2.2/24 interface=ether1**

QEMU4>**ip route add gateway= 1.1.2.1**

Настроим шлюзы между сетями

QEMU1>**ip route add gateway=1.1.2.1**

QEMU2>**ip route add gateway=1.1.1.1**

QEMU3 и QEMU4 увидят друг друга.

## Динамическая маршрутизация.

Прописать все статические маршруты уже для средней сети - задача нереальная. На помощь приходят протоколы динамической маршрутизации. Рассмотрим, как настраивать маршрутизацию в протоколах RIP OSPF BGP.

Теорию рассматривать не будем. Ограничимся соображениями, относящимися к настройке. Протоколы динамической маршрутизации работают в пределах какой-то сети. Устройства, на которых активен данный протокол, обмениваются маршрутной информацией друг с другом в пределах этой сети. В ходе этого обмена у каждого устройства сети через некоторое время так изменяется таблица маршрутов, что устройства в сети начинают видеть друг друга. Конкретно, кто кого увидит, определяется начальными настройками протоколов маршрутизации на каждом устройстве.

Протокол BGP предназначен для обмена маршрутной информацией между автономными системами - большими сетями с единым администратором, в которых уже настроена маршрутизация. При настройке BGP следует явно указать с какими автономными системами надо установить канал обмена маршрутной информацией. BGP в других

автономных системах также должно установить канал. Стороны на концах каналов называют пирами.

Для протоколов RIP и OSPF устройство обменивается маршрутной информацией со своими соседями. Состав информации определяется настройками.

В рамках OSPF устройство отсылает соседям информацию о своих связях с соседями. Оно же и получает такую информацию от своих соседей. Далее каждое устройство на основании полученной информации строит карту (граф, топологию) всей сети. Затем по этой карте создаёт таблицу маршрутов.

В рамках RIP устройства обмениваются между собой целыми таблицами маршрутов. В начале устройство отправляет соседям только информацию о непосредственно присоединённых сетях и получает такую же от соседей. Далее устройство отправляет соседям обновлённую таблицу. Принцип прост: если от соседа с адресом 1.1.1.1 в присланной таблице маршрутов есть строка 7.7.7.0/24 *, то значит сосед знает, что делать с пакетами в сеть 7.7.7.0/24 и значит их можно ему отправлять, то есть в свою таблицу надо добавить строку 7.7.7.0/24 1.1.1.1

При настройке RIP и OSPF следует указать адреса сетей, которые мы хотим включить в процесс маршрутизации. Для настройки BGP надо указать адрес пира.

Можно настроить протоколы, чтобы они получали маршрутную информацию и из таких источников: непосредственно-подключенные маршруты, статические маршруты и другие протоколы маршрутизации. То есть использовать перераспределение (redistribution) маршрутов. Например. Пусть в сети A используется OSPF, а в сети B - RIP. Имеется общий для двух сетей маршрутизатор Z. Тогда на маршрутизаторе Z для RIP надо включить перераспределение OSPF, а для OSPF надо включить перераспределение RIP.

## RIP

Сделаем копию routingRIP топологии routingBezSw. Запустим routingRIP и убъём все статические маршруты на всех устройствах. Например для R1

```
[admin@R1] > ip ro pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS       PREF-SRC        GATEWAY         DISTANCE
 0 A S  0.0.0.0/0                        2.2.2.3         1
 1 ADC  1.1.1.0/24       1.1.1.1         ether1          0
 2 ADC  2.2.2.0/24       2.2.2.2         ether2          0
 3 A S  7.7.7.0/24                       1.1.1.2         1
 4 ADC  10.0.1.0/24      10.0.1.1        ether7          0
[admin@R1] > ip ro remove 0,3
[admin@R1] > ip ro pr
0 ADC  1.1.1.0/24       1.1.1.1         ether1          0
1 ADC  2.2.2.0/24       2.2.2.2         ether2          0
2 ADC  10.0.1.0/24      10.0.1.1        ether7          0
```

Добавим на R6 интерфейс-петлю bridge1 с адресом 5.5.5.5/32

```
[admin@ R6] > interface bridge add
```

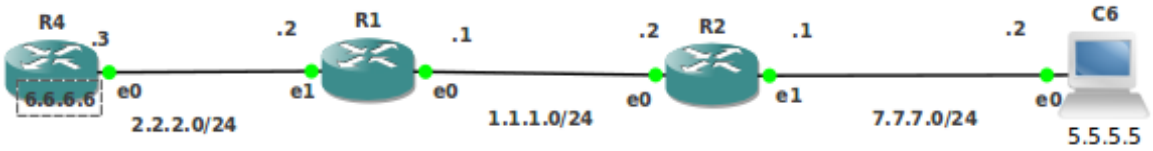



**Рис.6. Топологии routingRIP, routingRIP1, routingOSPF**

[admin@ R6] > **ip address add address= 5.5.5.5/32 interface=bridge1**

**1.** На всех устройствах разрешим перераспределение непосредственно-подключенных маршрутов

**routing rip set redistribute-connected=yes**

Укажем соседей

**[admin@R4] routing rip neighbor add address=2.2.2.2**
**[admin@R1] routing rip neighbor add address=2.2.2.3**
**[admin@R1] routing rip neighbor add address=1.1.1.2**
**[admin@R2] routing rip neighbor add address=1.1.1.1**
**[admin@R2] routing rip neighbor add address=7.7.7.2**
**[admin@C6] routing rip neighbor add address=7.7.7.1**

Крайние роутеры пингуют друг друга

**[admin@R4] ping 5.5.5.5 src-address=6.6.6.6**
**[admin@C6] ping 6.6.6.6 src-address= 5.5.5.5**

Посмотрим лог

[admin@C6] >**sys log add topics=route action= memory**
[admin@C6] >**log print follow-only**
**17:38:10 route,rip,debug ---=== RIPv2 RESPONSE ===---**
**17:38:10 route,rip,debug prefix=5.5.5.5/32 metric=16 nexthop=0.0.0.0 tag=0**
**17:38:10 route,rip,debug prefix=6.6.6.6/32 metric=3 nexthop=0.0.0.0 tag=0**
**17:38:36 route,rip,debug 44 bytes sent to 224.0.0.9 via bridge1:**
**17:38:36 route,rip,debug ---=== RIPv2 RESPONSE ===---**
**17:38:36 route,rip,debug prefix=5.5.5.5/32 metric=16 nexthop=0.0.0.0 tag=0**
**17:38:36 route,rip,debug prefix=6.6.6.6/32 metric=4 nexthop=0.0.0.0 tag=0**
**17:38:36 route,rip,debug 44 bytes sent to 7.7.7.1 via ether1:**

Широковещание

**17:38:36 route,rip,debug 44 bytes sent to 224.0.0.9 via bridge1:**

пойдёт через bridge1 и не выйдет во внешний мир. Маршрутная информация будет передана в режиме точка-точка

**17:38:36 route,rip,debug 44 bytes sent to 7.7.7.1 via ether1:**

**2.** Сделаем копию routingRIP1 топологии routingRIP. Работаем с копией. На всех устройствах отключим перераспределение непосредственно-подключенных маршрутов

**routing rip set redistribute-connected=no**

Добавим сети для маршрутизации по протоколу RIP

[admin@R4] > **routing rip network add network=6.6.6.6/32**

[admin@C6] >**routing rip network add network=5.5.5.5/32**

Интересно наблюдать, что такие пинги

[admin@R4] > **ping 5.5.5.5 src-address=6.6.6.6**
[admin@C6] > **ping 6.6.6.6 src-address=5.5.5.5**

проходят

А такие

[admin@R4] > **ping 5.5.5.5**
[admin@C6] > **ping 6.6.6.6**

нет

У первых пингов адрес источника либо приёмника равен **5.5.5.5 или 6.6.6.6 .** Посмотрите таблицы марщрутов на всех устройствах вы увидите маршруты на соответствующие сети.

Рассмотрим

[admin@R4] > **ping 5.5.5.5**

У ICMP-пакета адрес источника равен адресу R4 2.2.2.3. Пакет дойдёт до адреса 5.5.5.5 на C4, но не вернётся, так как на C4 маршрута в сторону сети 2.2.2.3 2.2.2.0/24 -нет.

Добавим сети

[admin@R4] > **routing rip network add network=2.2.2.0/24**
[admin@R1] > **routing rip network add network=2.2.2.0/24**
[admin@R1] > **routing rip network add network=1.1.1.0/24**
[admin@R2] > **routing rip network add network=1.1.1.0/24**
[admin@R2] **> routing rip network add network=7.7.7.0/24**
[admin@C6] >**routing rip network add network=7.7.7.0/24**

Теперь все устройства в сети попарно пингуют друг друга. Снова посмотрим лог

[admin@C6] >**log print follow-only**

Увидим, что широковещание

**17:51:26 route,rip,debug 104 bytes sent to 224.0.0.9 via ether1:**

пойдёт через ether1 и выйдет во внешний мир. Если C6 подключен к R2 через свич (как оно обычно и бывает), то широковещание пойдёт на другие устройства, подключенные к свичу, что не всегда желательно. Таким образом использование сетей при настройке RIP приводит к широковещанию.

**3.** Рассмотрим RIP для соединений точка-точка. Возьмите топологию PtPRoutingBezSw, изображённую на Рис.4. Сделайте копию routingRIPPtP. Запустите её и уничтожьте статические маршруты. На всех устройствах разрешим перераспределение непосредственно-подключенных маршрутов

**routing rip set redistribute-connected=yes**

Укажем соседей

[admin@R2] > **routing rip neighbor add address=2.2.2.2**
[admin@R3] > **routing rip neighbor add address=1.1.1.1**
[admin@R3] > **routing rip neighbor add address=4.4.4.4**
[admin@R4] > **routing rip neighbor add address=3.3.3.3**

Крайние роутеры пингуют друг друга. В таблицах маршрутах есть пути на сети tap-интерфейсов, например

[admin@R2] **> ip ro pr**

```
2 ADC  10.0.2.0/24       10.0.2.1       ether7          0
3 ADr  10.0.3.0/24                      2.2.2.2         120
4 ADr  10.0.4.0/24                      2.2.2.2         120
```

tap-интерфейсы 10.0.4.1 в R4 пинуется из R2 . Другая сторона tap-интерфейса 10.0.4.2 (которая в Ubuntu) нет. Это потому, что у Ubuntu нет маршрута в нашу сеть.

Сделаем копию routingRIPPtP1 топологии routingRIPPtP. Работаем с копией. На всех устройствах отключим перераспределение непосредственно-подключенных маршрутов
**routing rip set redistribute-connected=no**
Соседей можно убрать

**routing rip neighbor print**

**routing rip neighbor remove …**

Добавим особым образом сети для маршрутизации по протоколу RIP.

```
[admin@R2] > ip ad pr
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   10.0.2.1/24        10.0.2.0        ether7
 1   1.1.1.1/32         2.2.2.2         ether1
[admin@R2] > routing rip network add network=2.2.2.2
[admin@R3] > ip ad pr
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   10.0.3.1/24        10.0.3.0        ether7
 1   2.2.2.2/32         1.1.1.1         ether1
 2   3.3.3.3/32         4.4.4.4         ether2
[admin@R3] > routing rip network add network=1.1.1.1
[admin@R3] > routing rip network add network=4.4.4.4
[admin@R4] > ip ad pr
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   10.0.4.1/24        10.0.4.0        ether7
 1   4.4.4.4/32         3.3.3.3         ether1
[admin@R4] > routing rip network add network=3.3.3.3
[admin@R4] > ping 1.1.1.1
HOST                                     SIZE  TTL TIME  STATUS
1.1.1.1                                    56   63  1ms
```

Пинги идут. В таблицах маршрутах нет путей на сети чужих тап-интерфейсов, например

```
[admin@R2] > ip ro pr
0 ADC  2.2.2.2/32        1.1.1.1         ether1           0
 1 ADr  4.4.4.4/32                      2.2.2.2          120
 2 ADC  10.0.2.0/24       10.0.2.1        ether7           0
```

## OSPF

**1.** Возьмите топологию routingRIP на рис. 6. Сделайте копию routingOSPF. Запустите её и уничтожте RIP-настройки. Настроим OSPF самым простым образом

```
[admin@R4] > routing ospf network add network=6.6.6.6.32 area=backbone
[admin@R4] > routing ospf network add network=2.2.2.0/24 area=backbone
[admin@R1] > routing ospf network add network=2.2.2.0/24 area=backbone
[admin@R1] > routing ospf network add network=1.1.1.0/24 area=backbone
[admin@R2] > routing ospf network add network=1.1.1.0/24 area=backbone
```

[admin@R2] **> routing ospf network add network=7.7.7.0/24 area=backbone**
[admin@C6] **>routing ospf network add network=7.7.7.0/24 area=backbone**
[admin@C6] **>routing ospf network add network=5.5.5.5/32 area=backbone**
**[admin@C6] > ip ro pr**
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit

| # | | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE |
|---|---|---|---|---|---|
| 0 | ADo | 1.1.1.0/24 | | 7.7.7.1 | 110 |
| 1 | ADo | 2.2.2.0/24 | | 7.7.7.1 | 110 |
| 2 | ADC | 5.5.5.5/32 | 5.5.5.5 | bridge1 | 0 |
| 3 | ADo | 6.6.6.6/32 | | 7.7.7.1 | 110 |
| 4 | ADC | 7.7.7.0/24 | 7.7.7.2 | ether1 | 0 |
| 5 | ADC | 10.0.6.0/24 | 10.0.6.1 | ether7 | 0 |

Проверим
**[admin@C6] > ping 6.6.6.6 src-address=5.5.5.5**

| HOST | SIZE | TTL | TIME | STATUS |
|---|---|---|---|---|
| 6.6.6.6 | 56 | 62 | 12ms | |

2. Рассмотрим OSPF для соединений точка точка. Возьмите топологию routingRIPPtP. Сделайте копию routingOSPFPtP. Запустите её и уничтожьте там RIP-настройки.
**routing rip network print**
**routing rip network remove …**
Добавим особым образом сети для маршрутизации по протоколу OSPF.
[admin@R2] > **routing ospf network add network=2.2.2.2 area=backbone**
[admin@R3] > **routing ospf network add network=1.1.1.1 area=backbone**
[admin@R3] > **routing ospf network add network=4.4.4.4 area=backbone**
[admin@R4] > **routing ospf network add network=3.3.3.3 area=backbone**
Проверим
[admin@R4] > **ping 1.1.1.1**

| **HOST** | **SIZE** | **TTL** | **TIME** | **STATUS** |
|---|---|---|---|---|
| **56** | **63** | **1ms** | | |

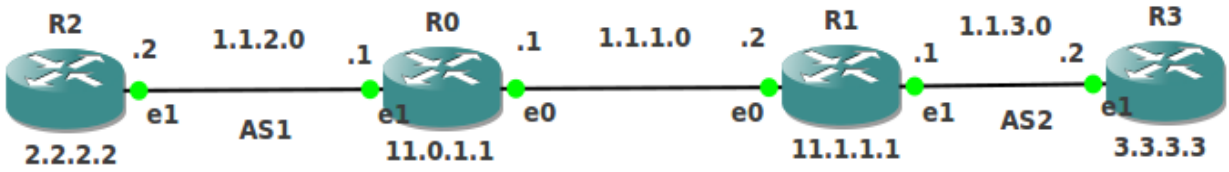



**Рис 7. топология bgp**

## Перераспределение маршрутов и BGP

Соберите топологию bgp изображённую на Рис 7. Запустите её и назначьте адреса согласно рисунку. В каждом устройстве создайте пустые мосты bridge1 и назначьте на них адреса 2.2.2.2 11.0.1.1 11.1.1.1 3.3.3.3 с маской /32 согласно рисунку. Остальные адреса имеют

маску /24. Топология состоит из двух автономных систем 1 и 2. В каждой из них настройте OSPF

AS1

**[admin@R0] > routing ospf network add network=1.1.2.0/24 area=backbone**
**[admin@R2] > routing ospf network add network=1.1.2.0/24 area=backbone**
**[admin@R2] > routing ospf network add network=2.2.2.2/32 area=backbone**

AS2

**[admin@R1] > routing ospf network add network=1.1.3.0/24 area=backbone**
**[admin@R3] > routing ospf network add network=1.1.2.0/24 area=backbone**
**[admin@R3] > routing ospf network add network=3.3.3.3/32 area=backbone**

На R0 и R1 благодаря OSPF появятся маршруты

```
[admin@R0] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS       PREF-SRC       GATEWAY         DISTANCE
 2 ADo  2.2.2.2/32                 1.1.2.2              110
[admin@R1] > ip route  print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS       PREF-SRC       GATEWAY         DISTANCE
 2 ADo  3.3.3.3/32                 1.1.3.2              110
```

Настроим BGP для связи автономных систем AS1 и AS1. Назначим номера автономных систем 1 и 2

**[admin@R0] >routing bgp instance set 0 as=1 router-id=11.0.1.1**
**[admin@R1] >routing bgp instance set 0 as=2 router-id=11.1.1.1**

Рекомендуется устанавливать BGP-сессию между интерфейсами петлями, в роли которых у нас выступают пустые мосты. Но сначала необходимо настроить маршрут между адресами мостов 11.0.1.1 и 11.0.1.1

**[admin@R0] >ip ro add dst-address=11.1.1.1/32 gateway=1.1.1.2**
**[admin@R1] >ip ro add dst-address=10.1.1.1/32 gateway=1.1.1.1**

Установим BGP-сессию

**[admin@R0] > routing bgp peer add remote-address=11.1.1.1 remote-as=2 multihop=yes update-source=bridge1**
**[admin@R1] > routing bgp peer add remote-address=10.1.1.1 remote-as=1 multihop=yes update-source=bridge1**

**multihop=yes так как между мостами в R0 и R1 нет прямой связи. В качестве источника BGP-обновлений выбран мост bridge1**.

Об установке BGP-сессии лучше нам покажет Windox в поле Uptime меню routing bgp peer. Смотрим на маршруты. Они не менялись. Включим в BGP перераспределение OSPF-маршрутов

**[admin@R0] > routing bgp instance set 0 redistribute-ospf=yes**
**[admin@R1] > routing bgp instance set 0 redistribute-ospf=yes**

На R0 и R1 появятся новые маршруты,

**[admin@R0] > ip route print**

Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit

| # | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE | |
|---|---|---|---|---|---|
| 2 | ADo | 2.2.2.2/32 | 1.1.2.2 | 110 | |
| 3 | ADb | 3.3.3.3/32 | 11.1.1.1 | 20 | **от OSPF на R1** |

[admin@R1] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit

| # | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE | |
|---|---|---|---|---|---|
| 2 | ADb | 2.2.2.2/32 | 11.0.1.1 | 20 | **от OSPF на R0** |
| 3 | ADo | 3.3.3.3/32 | 1.1.3.2 | 110 | |

На R2 и R3 ничего не изменилось.
Включим в OSPF перераспределение BGP -маршрутов.
[admin@R0] > routing ospf instance set 0 redistribute-bgp=as-type-1
[admin@R1] > routing ospf instance set 0 redistribute-bgp=as-type-1
На R0 и R1 ничего не изменилось. Но на R2 и R3 появились новые маршруты.
[admin@R2] > ip ro pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit

| # | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE | |
|---|---|---|---|---|---|
| 2 | ADo | 3.3.3.3/32 | 1.1.2.1 | 110 | **от** BGP **на R0** |

[admin@R3] > ip ro pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit

| # | DST-ADDRESS | PREF-SRC | GATEWAY | DISTANCE | |
|---|---|---|---|---|---|
| 1 | ADo | 2.2.2.2/32 | 1.1.3.1 | 110 | **от** BGP **на R1** |

Следующие пинги будут успешными
**[admin@R2] > ping 3.3.3.3 src-address=2.2.2.2**
**[admin@R3] > ping 2.2.2.2 src-address=3.3.3.3**

А эти нет
**[admin@R2] > ping 3.3.3.3**
**[admin@R3] > ping 2.2.2.2**

Пинги дойдут до назначения но не смогут вернуться. Это объясняется тем, что по пути обратного следования пакеты пингов не найдут маршрута к источнику.

Заставим BGP анонсировать сети
**[admin@R0] > routing bgp network add network=1.1.2.0/24**
**[admin@R1] > routing bgp network add network=1.1.3.0/24**
Теперь пинги пойдут.

Итоговая таблица маршрутов (без tap-интерфейсов)

**[admin@R2] > ip ro pr**

```
 0 ADC  1.1.2.0/24        1.1.2.2         ether2              0
 1 ADo  1.1.3.0/24                        1.1.2.1           110
 2 ADC  2.2.2.2/32        2.2.2.2         bridge1             0
 3 ADo  3.3.3.3/32                        1.1.2.1           110
```

**[admin@R0] > ip route print**

```
 0 ADC  1.1.1.0/24        1.1.1.1         ether1              0
 1 ADC  1.1.2.0/24        1.1.2.1         ether2              0
 2 ADb  1.1.3.0/24                        11.1.1.1           20
 3 ADo  2.2.2.2/32                        1.1.2.2           110
 4 ADb  3.3.3.3/32                        11.1.1.1           20
 6 ADC  11.0.1.1/32       11.0.1.1        bridge1             0
 7 A S  11.1.1.1/32                       1.1.1.2             1
```

**admin@R1] > ip route  print**

```
 0 ADC  1.1.1.0/24        1.1.1.2         ether1              0
 1 ADb  1.1.2.0/24                        11.0.1.1           20
 2 ADC  1.1.3.0/24        1.1.3.1         ether2              0
 3 ADb  2.2.2.2/32                        11.0.1.1           20
 4 ADo  3.3.3.3/32                        1.1.3.2           110
 6 A S  11.0.1.1/32                       1.1.1.1             1
 7 ADC  11.1.1.1/32       11.1.1.1        bridge1             0
```

**[admin@R3] > ip ro pr**

```
 0 ADo  1.1.2.0/24                        1.1.3.1           110
 1 ADC  1.1.3.0/24        1.1.3.2         ether2              0
 2 ADo  2.2.2.2/32                        1.1.3.1           110
 3 ADC  3.3.3.3/32        3.3.3.3         bridge1             0
```

Например ни из R2 ни из R3 недоступна сеть 1.1.1.0/24. В этом нет недостатка. Эта сеть ничто иное как канал связи между автономными системами. Устойства, входящие в эту сеть пингуются по другим адресам. При желании легко настроить маршруты в эту сеть из R2 и R3. Мы не будем этого делать.

Для сдачи этой Лабы необходимо предъявить 13 работающих топологий routing routingBezSw Zadanie M32 PtPRoutingBezSw unnumbered routingRIP routingRIP1 routingRIPPtP routingRIPPtP1 routingOSPF routingOSPFPtP bgp.

# 4. DHCP, преобразование адресов и балансировка нагрузки

**Хост-машина Ubuntu как модель интернета**
**DHCP и преобразование адресов источника**
**Преобразование адресов приёмника**
**Балансировка нагрузки**

## Хост-машина Ubuntu как модель интернета

Все адреса tap-интерфейсов внутри маршрутизаторов Mikrotik должны иметь вид 10.D.x.1/24, где D –ваш номер,а x – номер маршрутизатора в вашей текущей запущенной в gns3 топологии. Вы сами их должны назначить в консолях маршрутизаторов в запущенной топологии. tap-интерфейсу внутри маршрутизатора Mikrotik с номером x соответствует tap-интерфейс ubuntu, помещённый в мост с адресом 10.D.x.2/24. В ubuntu включена проборска IP- пакетов (forvarding). Это означает, что любые несоединённые в gns3 маршрутизаторы Mikrotik (и не только ваши) могут связаться по IP. Для этого достаточно в этих маршрутизаторах добавить в качестве адреса шлюза адрес другого конца tap-интерфейса, расположенного в Ubuntu. Для избежаний коллизий и накладок с другими работающими студентами в маршрутизаторе с номером x можно добавить статический маршрут только на tap-интерфейсы своих маршрутизаторов

Ip route add dst-address=10.D.0.0/16 gateway=10.D.x.2.

Тогда для каждого студента D моделью Интернета является та совокупность маршрутизаторов, у которых настроен статический маршрут на подсеть 10.D.0.0/16.

## DHCP и преобразование адресов источника

Корпорация имеет локальную сеть, организованную с помощью коммутатора 2-го уровня (свича) R0 и состоящую из компьютеров R2, R3 и маршрутизатора R1 с выходом в Интернет.

Корпорации выделен один внешний Интернет-адрес 10.0.1.1/24. Для внутренних адресов решено использовать IP-сеть 192.168.0.0/24. Эта сеть не видна в Интернете. Адрес 192.168.0.1 из этой сети назначен на маршрутизатора R1. Необходимо обеспечить для компьютеров R2 и R3 выход в Интернет и автоматическое назначение адресов из сети 192.168.0.0/24 с помощью протокола DHCP (Dynamic Host Configuration Protocol). Соберите топологию dhcp, согласно Рис.1. Облако на рисунке имеет чисто декоративный характер, изображая Интернет. Дайте сетевым устройствам имена

**system identity set name=**

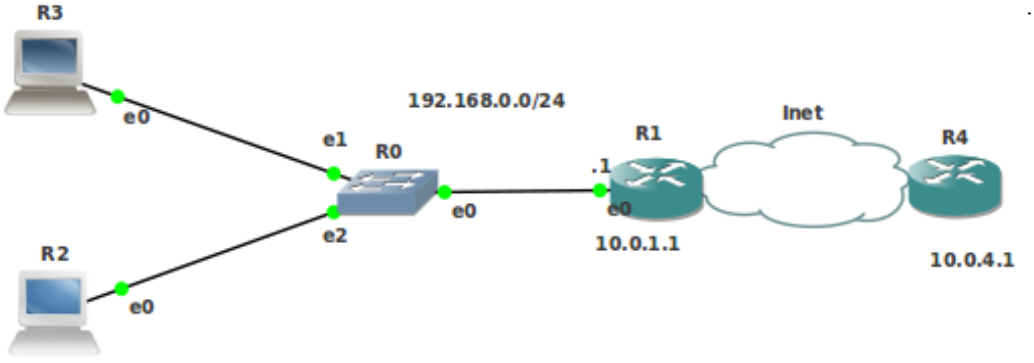



**Рис.1**

Превратим R0 в свич

[admin@R4] >**interface bridge add**

[admin@R4] >**interface bridge port add bridge= bridge1 interface=ether1**
[admin@R4] >**interface bridge port add bridge= bridge1 interface=ether2**
[admin@R4] >**interface bridge port add bridge= bridge1 interface=ether3**

Проверьте соседей. Назначьте адреса на tap-интерфейсы и адрес 192.168.0.1/24 на R1.

Сделаем модель Интернета из маршрутизаторов R1 и R4, прописав шлюзы в виде адреса другого конца tap-интерфейса

[admin@R4] > **ip ro add dst-address=10.0.0.0/16 gateway=10.0.4.2**
[admin@R1] > **ip ro add dst-address=10.0.0.0/16 gateway= 10.0.1.2**

Функции DHCP-сервера далеко не исчерпываются раздачей адресов по запросу от DHCP-клиентов. DHCP-сервер может снабдить клиентов адресами шлюза, сервера имён и другой информацией. При настройке DHCP-сервера надо указать имя интерфейса, через который он будет принимать запросы, IP-сеть и диапазон адресов из которого он будет раздавать адреса, шлюз, который будет назначен клиентам и т. д. Для быстрой настройки DHCP-сервера вводим

[admin@R1] > **ip dhcp-server setup**
Select interface to run DHCP server on
dhcp server interface: ether1- вводим имя интерфейса, смотрящего во внутреннюю сеть.
Select network for DHCP addresses
dhcp address space: 192.168.0.0/24 - соглашаемся с предлагаемой внутренней IP-сетью.
Select gateway for given network
gateway for dhcp network: 192.168.0.1 - соглашаемся с предлагаемым шлюзом во внешний мир.
Select pool of ip addresses given out by DHCP server
addresses to give out: 192.168.0.2-192.168.0.254- соглашаемся с предлагаемым диапазоном IP-адресов.
Select DNS servers
dns servers: **192.168.0.1** - вводим любой адрес, так как мы не рассматртваем вопросы DNS.
Select lease time
lease time: 3d - соглашаемся с предлагаемым временем аренды алресов.

Чтобы для R2 и R3 автоматически назначались адреса и шлюзы их надо сделать DHCP-клиентами, указывая интерфейс через который они будут посылать DHCP-запросы

[admin@R2] > **ip dhcp-client add interface=ether1 disabled=no**
[admin@R3] > **ip dhcp-client add interface=ether1 disabled=no**

Проверим, что R2 и R3 получили адрес

[admin@R2] > **ip ad pr**
**1 D 192.168.0.254/24 192.168.0.0 ether1**
[admin@R3] > **ip ad pr**
**1 D 192.168.0.253/24 192.168.0.0 ether1**

и шлюз во внешний мир

[admin@R2] > **ip ro pr**
**0 ADS 0.0.0.0/0 192.168.0.1 1**
[admin@R3] > **ip ro pr**
**0 ADS 0.0.0.0/0 192.168.0.1 1**

Проверим, что R1, R2 и R3 видят друг друга по IP, например

[admin@R3] > **ping 192.168.0.253**
**HOST SIZE TTL TIME STATUS**
**192.168.0.253 56 64 9ms**

Попытка пропинговать из R2 или R3 внешний маршрутизатор R4 закончится неудачей. На R2 и R3 есть маршрут в сорону R4. Пакет пинга дойдёт до назначения. Адреса источника и приёмника поменяются местами. Пакет не сумеет вернутся, так как у Ubuntu нет маршрутов на сеть 192.168.0.0/24 .

Добьемся, чтобы R2 и R3 видели маршрутизатор R4. Для этого у каждого исходящего из R2 (R3) пакета надо поменять адрес источника на адрес, который есть в таблицах маршрутов Ubuntu, то есть на адрес tap-интерфейса ether7 R2 10.0.1.1. Каждая такая подмена записывается в таблицу преобразований. Когда пакет возвращается, производится поиск преобразования в таблице и осуществляется обратная замена адреса. Принимаются меры для обеспечения уникальности поиска в таблице преобразований. Так для TCP/UDP-пакета порт источника заменяется на порт из списка свободных. Настраиваем на R1 преобразование адресов источника

[admin@R1] > **ip firewall nat add chain=srcnat action=masquerade out-interface=ether7**

Такое преобразование адресов называется преобразованием адреса источника. Теперь и R2 и R3 видят внешний мир, то есть адрес 10.0.4.1
маршрутизатора R4

[admin@R2] > ping 10.0.4.1

[admin@R3] > ping 10.0.4.1

## Преобразование адресов приёмника

Корпорация имеет два сервера R0 и R1 в локальной сети одного филиала и два сервера R5 и R7 в локальной сети второго филиала. Первый филиал имеет выход в Интернет через маршрутизатор R3, второй через маршрутизатор R4. Адреса локальных сетей филиалов не видны в Интернете. Необходимо так настроить преобразование адресов, чтобы сервера обеих филиалов видели друг друга. Соберите топологию nat, согласно Рис.2.

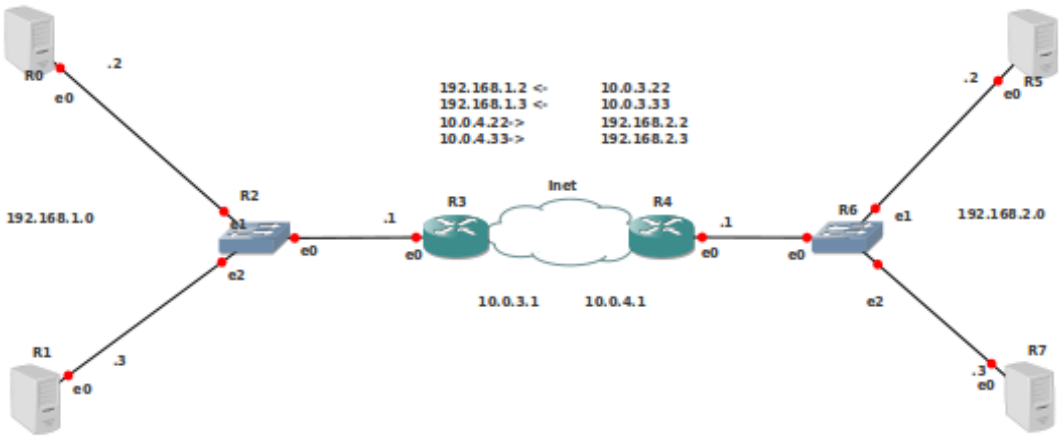



**Рис.2**.

Назначим имена согласно рисунку **Рис.2**. Настроим локальные сети филиалов. Превратите R2 и R6 в свичи

**interface bridge add**
**interface bridge port add bridge= bridge1 interface=ether1**
**interface bridge port add bridge= bridge1 interface=ether2**

**interface bridge port add bridge= bridge1 interface=ether3**

Назначим адреса. Специально не используем протокол DHCP, чтобы показать его преимущества

[admin@R0] > **ip address add address=192.168.1.2/24 interface=ether1**
[admin@R1] > **ip address add address=192.168.1.3/24 interface=ether1**
[admin@R3] > **ip address add address=192.168.1.1/24 interface=ether1**
[admin@R5] > **ip address add address=192.168.2.2/24 interface=ether1**
[admin@R7] > **ip address add address=192.168.2.3/24 interface=ether1**
[admin@R4] > **ip address add address=192.168.2.1/24 interface=ether1**

В каждой локальной сети пропингуйте маршрутизатор из серверов. Назначим для серверов маршрут по умолчанию на маршрутизатор, имеющий выход в Интернет

[admin@R0] > **ip route add gateway=192.168.1.1**
[admin@R1] > **ip route add gateway=192.168.1.1**
[admin@R5] > **ip route add gateway=192.168.2.1**
[admin@R7] > **ip route add gateway=192.168.2.1**

Маршрутизаторы R3 и R4 имеют маршрут по умолчанию направленный в интернет. В качестве адреса шлюза используем адрес другого конца tap-интерфейса, расположенного в Ubuntu

[admin@R3] > **ip route add dst-address=10.0.0.0/16 gateway=10.0.3.2**
[admin@R4] > **ip route add dst-address=10.0.0.0/16 gateway=10.0.4.2**

Пропингуйте R3 и R4 между собой. Попытка пропинговать сервера одного филиала из серверов другого филиала закончится неудачей. Сеть Ubuntu, встретив такой пакет пинга не будет знать, что с ним делать – у неё нет маршрутов на сети 192.168.1.0/24 и 192.168.2.0/24. Попытка пропинговать маршрутизатор чужого филиала также не удастся. Пакет пинга дойдёт до назначения. Адреса источника и приёмника поменяются местами. Пакет не сумеет вернутся, так как у Ubuntu нет маршрутов на сети 192.168.1.0/24 и 192.168.2.0/24..

В начале добьемся, чтобы сервера одного филиала видели маршрутизатор второго филиала. Для этого у каждого исходящего из R3 (R4) пакета поменяем адрес источника на адрес, который есть в таблицах маршрутов Ubuntu, то есть на адрес tap-интерфейса ether7. Настраиваем

[admin@R3] > **/ip firewall nat add chain=srcnat action=masquerade out-interface=ether7**
[admin@R4] > **/ip firewall nat add chain=srcnat action=masquerade out-interface=ether7**

Теперь сервера одного филиала могут пинговать маршрутизатор второго филиала. Проверьте.

Рассмотрим преобразование адреса назначения. Назначим на tap-интерфейсы дополнительные адреса

[admin@R3] > **ip address add address=10.0.3.22/24 interface=ether7**
[admin@R3] > **ip address add address=10.0.3.33/24 interface=ether7**
[admin@R4] > **ip address add address=10.0.4.22/24 interface=ether7**
[admin@R4] > **ip address add address=10.0.4.33/24 interface=ether7**

Для устранения неоднозначности определим предпочтительный исходящий адрес для маршрутизации

[admin@R3] > **ip route set 0 pref-src= 10.0.3.1**
[admin@R3] > **ip route set 0 pref-src= 10.0.4.1**

Введём правила преобразования адресов

[admin@R3] > **/ip firewall nat add chain=dstnat action=dst-nat to-addresses=192.168.1.2 dst-address=10.0.3.22**
[admin@R3] > **/ip firewall nat add chain=dstnat action=dst-nat to-addresses=192.168.1.3 dst-address=10.0.3.33**
[admin@R4] > **/ip firewall nat add chain=dstnat action=dst-nat to-addresses=192.168.2.2 dst-address=10.0.4.22**
[admin@R3] > **/ip firewall nat add chain=dstnat action=dst-nat to-addresses=192.168.2.3 dst-address=10.0.4.33**

Проверим работу. На сервере R0 или R1 для входа через телнет на R5 (192.168.2.2) вводим

**system telnet 10.0.4.22**

видим

[admin@R5] >

Попали правильно на 192.168.2.2. Выходим- CtrlD

Для входа через телнет на R7 (192.168.2.3) вводим

**system telnet 10.0.4.33**

видим

[admin@R7] >

Попали правильно на 192.168.2.3. Выходим- CtrlD

На сервере R5 или R7 для входа через телнет на R0 (192.168.1.2) вводим

**system telnet 10.0.3.22**

видим

[admin@R0] >

Попали правильно на 192.168.1.2

Выходим- CtrlD

Для входа через телнет на R1 (192.168.1.3) вводим

**system telnet 10.0.1.33**

видим

[admin@R1] >

Попали правильно на 192.168.1.3. Выходим- CtrlD

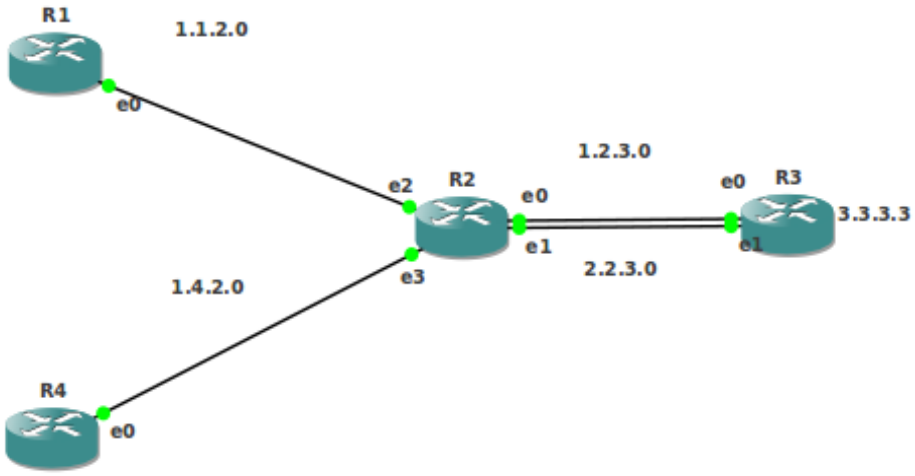


**Рис.3**

## Балансировка нагрузки

Собираем топологию lb1 согласно рисунку **Рис.3**.. Последняя цифра адреса равна номеру маршрутизатора. Рассмотрим 2 случая.

1. Хочу, чтобы трафик от R1 всегда шел от R2 e0 к R3 e0, а трафик от R4 всегда шел от R2 e1 к R3 e1

Назначаем адреса и шлюзы

R2

[admin@R2] >**ip ad ad address=1.2.3.2/24 interface=ether1**
[admin@R2] >**ip ad ad address=2.2.3.2/24 interface=ether2**
[admin@R2] >**ip ad ad address=1.1.2.2/24 interface=ether3**
[admin@R2] >**ip ad ad address=1.4.2.2/24 interface=ether4**

R1

[admin@R1] >**ip ad ad address=1.1.2.1/24 interface=ether1**
[admin@R1] >**ip ro add g= 1.1.2.2**

R4

[admin@R4] >**ip ad ad address=1.4.2.1/24 interface=ether1**
[admin@R4] >**ip ro add g= 1.4.2.2**

R3

[admin@R3] >**ip ad ad address=1.2.3.3/24 interface=ether1**
[admin@R3] >**ip ad ad address= 2.2.3.3/24 interface=ether2**

Явно пропишем на R3 желаемые маршруты для обратного трафика в сторону R2

Через e0 (ether1 )

[admin@R3] >**ip ro add dst-address=1.1.2.0/24 gateway=1.2.3.2**

Через e1 (ether2)

[admin@R3] >**ip ro add dst-address=1.4.2.0/24 gateway=2.2.3.2**

Пометим на R2 пакеты метками **1.1.2.0in** либо **1.4.2.0in** в зависимости от адреса источника

[admin@R2] >**ip firewall mangle add chain=prerouting action=mark-routing new-routing-mark=1.1.2.0in src-address=1.1.2.0/24**
[admin@R2] >**ip firewall mangle add chain=prerouting action=mark-routing new-routing-mark=1.4.2.0in src-address=1.4.2.0/24**

В зависимости от полученной метки **1.1.2.0in** либо **1.4.2.0in** пакеты направляются на различные адреса и различные интерфейсы:

[admin@R2] >**ip ro add gateway=1.2.3.3%ether1 routing-mark=1.1.2.0in**
[admin@R2] >**ip ro add gateway==2.2.3.3%ether2 routing-mark=1.4.2.0in**

Запустим Winbox на R1 и выполним тест полосы пропускания в сторону адреса 2.2.2.3 на R3 (Рис.4). На Рис.5 видим,что весь трафик на R3 пошёл через ether1, то есть по пути от R2 e0 к R3 e0.

Запустим Winbox на R4 и выполним тест полосы пропускания в сторону адреса 1.2.3.3 на R3 (Рис.6). На Рис.7 видим, что весь трафик на R3 пошёл через ether2, то есть по пути от R2 e1 к R3 e1.

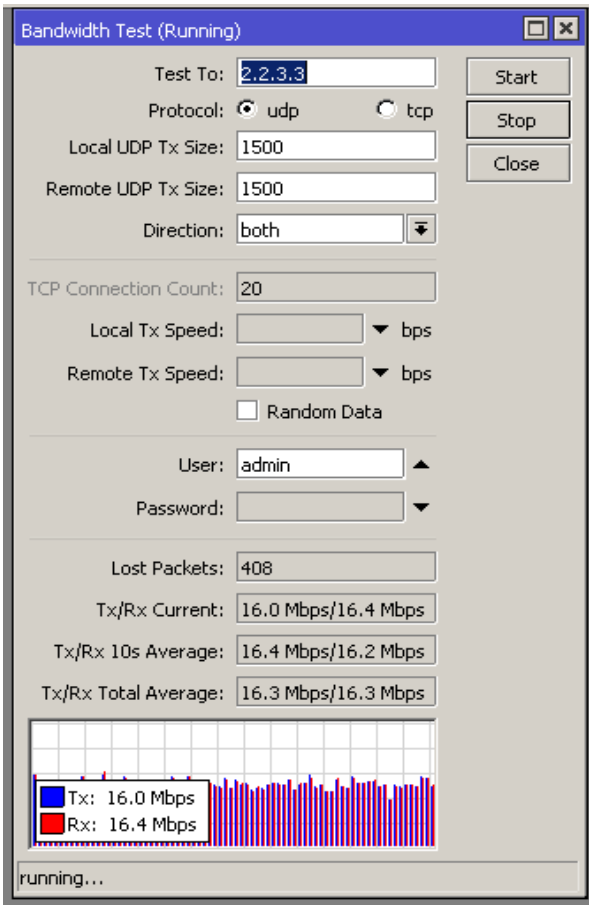

Bandwidth Test (Running)
Test To: 2.2.3.3
Protocol: udp tcp
Local UDP Tx Size: 1500
Remote UDP Tx Size: 1500
Direction: both
TCP Connection Count: 20
Local Tx Speed: bps
Remote Tx Speed: bps
Random Data
User: admin
Password:
Lost Packets: 408
Tx/Rx Current: 16.0 Mbps/16.4 Mbps
Tx/Rx 10s Average: 16.4 Mbps/16.2 Mbps
Tx/Rx Total Average: 16.3 Mbps/16.3 Mbps
Start
Stop
Close
Tx: 16.0 Mbps
Rx: 16.4 Mbps
running...

Рис.4

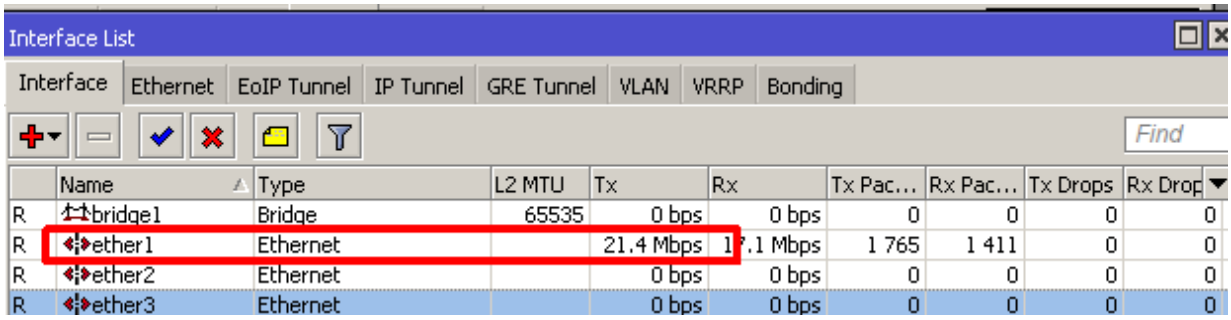

Interface List
Interface
Ethernet
EoIP Tunnel
IP Tunnel
GRE Tunnel
VLAN
VRRP
Bonding
Find
Name
Type
L2 MTU
Tx
Rx
Tx Pac...
Rx Pac...
Tx Drops
Rx Drop
R bridge1 Bridge 65535 0 bps 0 bps 0 0 0 0
R ether1 Ethernet 21.4 Mbps 1 765 1 411 0 0
R ether2 Ethernet 0 bps 0 bps 0 0 0 0
R ether3 Ethernet 0 bps 0 bps 0 0 0 0

Рис.5

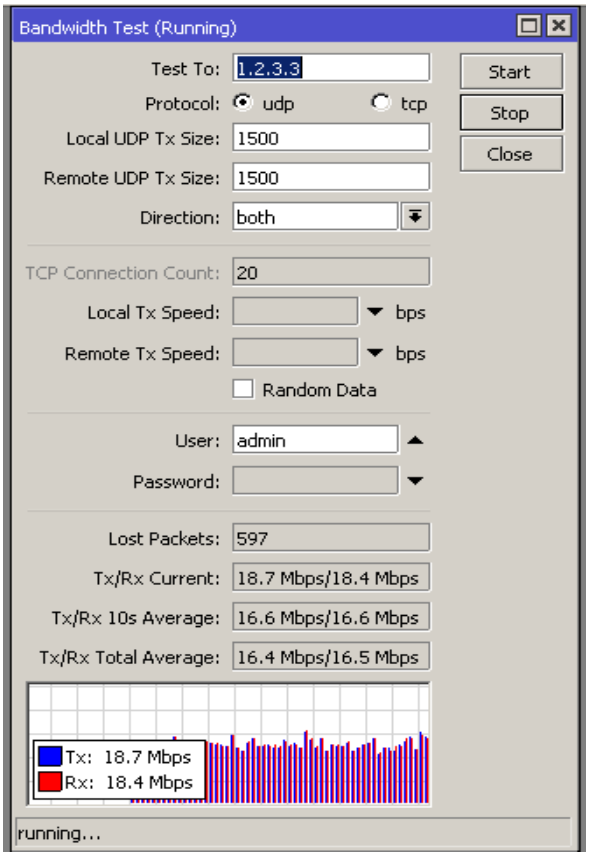



Рис.6

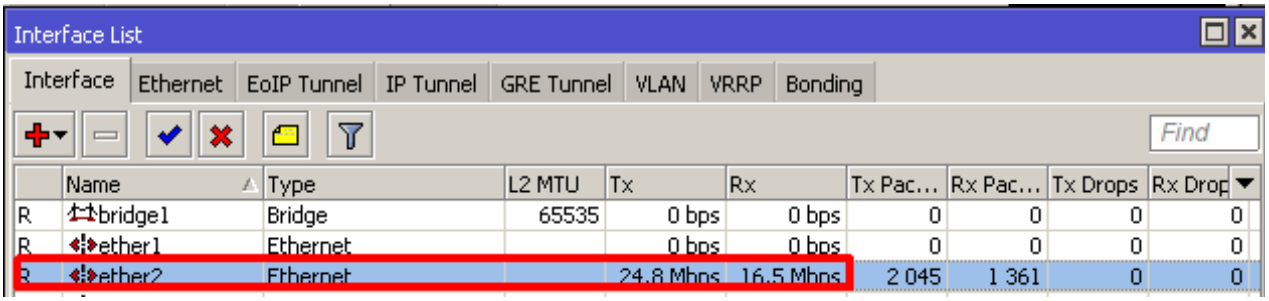



Рис.7

2. Делаем копию lb1 топологии lb.Хочу, чтобы трафик от R1 и R4 равномерно распределялся по двум путям от R2 e0 к R3 e0 и R2 e1 к R3 e1. Уберём манглы на R2

[admin@R2] >**ip firewall mangle print**

[admin@R2] >**ip firewall mangle rem**

и сделаем новые
[admin@R2] >**ip firewall mangle add chain=prerouting action=mark-routing new-routing-mark=gw1 passthrough=no nth=2,1**
[admin@R2] >**ip firewall mangle add chain=prerouting action=mark-routing new-routing-mark=gw2 passthrough=no nth=2,2**

Пакеты получаюь номера 1,2,1,2,1,2. Каждый первый получает маркер gw1, каждый второй получает маркер gw2. Уберём старые маршруты на R2 (которые с **routing-mark)**
[admin@R3] >**ip ro pri**
**[admin@R3] >ip ro rem**
и добавляем маршрутизацию
[admin@R2] >**ip route add gateway=2.2.3.3 routing-mark=gw1**
[admin@R2] >**ip route add gateway=1.2.3.3 routing-mark=gw2**

Уберём на R3 старые 2 маршрута в сторону R2
[admin@R3] >**ip ro pri**
[admin@R3] >**ip ro rem**
и добавляем сразу два шлюза
[admin@R3] >**ip ro add gateway=1.2.3.2,2.2.3.2**

Запуская на R1 или R4 тест полосы пропускания в сторону адреса 1.2.3.3 или 2.2.3.3 на R3. На Рис. 8 видим одновременную загрузку двух интерфейсов.

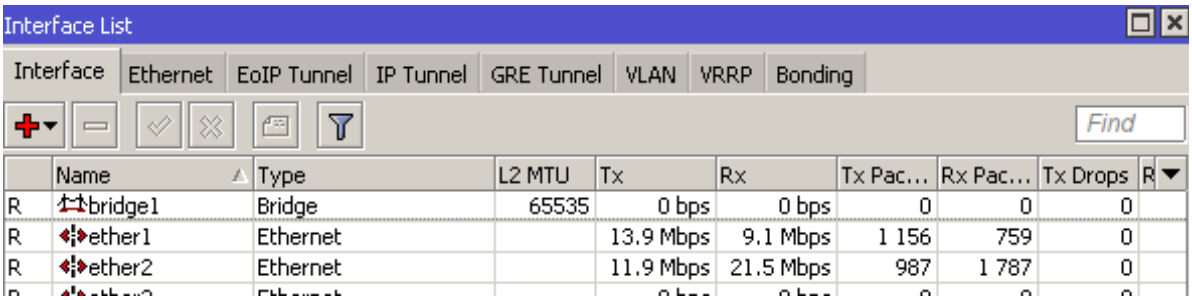



**Рис.8**

Для сдачи лабы необходимо предоставить работающие топологии dhcp, nat, lb и lb1.

# 5. Мосты. EoIP - эсернет через IP . VPN уровня 2

**Мосты**
**EoIP. VPN уровня 2**
**VPN уровня 2 через NAT**

## Мосты

Mikrotik RouterOs поддерживает объединение эсернет-портов в мост. Как уже указывалось, объединяя несколько эсернет-портов в мост, мы получим коммутатор второго уровня или свич на этих портах.

Tap-интерфейс на стороне Ubuntu лежит в мосту. Ubuntu и RouterOs обмениваются информацией о мостах. Загрузите в GNS копию bridge1 топологии template и стартуйте только маршрутизатор R0. Введём в Ubuntu (студент 7)

```
student7@hu1104:~$ brctl showmacs B700
port no mac addr                is local?       ageing timer
  1     00:00:ab:73:50:00       no                50.40
  1     52:54:00:12:34:5c       no                50.40
  1     be:79:1f:04:47:4c       yes                0.00
```

Посмотрим в Ubuntu MAC-адрес моста **B700**

```
student7@hu1104:~$ ifconfig B700
br0       Link encap:Ethernet  HWaddr be:79:1f:04:47:4c
          inet addr:10.0.0.2  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::bc79:1fff:fe04:474c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:278 errors:0 dropped:0 overruns:0 frame:0
          TX packets:179 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27403 (27.4 KB)  TX bytes:18500 (18.5 KB)
```

Посмотрим MAC-адреса в маршрутизаторе R0 (студент 7)

```
[admin@R0] > int et pr
 0 R  ether1                    1500  00:AA:00:61:15:00 enabled
 6 R  ether7                    1500  52:54:00:12:34:5C enabled
```

Видим, что Ubuntu видит MAC-адрес **52:54:00:12:34:5c** ether7 в маршрутизаторе R0.

Соединим интерфейсы e0 маршрутизаторов R0 и R1. Запустим топологию. Посмотрим соседей

```
[admin@R0] > ip neighbor pr
[admin@R1] > ip neighbor pr
```

Посмотрим MAC-адреса в маршрутизаторе R1 (студент 7)

```
[admin@R1] > int et pr
 0 R  ether1                    1500  00:AA:00:A2:FE:00 enabled
 6 R  ether7                    1500  52:54:00:12:34:5C enabled
```

Посмотрим в Ubuntu MAC-адреса моста **B700**

```
student7@hu1104:~$ brctl showmacs B700
port no mac addr                is local?       ageing timer
  1     00:aa:00:61:15:00       no                 8.71
  1     00:aa:00:a2:fe:00       no                 7.14
  1     52:54:00:12:34:5c       no                 7.13
```

```
 1     8a:d6:19:c9:ae:29     no          96.80
 1     be:79:1f:04:47:4c     yes          0.00
```

Мост **B700** увидел интерфейсы ether1 R0 (00:AA:00:61:15:00) и ether1 R1 (00:AA:00:A2:FE:00**)**

```
p@hu1104:~$ brctl showmacs B701
port no mac addr                is local?       ageing timer
  1     00:aa:00:61:15:00       no              21.25
  1     00:aa:00:a2:fe:00       no              19.68
  1     52:54:00:12:34:5c       no              19.68
  1     8a:d6:19:c9:ae:29       yes              0.00
  1     be:79:1f:04:47:4c       no             113.20
```

Мост **B701** увидел интерфейсы ether1 R0 (00:AA:00:61:15:00) и ether1 R1 (00:AA:00:A2:FE:00**)** и мосты Ubuntu увидели друг друга (**8a:d6:19:c9:ae:29 be:79:1f:04:47:4c)**.

Для более сложных топологий маршрутизаторы начинают взаимодействовать с мостами Ubuntu, и объём информации увеличивается. Для сокращения объёма информации временно откажемся от tap-интерфейсов и сократим число эсернет-адаптеров в каждом маршрутизаторе до реально испльзуемого количества портов. Временно откажемся от шаблона и создадим топологию bridge2, приведённую на рисунке Рис.1. До установки связей назначьте в контекстном меню для R0 и R1 по одной сетевой карточке (NIC), а для Sw1 – две. Запустите топологию. Подводя мышь к маршрутизаторам, узнаём порты консоли. Открываем 3 таба в консоли Ubuntu и для открытия консолей Mikrotik вводим в табах

telnet 127.0.0.1 порт консоли

Назначаем имена

```
[admin@MikroTik] > system identity set name= R0
[admin@MikroTik] > system identity set name= R1
[admin@MikroTik] > sy id s name=Sw1
```

Проверим соседей

```
[admin@Sw1] > ip neighbor print
 # INTERFACE ADDRESS       MAC-ADDRESS       IDENTITY  VERSION   BOARD
 0 ether1                  00:AA:00:CF:53:00 R0        5.2       x86
 1 ether2                  00:AA:00:3F:C9:00 R2        5.2       x86
```

Объединим интерфейсы Sw1 в мост

```
[admin@Sw1] > interface bridge add
[admin@Sw1] > interface bridge port add bridge=bridge1 interface=ether1
[admin@Sw1] > interface bridge port add bridge=bridge1 interface=ether2
```

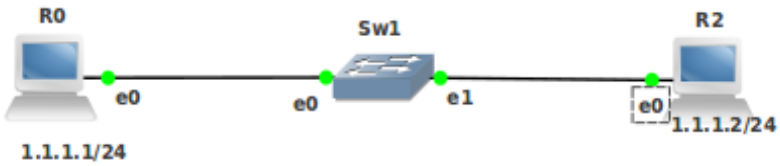



Рис.1.

Снова проверим соседей. Теперь все видят всех

```
[admin@R0] > ip neighbor print
 # INTERFACE ADDRESS       MAC-ADDRESS       IDENTITY  VERSION   BOARD
```

```
 0 ether1              00:AA:00:E1:33:00 Sw1       5.2       x86
 1 ether1              00:AA:00:3F:C9:00 R2        5.2       x86
[admin@Sw1] > ip neighbor print
 # INTERFACE ADDRESS        MAC-ADDRESS      IDENTITY   VERSION    BOARD
 0 bridge1                 00:AA:00:CF:53:00 R0        5.2       x86
 1 bridge1                 00:AA:00:3F:C9:00 R2        5.2       x86
[admin@R2] > ip neighbor pr
 # INTERFACE ADDRESS        MAC-ADDRESS      IDENTITY   VERSION    BOARD
 0 ether1                  00:AA:00:E1:33:00 Sw1       5.2       x86
 1 ether1                  00:AA:00:CF:53:00 R0        5.2       x86
```

Если интерфейс подключается к мосту, то мост считывает с этого интерфейса все MAC-адреса доступные через этот интерфейс и запоминает в таблице. По этой таблице мост направляет пакеты.

Посмотрим MAC-адреса

```
[admin@R0] > interface ethernet print
 0 R  ether1                       1500  00:AA:00:CF:53:00 enabled
[admin@R2] > interface ethernet print
0 R  ether1                       1500  00:AA:00:3F:C9:00 enabled
```

Выведем таблицу, показывающую на какой интерфейс направлять пакет с определённым MAC-адресом

```
[admin@Sw1] > interface bridge host print
Flags: L - local, E - external-fdb
  BRIDGE       MAC-ADDRESS       ON-INTERFACE       AGE
  bridge1      00:AA:00:3F:C9:00 ether2             28s       MAC-адрес R1
  bridge1      00:AA:00:CF:53:00 ether1             30s       MAC-адрес R0
L bridge1      00:AA:00:E1:33:00 ether1             0s
L bridge1      00:AA:00:E1:33:01 ether2             0s
```

Все устройства могут видеть друг друга по эсернету. Для полноты картины назначьте адреса, указанные на рисунке и пропингуйте. Пинги не могут не пойти.

Создадим топологию bridge2, приведённую на рисунке. До установки связей назначьте в контекстном меню для R0 и R3 по одной сетевой карточке (NIC), а для Sw1 и Sw2 – две. Запустите топологию. Подводя мышь к маршрутизаторам, узнаём порты консоли. Открываем 4 таба в консоли Ubuntu и для открытия консолей Mikrotik вводим в табах
telnet 127.0.0.1 порт консоли

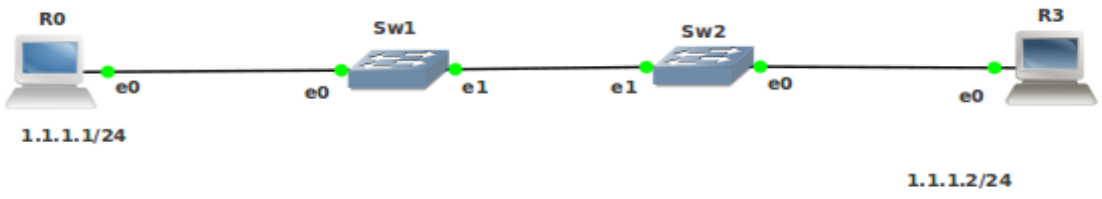



Рис.2.

Назначаем имена. Объединим интерфейсы в Sw1 и в Sw2 в мост. Посмотрим MAC-адреса

```
[admin@R0] > int ethernet print
 0 R  ether1                    1500  00:AA:00:CF:53:00 enabled
[admin@R3] > int ethernet print
 0 R  ether1                    1500  00:AA:00:35:2D:00 enabled
```

Выведем таблицы, показывающие мостам на какой интерфейс направлять пакет с определённым MAC-адресом

```
[admin@Sw1] > interface bridge host pr
Flags: L - local, E - external-fdb
  BRIDGE        MAC-ADDRESS       ON-INTERFACE        AGE
  bridge1       00:AA:00:35:2D:00 ether2              4s          MAC-адрес  R3
  bridge1       00:AA:00:63:2B:00 ether2              5s
  bridge1       00:AA:00:CF:53:00 ether1              8s          MAC-адрес  R0
L bridge1       00:AA:00:E1:33:00 ether1              0s
L bridge1       00:AA:00:E1:33:01 ether2              0s

[admin@Sw2] > interface bridge host pr
Flags: L - local, E - external-fdb
  BRIDGE        MAC-ADDRESS       ON-INTERFACE        AGE
  bridge1       00:AA:00:35:2D:00 ether1              8s          MAC-адрес  R3
L bridge1       00:AA:00:63:2B:00 ether1              0s
L bridge1       00:AA:00:63:2B:01 ether2              0s
  bridge1       00:AA:00:CF:53:00 ether2              12s         MAC-адрес  R0
  bridge1       00:AA:00:E1:33:01 ether2              9s
```

Все устройства могут видеть друг друга по эсернету. Для полноты картины назначьте адреса, указанные на рисунке и пропингуйте. Пинги не могут не пойти.

Можно конструировать сколь угодо сложную топологию из мостов. Если предвидятся циклы (пели), то следует запустить в мостах протокол покрывающего дерева STP или его разновидность RSTP, например

**interface bridge set 0 protocol-mode=rstp**

Этот вопрос здесь не обсуждается

Обычно эсернет пакеты ходят по проводам или оптическому волокну. Ничего не мешает им ходить внутри IP-пакетов.

## EoIP

Туннелирование эсернет через IP (Ethernet over IP - EoIP) это протокол MikroTik RouterOS, который создаёт эсернет-туннель между двумя маршрутизаторами поверх IP-соединения. EoIP-тунель может работать через любое соединение, способное передавать IP-пакеты: IPIP-туннель, PPP и т.д.

Если в двух маршрутизаторах настроена поддержка EoIP, то эсернет-трафик (все эсернет протоколы) пойдут через интерфейс EoIP также, как если бы существовали физические эмернет-интерфейсы и был проложен кабель между маршрутизаторами.

EoIP позволяет объединить через Интернет локальные сети с помошью моста и с возможностью шифрованияя трафика.

Протокол EoIP (подобно PPTP – см. Лабу 6) инкапсулирует эсернет-фреймы в GRE пакеты (протокол IP № 47) и пересылает на удалённую сторону EoIP-туннеля.

| Свойства | Описание |
|---|---|
| ***arp*** (*disabled* \| *enabled* \| *proxy-arp* \| *reply-only*; Default: **enabled**) | Режим ARP |
| ***l2mtu*** (*integer*; Default: ) | Максимальный размер блока 2 уровня на передачу. Только для чтения |
| ***mac-address*** (*MAC*; Default: ) | MAC -адрес интерфейса. Разрешённый диапазон **00:00:5E:80:00:00 - 00:00:5E:FF:FF:FF** |
| ***mtu*** (*integer*; Default: **1500**) | Максимальный размер блока 3 уровня на передачу. |
| ***name*** (*string*; Default: ) | Имя интерфейса |
| ***remote-address*** (*IP*; Default: ) | IP-адрес удалённой стороны EoIP-туннеля |
| ***local-address*** (*IP*; Default: ) | IP-адрес локальной стороны EoIP-туннеля Не обязательно |
| ***tunnel-id*** (*integer: 65536*; Default: ) | Уникальный для каждого туннеля идентификатор. Должен совпадать на обеих концах |

mtu следует установить в 1500 для избежания перефрагментации пакета внутри туннеля. Это позволяет так прозрачно объединить эсернет-сети с помощью моста, что станет возможным транспорт полноразмерных эсернет-фреймов через туннель.

При использовании мостов для EoIP-тунелей для корректной работы алгоритмов мостов строго рекомендуется устанавливать уникальный MAC-адрес для каждого туннеля. Иначе вы должны быть уверены в уникальности хостов, подсоединённых к мосту.

Сделаем копию EoIP шаблона template. Соберём топологию, указанную на рисунке Рис.3. (student0)

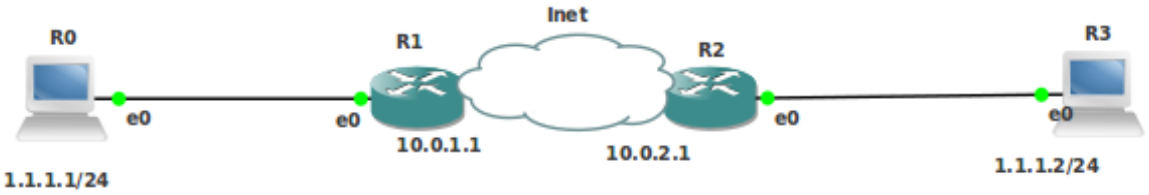



Рис.3.

Назначим имена маршрутизаторам. Проверим соседей. R1 не будет видеть R2. Это нормально. Проложим маршрут между R1 и R2 через нашу модель интернета.

[admin@R1] > **ip route add dst-address=10.0.0.0/16 gateway=10.0.1.2**
[admin@R2] > **ip route add dst-address=10.0.0.0/16 gateway=10.0.2.2**
[admin@R2] > **ping 10.0.1.1**
HOST SIZE TTL TIME STATUS
10.0.1.1 56 63 11ms
1. 56 63 0ms

Пинги пошли. R1 и R2 подсоединены к Интернету. Настроим EoIP-туннель

[admin@R1] > **int eoip add remote-address=10.0.2.1 disabled=no**

[admin@R2] > **int eoip add remote-address=10.0.1.1 disabled=no**
[admin@R2] > **int pr**
7 R eoip-tunnel1 eoip-tunnel 1500 65535

Создадим мосты и добавим в них физический эсернет интерфейс e0 (ether1) и интерфейс EoIP

[admin@R1] > **int bridge add**
[admin@R1] > **int bridge port add bridge=bridge1 interface=eoip-tunnel1**
[admin@R1] > **int bridge port add bridge=bridge1 interface=ether1**
[admin@R2] > **int bridge add**
[admin@R2] > **int bridge port add bridge=bridge1 interface=eoip-tunnel1**
[admin@R2] > **int bridge port add bridge=bridge1 interface=ether1**

Посмотрим MAC-адреса

[admin@R0] > **interface ethernet print**
0 R ether1 1500 00:AA:00:35:D1:00 enabled
[admin@R3] **>** **interface ethernet print**
0 R ether1 1500 00:AA:00:1A:6C:00 enabled

Выведем таблицы, показывающие мостам на какой интерфейс направлять пакет с определённым MAC-адресом

[admin@R1] > **int bridge host pr**
bridge1 00:AA:00:1A:6C:00 eoip-tunnel1 11s MAC-адрес R3
bridge1 00:AA:00:35:D1:00 ether1 57s MAC-адрес R0
L bridge1 00:AA:00:68:45:00 ether1 0s
bridge1 52:54:00:12:34:5C ether1 6s
bridge1 5E:0A:8C:7F:91:9E eoip-tunnel1 49s
bridge1 8A:D6:19:C9:AE:29 ether1 6s
bridge1 BE:79:1F:04:47:4C ether1 50s
bridge1 DE:56:9B:B2:04:34 eoip-tunnel1 52s
bridge1 FE:48:7B:21:B1:87 eoip-tunnel1 54s
L bridge1 FE:ED:C6:98:F4:1C eoip-tunnel1 0s
admin@R2] > **int bridge host pr**
bridge1 00:AA:00:1A:6C:00 ether1 20s MAC-адрес R3
bridge1 00:AA:00:35:D1:00 eoip-tunnel1 21s MAC-адрес R0
L bridge1 00:AA:00:D8:AB:00 ether1 0s
bridge1 52:54:00:12:34:5C ether1 2s
bridge1 5E:0A:8C:7F:91:9E ether1 6s
bridge1 8A:D6:19:C9:AE:29 eoip-tunnel1 7s
bridge1 BE:79:1F:04:47:4C eoip-tunnel1 5s
bridge1 DE:56:9B:B2:04:34 ether1 2s
L bridge1 FE:48:7B:21:B1:87 eoip-tunnel1 0s
bridge1 FE:ED:C6:98:F4:1C eoip-tunnel1 21s

Смотрим соседей

[admin@R0] > **ip neighbor print**
# INTERFACE ADDRESS MAC-ADDRESS IDENTITY VERSION BOARD
0 ether1 10.0.1.1 52:54:00:12:34:5C R1 5.2 x86
1 ether7 10.0.1.1 52:54:00:12:34:5C R1 5.2
2 ether7 00:AA:00:35:D1:00 R0 5.2 x86
3 ether7 FE:ED:C6:98:F4:1C R1 5.2 x86

```
 4 ether1              FE:ED:C6:98:F4:1C R1       5.2       x86
 5 ether7              FE:48:7B:21:B1:87 R2       5.2       x86
 6 ether1              FE:48:7B:21:B1:87 R2       5.2       x86
 7 ether7   10.0.3.1       00:AA:00:1A:6C:00 R3       5.2       x86
 8 ether1   10.0.3.1       00:AA:00:1A:6C:00 R3       5.2       x86
```

R0 видит всех

```
[admin@R3] > ip neighbor print
 # INTERFACE ADDRESS        MAC-ADDRESS      IDENTITY   VERSION    BOARD
 0 ether1   10.0.2.1       52:54:00:12:34:5C R2       5.2       x86
 1 ether7              00:AA:00:1A:6C:00 R3       5.2       x86
 2 ether7   10.0.2.1       52:54:00:12:34:5C R2       5.2
 3 ether7              00:AA:00:35:D1:00 R0       5.2       x86
 4 ether1              00:AA:00:35:D1:00 R0       5.2       x86
 5 ether7              FE:48:7B:21:B1:87 R2       5.2       x86
 6 ether1              FE:48:7B:21:B1:87 R2       5.2       x86
 7 ether7   10.0.1.1       FE:ED:C6:98:F4:1C R1       5.2       x86
 8 ether1   10.0.1.1       FE:ED:C6:98:F4:1C R1       5.2       x86
```

R1 видит всех

Есть связь на втором сетевом уровне. Окончательно убедимся в этом. Например, на R0 с помощью специальной утилиты mac-telnet соединимся по эсернет с R3, введя MAC-адрес его эсернет интерфейса ether1

[admin@R0] **/tool mac-telnet 00:AA:00:1A:6C:00**

Попадаем в R3. Возврат CtrlD

На R3 соединимся по эсернет с R0

[admin@R3] > **tool mac-telnet 00:AA:00:35:D1:00**

Попадаем в R0. Возврат CtrlD

Мы построили виртуальную частную сеть второго уровня над существующей сетью в виде модель интернета для Ubuntu.

Простота топологии нисколько не уменьшает её значимости: Вместо R0 и R1 можно исползовать свичи и к ним присоединить произвольное число сетевых устройств. VPN уовня 2 продолжит функционирование.

Для полноты картины назначьте IP-адреса для R0 и R1 согласно рисунку и пропингуйте крайние роутеры друг из друга.

**Задание.** Для топологии EoIP3, изображённой на Рис. 4, Организовать VPN уовня 2 с помощью EoIP. Не забудьте использовать разные ***tunnel-id.***

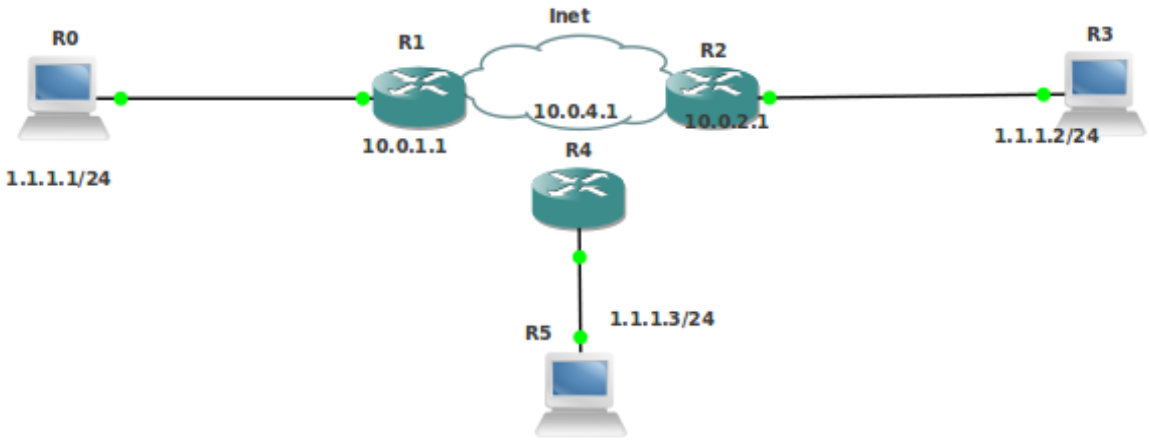



**Рис. 4**

Самостоятельно с помощью команд **int bridge host pr** посмотрите таблицы, показывающие мостам на какой интерфейс направлять пакет с определённым MAC-адресом. Убедитесь командой **ip neighbour print**, что все маршрутизаторы видит друг друга как соседа. С помощью команды **/tool mac-telnet о**кончательно убедитесь, что **е**сть связь на втором сетевом уровне. Назначаем IP-адреса на R0, R3 и R5 согласно рисунку. Пингуем маршрутизаторы. Заметим, что R0, R3 и R5 будут находится в одном домене широковещания. Это позволяет успешно функционировать широковещательным ARP-запосам: от одного ко всем (через EoIP туннели через Интернет)

## VPN уровня 2 через NAT

Рассмотрим, случай когда филиалы корпорации не имеют прямого выхода в интернет. Рассмотрим топологию EoIPNAT, изображённую на Рис. 5. Здесь R3 и R4 – маршрутизаторы Интернет провайдера. Он их конфигурирует по запросу корпорации. Проверьте соседей.

В начале согласно рисунку назначьте для R2 и R3 адреса из сети 192.168.1.0/24. Для R2 назначьте назначте шлюз 192.168.1.1. Назначьте для R4 и R6 адреса из сети 192.168.2.0/24. Для R2 назначьте шлюз 192.168.2.1. Настройте NAT для исходящих и приходящих адресов

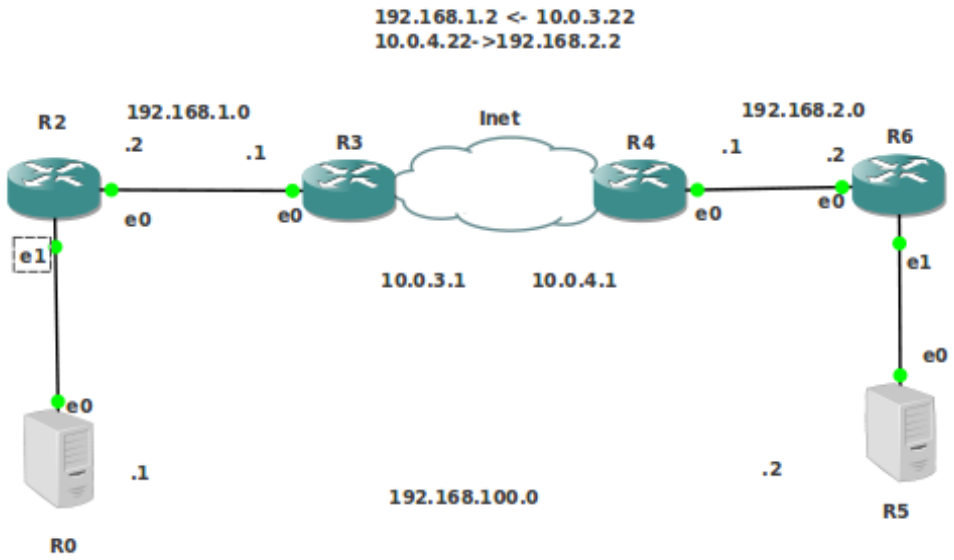



Рис. 5

.

Должны получить нечто подобное

[admin@R3] > **ip firewall nat pr**

Flags: X - disabled, I - invalid, D - dynamic

0 chain=srcnat action=masquerade out-interface=ether7

1 chain=dstnat action=dst-nat to-addresses=192.168.1.2 dst-address=10.0.3.22

[admin@R4] > **ip firewall nat print**

Flags: X - disabled, I - invalid, D - dynamic

0 chain=dstnat action=dst-nat to-addresses=192.168.2.2 dst-address=10.0.4.22

1 chain=srcnat action=masquerade out-interface=ether7

Проверьте работу NAT (см. Соотв. Лабу 4)

Настроим EoIP-туннель между R2 и R6 особым образом, учитывая AT

[admin@R2] > **int eoip add remote-address=10.0.4.22 disabled=no**

[admin@R6] > **int eoip add remote-address=10.0.3.22 disabled=no**

[admin@R6] > int pr

7 R eoip-tunnel1 eoip-tunnel 1500 65535

Самостоятельно посмотрите таблицы, показывающие мостам на какой интерфейс направлять пакет с определённым MAC-адресом

[admin@R2] > **int bridge host pr**

[admin@R6] > **int bridge host pr**

Убедитесь командой **ip neighbour print**, что крайний маршрутизатор R0 ( R5) видит R5 ( R0) как соседа. С помощью команды **/tool mac-telnet о**кончательно убедитесь, что **е**сть связь на втором сетевом уровне. Назначаем IP-адреса на R0 и R5 согласно рисунку. Пингуем из R0 R5 или наоборот.

Для сдачи работы необходимо продемонстрировать работающие топологии bridge1 bridge2 EoIP EoIP3 EoIPNAT

# 6. Построение VPN с помощью производных от PPP протоколов и OPENVPN

**PPP**
**Протоколы PPTP, SSTP и L2TP**
**OPENVPN**
**PPPoE**
**Настройка PPP, PPTP, SSTP, L2TP и OPENVPN в Routeros Mikrotik с помощью winbox**
- **Настройка PPP**
- **Настройка PPTP**
- **Настройка L2TP**
- **RSA-сертификаты**
- **Настройка SSTP**
- **Настройка OPENVPN**

**VPN уровня 2**
- **PPP**
- **PPTP**
- **L2TP**
- **SSTP**
- **OPENVPN**

**Особенности работы из командной строки**
- **PPP**
- **PPTP**
- **L2TP**
- **SSTP**
- **OPENVPN**

**Распределённый мост**
**Использование профилей пользователя.**
**VPN уровня 3**
- **Маршрутизация RIP**
- **Маршрутизация OSPF**
- **VPN 3 уровня через NAT**

**Настройка PPPoE**

## PPP

**PPP** (англ. *Point-to-Point Protocol*) — двухточечный протокол канального уровня сетевой модели OSI. Обычно используется для установления прямой связи между двумя узлами сети, причем он может обеспечить аутентификацию соединения, шифрование и сжатие данных. Используется на многих типах физических сетей: нуль-модемный кабель, телефонная линия, сотовая связь, последовательные каналы связи и т. д.

PPP представляет собой целое семейство протоколов: протокол управления линией связи (LCP - Link Control Protocol), протокол управления сетью (NCP - Network Control Protocol), протоколы аутентификации (PAP, CHAP), многоканальный протокол PPP (MLPPP), протокол сжатия CCP (compression control protocol), протокол шифрования ECP (encryption control protocol) и т. д.

PPP протокол был разработан на основе протокола HDLC и дополнен некоторыми возможностями, которые до этого встречались только в коммерческих протоколах.

PPP позволяет работать нескольким протоколам сетевого уровня на одном канале связи. Другими словами, внутри одного PPP-соединения могут передаваться потоки данных различных сетевых протоколов (IP, Novell IPX и т. д.), а также данные протоколов канального уровня локальной сети. После установления соединение для настройки каждого сетевого протокола используется протокол NCP. Он используется для согласования и определения

настроек сетевого уровня, таких как сетевой адрес или настройки сжатия.

Каждый кадр PPP всегда начинается и завершается флагом 0x7E. Затем следует байт адреса и байт управления, которые тоже всегда равны 0xFF и 0x03 соответственно. В связи с вероятностью совпадения байтов внутри блока данных с зарезервированными флагами, существует система автоматической корректировки «проблемных» данных с последующим восстановлением.

| **Флаг 0x7E** | **Адрес 0xFF** | **Управление 0x03** | **Данные** | **Контрольная сумма** | **Флаг 0x7E** |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1-1500 | 2 | 1 |

Поля «Флаг», «Адрес» и «Управление» образуют заголовок кадра HDLC. Заголовок HDLC может быть опущен и не передаваться, если PPP в процессе конфигурирования с помощью LCP договорится об этом с другой стороной.

Поле «Данные», PPP кадра, в свою очередь разбиты ещё на два поля: флаг протокола (1-2 байта), который определяет тип данных до конца кадра и сами данные.

Флаги протокола от 0x0XXX до 0x3XXX идентифицируют протоколы сетевого уровня. Например, популярному IP протоколу соответствует флаг 0x0021, а Novell IPX — 002B.

Флаги протокола от 0x4XXX до 0x7XXX идентифицируют протоколы с низким уровнем трафика.

Флаги протокола от 0x8XXX до 0xBXXX идентифицируют протокол управления сетью (NCP).

Флаги протокола от 0xCXXX до 0xEXXX идентифицируют управляющие протоколы. Например, 0xC021 обозначает, что кадр содержит данные протокола управления соединением LCP.

Фазы PPP:

**Link Dead**. Эта фаза наступает, когда связь нарушена, либо одной из сторон указали не подключаться (например, пользователь завершил модемное соединение.)

**Установки связи (Link Establishment Phase)**. В данной фазе проводится настройка линии с помощью протокола LCP. Если настройка была успешно, управление переходит в фазу аутентификации, либо в фазу Network-Layer Protocol, в зависимости от того, требуется ли аутентификация.

**Аутентификации (Authentication Phase)**. Данная фаза является необязательной. Она позволяет сторонам проверить друг друга перед установкой соединения. Если проверка успешна, управление переходит в фазу Network-Layer Protocol.

**Протокола сетевого уровня (Network-Layer Protocol Phase)**. В данной фазе вызывается NCP для желаемого протокола. Например, IPCP используется для установки IP сервисов. Передача данных также проходит в этой фазе. Закрытие сетевых протоколов тоже включается в данную фазу.

**Link Termination Phase**. Эта фаза закрывает соединение. Она вызывается в случае ошибок аутентификации, если было настолько много ошибок контрольных сумм, что обе стороны решили закрыть соединение, если соединение неожиданно оборвалось, либо если пользователь отключился. Данная фаза пытается закрыть все настолько аккуратно, насколько возможно в данных обстоятельствах.

Протокол LCP обеспечивает автоматическую настройку интерфейсов на каждом конце (например, установка размера пакетов). Так как LCP инкапсулируется в кадры PPP, то

необходимо установление первоначального соединения PPP. После установления PPP-соединения передающее и принимающее устройство обмениваются пакетами LCP для уточнения параметров соединения и специфической информации, которая потребуется при передаче данных. Далее LCP переопределяет это соединение. Устройства не могут передавать данные друг другу по сети, прежде чем LCP пакеты не определят доступность устанавливаемого соединения.

LCP протокол осуществляет: идентификацию соединяемых устройств и, вследствие этого разрешает или отклоняет установку соединения; определение приемлемого размера кадров для передачи (MTU) и приёма (MRU); ограничение по ширине канала; шифрование и аутентификацию соединения; сжатие данных; обнаружение петель маршрутизации; проверку синтаксиса и поиск ошибок в конфигурации; разрыв соединения, если какое-либо значение превышает заданный параметр.

Среди протоколов аутентификации известен CHAP (Challenge-handshake authentication protocol), который является предпочтительным для соединений с провайдерами. Всё еще иногда используется устаревший протокол PAP (Password authentication protocol ). Другим вариантом аунтентификации через PPP является Extensible Authentication Protocol (EAP).

После того, как соединение было установлено, поверх него может быть настроен сетевой уровень. Если в качестве протокола сетевого уровня используется IP, то для настройки используется протокол IPCP (Internet Protocol Control Protocol - протокол управления IP) как частный случай протокола NCP. IPCP использует тот же механизм обмена пакетами, что и LCP. Обмен пакетами IPCP не происходит до тех пор, пока PPP не завершит успешно фазу установки связи по протоколу LCP и, если требуется ли аутентификация, то и фазу аутентификации. В ходе фазы протокола сетевого уровня PPP-серверу назначается IP-адрес. Далее клиенту по протоколу IPCP передаётся назначенный ему IP-адрес.

После установки соединения, стороны, участвующие в соединении, могут быть объединены в мост. Для согласования параметров мостов используется ещё одна разновидность NCP – протокол управления мостами BCP (bridge control protocol)

## Протоколы PPTP, SSTP и L2TP

Протоколы PPTP (англ. *Point-to-Point Tunneling Protocol*) — туннельный протокол типа точка-точка, L2TP (*Layer 2 Tunneling Protocol* — протокол туннелирования второго уровня, *SSTP* (Secure Socket Tunneling Protocol – протокол безопасного туннелирования сокетов) основываются на протоколе PPP, включают его в себя полностью и отличаются от него лишь способом организации канала связи. Во всех трёх случаях для организации канала используется уже существующая связь между клиентом и сервером по протоколу IP.

В протоколе PPTP для организации канала используется протокол GRE (англ. *Generic Routing Encapsulation* — общая инкапсуляция маршрутов), кадры которого помещаются в поля данных протокола IP. Далее кадры протокола PPP инкапсулируются в кадры GRE. PPTP- сервер слушает порт 1723.

В самой начальной стадии PPTP-клиент договаривается с сервером о параметрах GRE PPP тоннеля. Далее всё берёт на себя PPP. На Рис. 1 приведён соответствующий снимок мониторинга трафика, осуществлённый с помощью программы Wireshark.

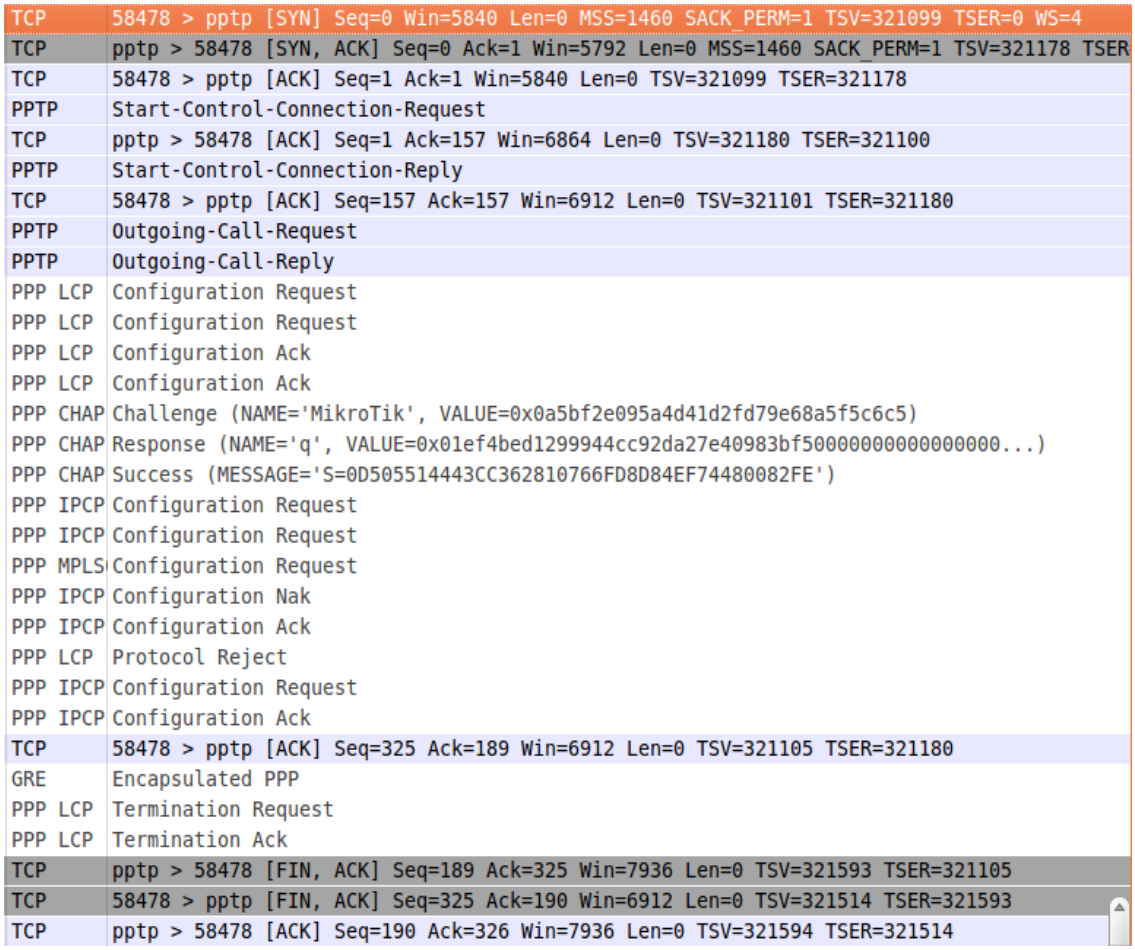



**Рис. 1.**

После организации связи и установления TCP-соединения поверх PPTP-туннеля имеет место следующая инкапсуляция протоколов: IP над GRE над PPP над IP над TCP (если не используется шифрование и сжатие). Стек протоколов можно видеть на Рис.2 .

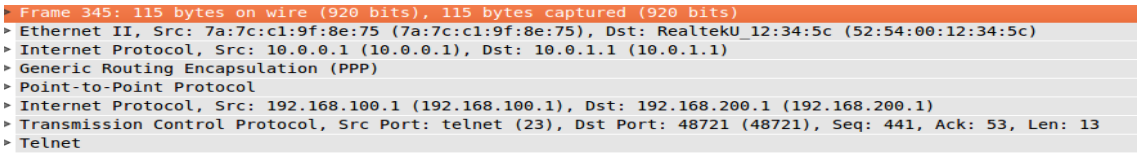



**Рис. 2.**

Протокол L2TP и для организации связи и для передачи данных использует UDP-протокол. Для установки связи используется UDP порт 1701. Далее всё берёт на себя PPP. На Рис. 3 приведён соответствующий снимок мониторинга трафика

| | | |
|---|---|---|
| L2TP | Control Message - SCCRQ | (tunnel id=0, session id=0) |
| L2TP | Control Message - SCCRP | (tunnel id=3, session id=0) |
| L2TP | Control Message - SCCCN | (tunnel id=3, session id=0) |
| L2TP | Control Message - ZLB | (tunnel id=3, session id=0) |
| L2TP | Control Message - ICRQ | (tunnel id=3, session id=0) |
| L2TP | Control Message - ICRP | (tunnel id=3, session id=1) |
| L2TP | Control Message - ICCN | (tunnel id=3, session id=1) |
| L2TP | Control Message - ZLB | (tunnel id=3, session id=0) |
| PPP LCP | Configuration Request | |
| PPP LCP | Configuration Request | |
| PPP LCP | Configuration Ack | |
| PPP LCP | Configuration Ack | |
| PPP CHAP | Challenge (NAME='MikroTik', VALUE=0x3b3e6c60de62d252a7fb9aa5a61e4f46) | |
| PPP CHAP | Response (NAME='q', VALUE=0x21553ba65ea67a48405b4700d0857c620000000000000000...) | |
| PPP CHAP | Success (MESSAGE='S=956D1FF3B9738054EEE392DB523E19FB021BDB72') | |
| PPP IPCP | Configuration Request | |
| PPP IPCP | Configuration Request | |
| PPP IPCP | Configuration Nak | |
| PPP IPCP | Configuration Ack | |
| PPP IPCP | Configuration Request | |
| PPP IPCP | Configuration Ack | |
| UDP | Source port: rrac  Destination port: rrac | |
| UDP | Source port: rrac  Destination port: rrac | |

**Рис. 3**

После организации связи и установления TCP-соединения поверх L2TP-туннеля имеет место следующая инкапсуляция протоколов: IP над UDP над L2TP над IP над TCP (если не используется шифрование и сжатие). Стек протоколов можно видеть на Рис.4 .

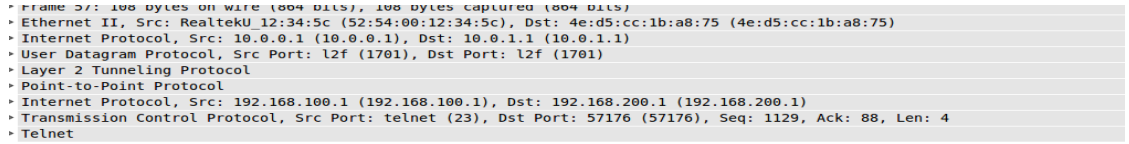



**Рис. 4**

В протоколе SSTP для организации канала используется шифрование по протоколу SSL. В SSTP установка соединения и обмен данными происходит в зашифрованном виде и мы ничего не сможем с помощью программы Wireshark наблюдать инкапсуляцию протоколов . Как правило, для установки связи по SSL протоколу клиент и сервер используют сертификаты. Ключи из сертификатов применяются для аутентификации и шифровки трафика.

**OPENVPN**

OPENVPN не использует PPP. OPENVPN для организации связи и для передачи данных использует протокол UDP или TCP. Далее в поля данных этих протоколов помещаются кадры протоколов IP (интерфейс tun) или Ethernet (интерфейс tap), которые шифруются с помощью OpenSSL. Процедура установки связи и поддержка протокола сетевого уровня являются оригинальными разработками. Для установки связи по OpenSSL

протоколу клиент и сервер используют сертификаты и/или пароли. Ключи из сертификатов применяются для аутентификации и шифровки трафика

В OPENVPN установка соединения и обмен данными происходит в зашифрованном виде и мы ничего не увидим в анализаторе пакетов

## PPPoE

***PPPoE*** *(Point-to-point protocol over Ethernet)* — протокол передачи кадров PPP через Ethernet. В основном используется устройствами DSL. Это позволяет организовать через Ethernet соединение с именем и паролем.

Клиент посылает широковещательный Ethernet-фрейм, на который должен ответить один из PPPoE-серверов. PPPoE-сервера посылает клиенту ответ. Клиент выбирает подходящий сервер и посылает ему запрос на соединение. Сервер посылает клиенту подтверждение. Между сервером и клиентом создается виртуальный канал, который идентифицируется идентификатором сессии и MAC-адресами клиента и сервера. Затем в этом канале поднимается PPP соединение.

# Настройка PPP, PPTP, SSTP, L2TP и OPENVPN в Routeros Mikrotik с помощью winbox

## Настройка PPP

Воспользовавшись шаблоном, создайте топологию PPP


**Рис.5. D=0**

Выбирая в GNS3 в контекстном меню маршрутизаторов configure -> qemu options, добавим в маршрутизаторы последовательные порты, соединённые друг с другом с помощью UDP

R0

```
options = -net nic -net tap,script=no,downscript=no,ifname=tapD00
-serial udp::D555@:D556
```

R1

```
options = -net nic -net tap,script=no,downscript=no,ifname=tapD01
-serial udp::D556@:D555
```

D – ваш номер. Имеем модель соединения двух устройств с помощью последовательного канала связи или, если конкретно, то с помощью нуль-моема. Запустим топологию.

Наберём в консоли ubuntu

**netstat –u –p**

```
student0@u11lib:~/projects/SSTProute2$ netstat -u -p
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 localhost:4555          localhost:4556          ESTABLISHED 12835/qemu
udp        0      0 localhost:4556          localhost:4555          ESTABLISHED 12832/qemu
```

**Рис.6. D=4**

видим на Рис.6, что два экземпляра программы qemu (внутри qemu запущен маршрутизатор) установили между собой два UDP-соединения, что обеспечило между ними двустороннюю дуплексную связь. Это соединение моделирует связь двух маршрутизаторов, путём соединения их последовательных портов через ноль-модем. Последовательный порт добавляется в маршрутизатор, строкам, которые мы добавили в контектном меню configure -> qemu options (D=4)

```
-serial udp::4555@:4556
-serial udp::4555@:4556
```

В консоли R0 вводим

[admin@R0] > **port print detail**

```
0  name="serial0" used-by="Serial Console" channels=1 baud-rate=9600
   data-bits=8 parity=none stop-bits=1 flow-control=none
1  name="serial1" used-by="" channels=1 baud-rate=9600 data-bits=8
   parity=none stop-bits=1 flow-control=none
```

Видим, что появился новый последовательный порт serial1, который работает на скорости 9600 бит в секунду, с длинной байта в 8 бит, без контроля чётности, с одним стоп-битом и без управления передачей. Маршрутизатор Mikrotik имеет встроенный последовательный порт serial0, который использует его консоль.

Запустим в Ubuntu два экземпляра программы winbox для конфигурации маршрутизатора R0(connect to 10.D.0.1) и R1(connect to 10.D.1.1).

Итак, между маршрутизаторами R0 и R1 установлен последовательный канал связи. Осуществим через этот канал соединение маршрутизаторов по проколу PPP. Пусть R0 будет PPP-сервером, а R1 - PPP-клиентом.

Mikrotik RouterOS обеспечивает масштабируемую аутентификацию, авторизацию и учёт пользователей AAA(Authentication, Athorization and Accounting (AAA)). Локальная аутентификация выполняется с помощью базы данных профилей и базы данных пользователей. Действительная конфигурация для данного пользователя составляется из записи базы данных пользователей, соответствующей записи из базы данных профилей и записи из базы данных профилей, которая является записью по умолчанию для той службы, к которой пытается подсоединиться пользователь. Запись по умолчанию из базы данных профилей имеет самый низкий приоритет. Запись из записи базы данных пользователей имеет наивысший приоритет, за некоторыми исключениями, касающимися адресации.

В winbox R0 добавим PPP-пользователя q с паролем q (PPP->Secrets +). Добавим PPP-сервер (PPP->Interfaces + PPP-server), связав его с новым последовательным портом `serial в режиме ноль-модема` См. Рис.7.

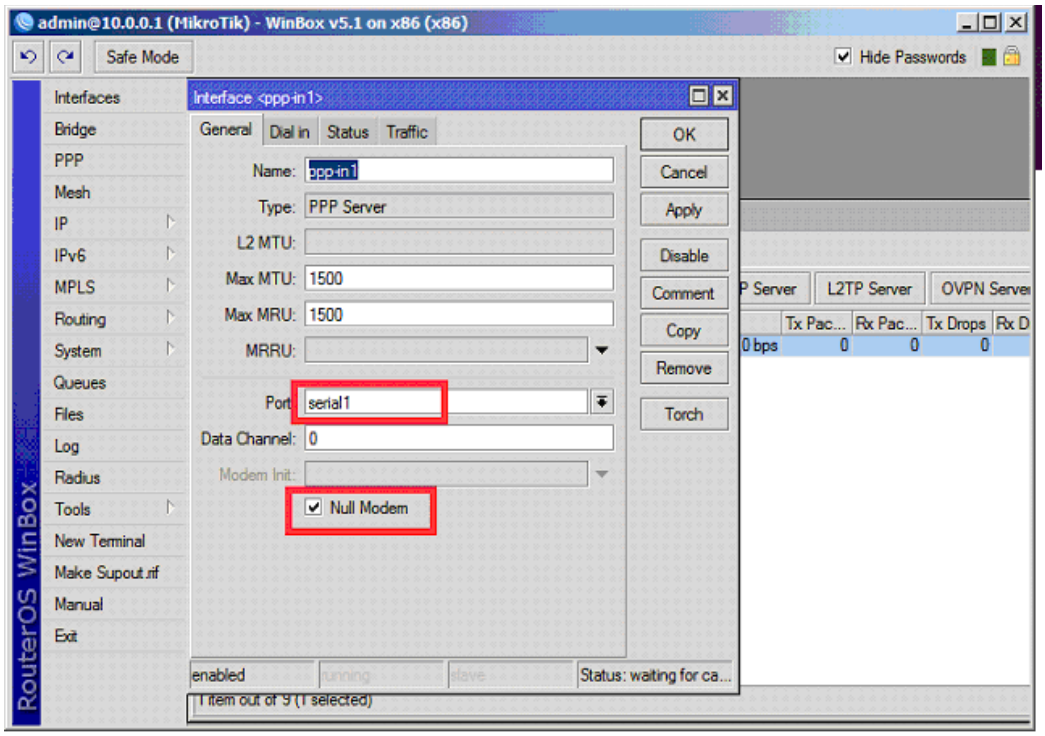



**Рис.7. PPP-сервер**

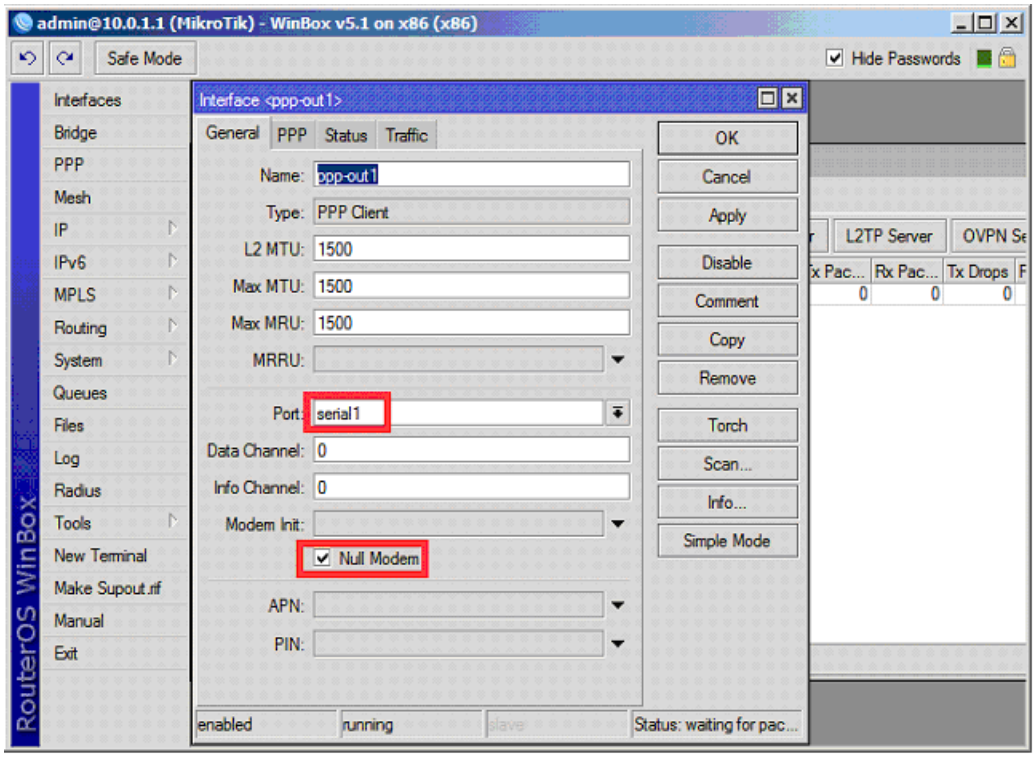



**Рис.8. PPP-клиент**

В winbox R1 добавим PPP-клиент (PPP->Interfaces + PPP-client ->advanced mode) связав его с новым последовательным портом `serial в режиме ноль-модема`. См.

Рис.8. В закладке PPP добавим PPP-пользователя q с паролем q, очистив флаги соединения по запросу, default route и DNS. См. Рис. 9.

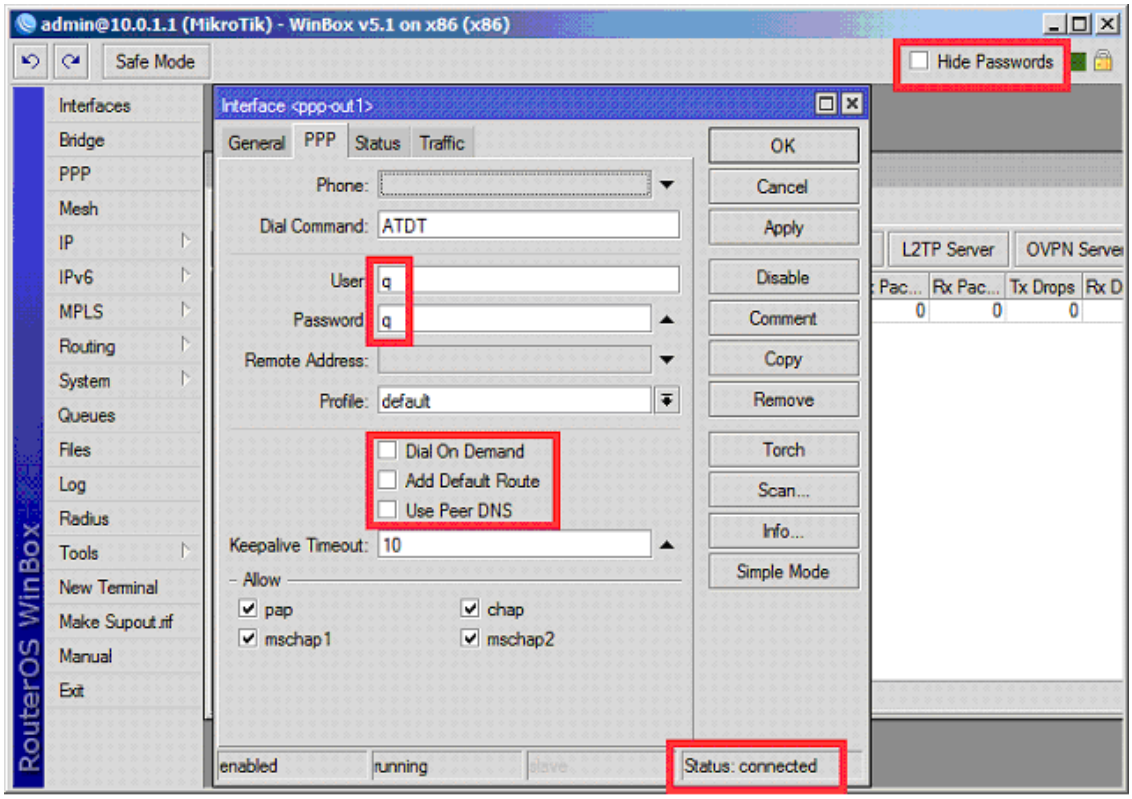



**Рис. 9. PPP-клиент**

У клиента и сервера видим статус connected (соединён). Соединён, но без IP-протокола. Об этом позже.

## Настройка PPTP

Для реализации PPTP, SSTP, L2TP и OPENVPN канала между маршрутизаторами R0 и R1 в топологии, изображённой на рис.5 нужно иметь IP-канал. Сделаем его м помощью модели Интернета. Для этого, как и ранее пропишем маршруты в сторону тап-инерффейса Ubuntu

[admin@R0] > **ip route add dst-address=10.0.0.0/16 gateway=10.0.0.2**

[admin@R1] > **ip route add dst-address=10.0.0.0/16 gateway=10.0.1.2**

Теперь маршрутизаторы R0 и R1 видят друг друга по IP. См. **Рис. 10.**

Итак между маршрутизаторами R0 и R1 установлена связь по IP. Осуществим через этот канал соединение маршрутизаторов по протоколу PPTP. Пусть R0 будет PPTP-сервером, а R1 PPTP-клиентом.

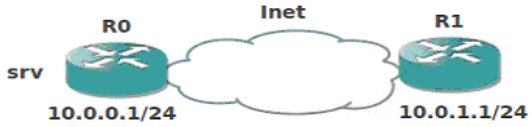



**Рис. 10.**

В winbox R0 добавим PPTP-сервер (PPP->**кнопка** PPTP-server). См. Рис. 11.

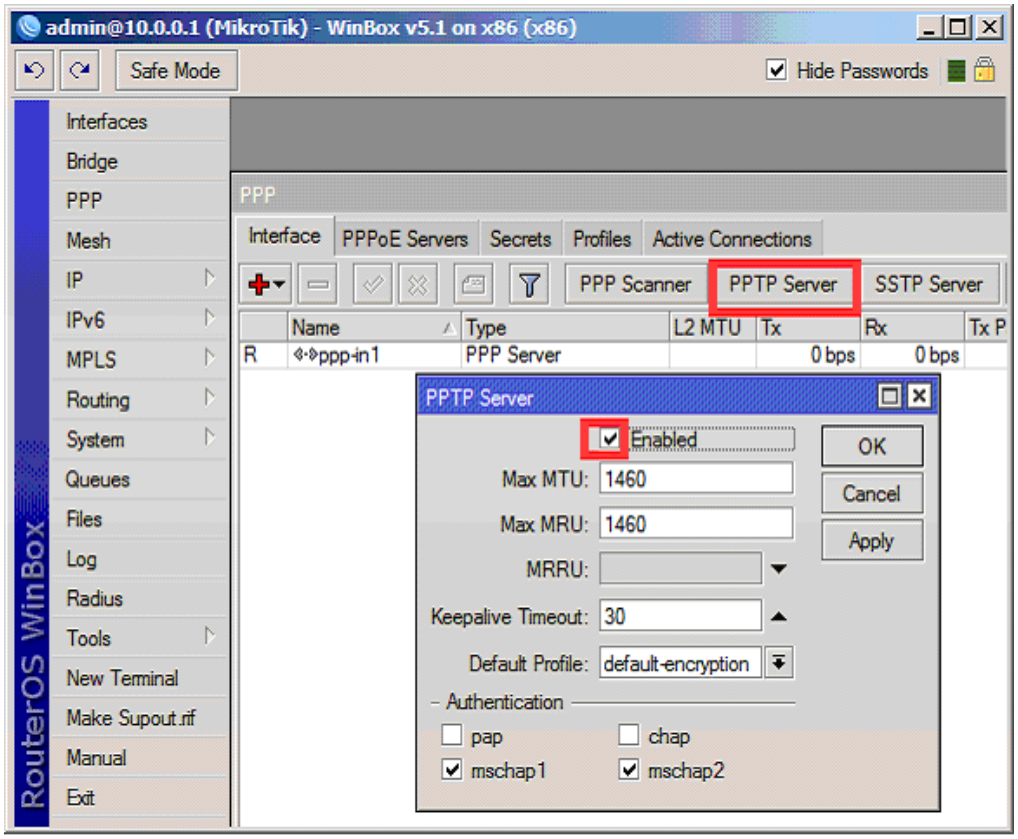



**Рис. 11. PPTP-сервер**

В winbox R1 добавим PPTP-клиент (PPP->Interfaces + PPTP-client ->Dial Out), указав адрес PPTP-сервера 10.0.0.1 и добавив созданного ранее PPP-пользователя q с паролем q. См. Рис. 12. У клиента и сервера видим статус connected (соединён). Соединен, но без IP-протокола. Об этом позже.

## Настройка L2TP

Осуществим соединение маршрутизаторов по протоколу L2TP. Пусть R0 будет L2TP-сервером, а R1 L2TP-клиентом. В winbox R0 добавим L2TP-сервер (PPP->**кнопка** L2TP-server). См. Рис. 13. В winbox R1 добавим L2TP-клиент (PPP->Interfaces + L2TP-client ->Dial Out), указав адрес L2TP-сервера 10.0.0.1 и добавив созданного ранее PPP-пользователя q с паролем q. См. Рис. 14.

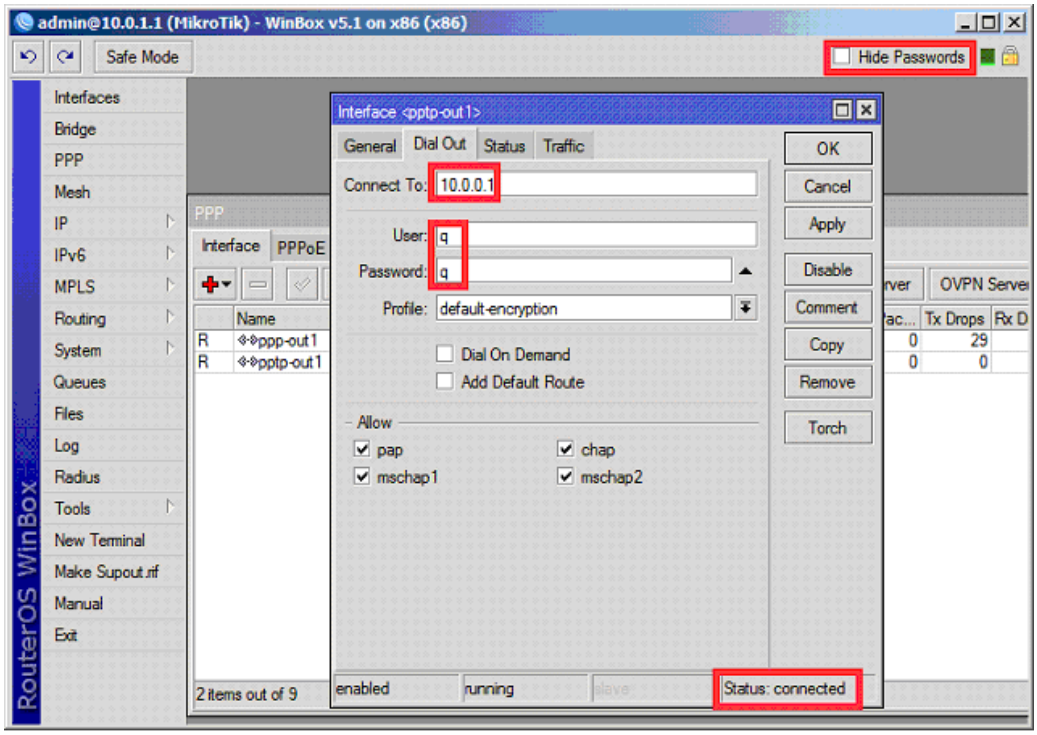



**Рис. 12. PPTP-клиент**

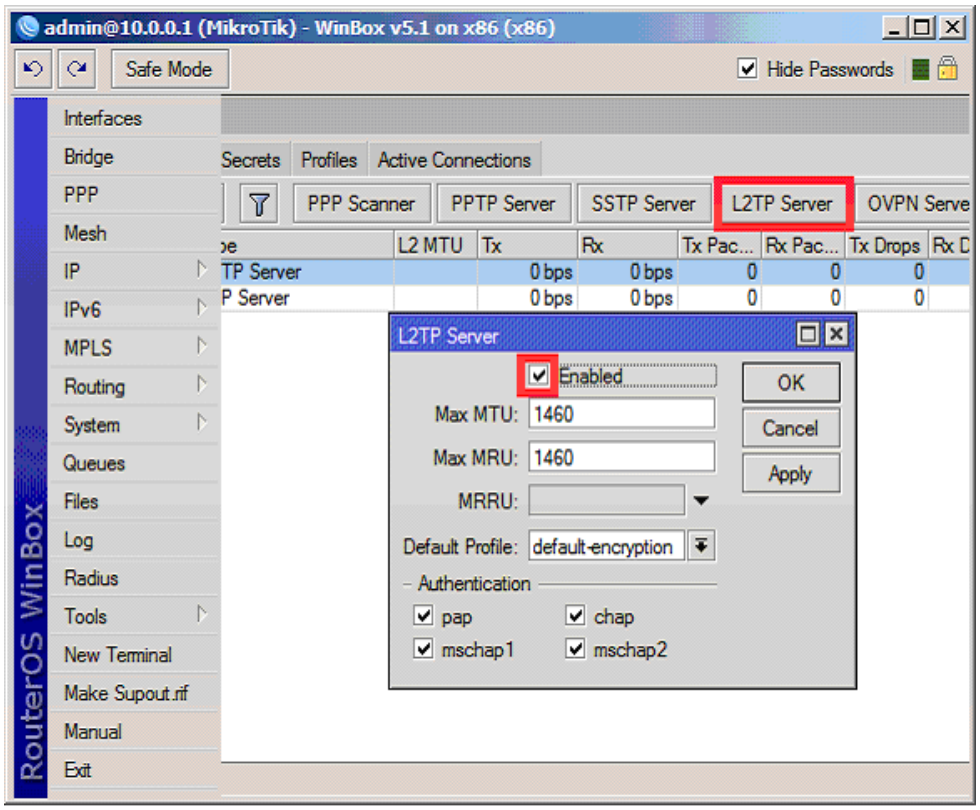



**Рис. 13. L2TP -сервер**

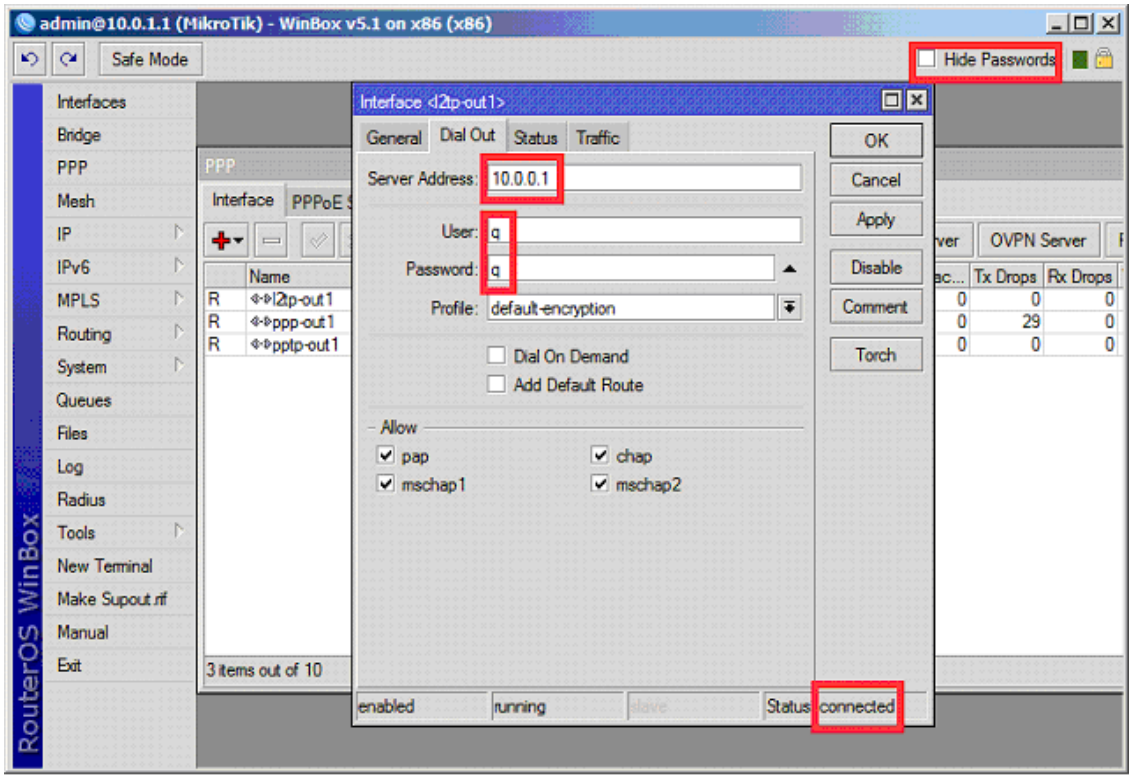



**Рис. 14. L2TP –клиент**

У клиента и сервера видим статус connected (соединён). Соединен, но без IP-протокола. Об этом позже. Процесс соединения полностью аналогичен PPTP.

## RSA-сертификаты

Механизм сертификатов основан на технологии несимметричного шифрования, осуществляемой парой ключей – открытым и закрытым. Если сообщение зашифровано открытым ключом, то может быть расшифровано только закрытым ключом и если сообщение зашифровано закрытым ключом, то может быть расшифровано только открытым ключом. Обмениваются только открытыми ключами.

Можно предложить такой механизм аутентификации (проверки подлинности), основанный на несимметричном шифрования

Б шлёт А запрос на аутентификацию

А в ответ шлёт Б случайную последовательность.

Б шифрует её закрытым ключом и отправляет зашифрованное А.

А расшифровывает принятое и сравнивает с отосланным. Если совпало, то Б это Б

А аутентифицировал Б и сообщает ему об этом

Раздачей сертификатов управляет удостоверяющий центр. Сертификат содержит «паспортные» данные получателя сертификата, открытый ключ и цифровую подпись удостоверяющего центра (зашифрованные открытым ключом удостоверяющего центра паспортные данные и открытый ключ из этого сертификата).

Обмен сертификатами равносилен обмену открытыми ключами. Два корреспондента получив сертификаты, могут ими обменятся и затем проверить их на подлинность, если у них есть сертификат удостоверяющего центра. Для этого они должны зашифровать данные

этого сертификата открытым ключом удостоверяющего центра и сравнивают с цифровой подписью проверяемого сертификата.

Сертификаты можно использовать для аутентификации (проверки подлинности). Уже сама проверка подписи присланного сертификата может являться аутентификацией отправителя.

В домашних условиях вы должны попрактиковаться с сертификатами, используя openssl, встроенный в openvpn. В начале отредактируйте файл openssl.conf. Вы ещё не раз вернётесь к этому файлу.

Работать с openssl надо только через cmd. Способов работы с сертификатами с помощью openssl много. Рассмотрим один из них.

Вам надо стать удостоверяющим центром. Для этого создаём закрытый ключ удостоверяющего центра по алгоритму DES длиной 1024 бит

**openssl genrsa -des3 -out ca.key 1024**

```
Generating RSA private key, 1024 bit long modulus
.............++++++
.........................++++++
e is 65537 (0x10001)
Enter pass phrase for ca.key:
Verifying - Enter pass phrase for ca.key:
```

Далее на основании этого закрытого ключа создаём самоподписанний корневой сертификат CA (Certificate Authority) ca.crt

**openssl req -new -x509 -days 3650 -key ca.key -out ca.crt**

```
Country Name (2 letter code) [UA]:
State or Province Name (full name) [DP]:
Locality Name (eg, city) [DNSK]:
Organization Name (eg, company) [DNU]:
Organizational Unit Name (eg, section) [FFECS]:
Common Name (eg, your name or your server's hostname) [CA]:
Name [name_default]:
Email Address [mail@ca.com]:
```

Для получения сертификата в удостоверяющем центре надо создать закрытый ключ, создать на основе этого ключа запрос на сертификат и отослать запрос в удостоверяющий центр. Удостоверяющий центр на основании запроса создаёт и подписывает сертификат и отправляет его запросившему. Для простоты будем получать сертификаты у самого себя.

Создаём закрытый ключ **qqq.key** по алгоритму DES длиной 1024 бит

**openssl genrsa -des3 -out qqq.key 1024**

Создаём запрос на сертификат у самого себя, то есть у созданного своего удостоверяющего центра

**openssl req -new -key qqq.key -out qqq.csr**

```
Country Name (2 letter code) [UA]:
State or Province Name (full name) [DP]:
Locality Name (eg, city) [DNSK]:
Organization Name (eg, company) [DNU]:
```

Organizational Unit Name (eg, section) [FFECS]:
Common Name (eg, your name or your server's hostname) [CA]:CNqqq
Name [name_default]:Nameqqq
Email Address [mail@ca.com]:qqq@mail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

Отправлять запрос **qqq.csr** самому себе естественно не надо. Просто создаём и подписываем сертификат **qqq.crt**

**openssl x509 -req -days 3650 -in qqq.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out qqq.crt**

```
Signature ok
subject=/C=UA/ST=DP/L=DNSK/O=DNU/OU=FFECS/CN=CNqqq/name=Nameqq
q/emailAddress=qqq@mail.com
Getting CA Private Key
Enter pass phrase for ca.key:
```

Теперь мы понимаем, что такое сертификаты. Вернёмся в Ubuntu.

## Настройка SSTP

Осуществим соединение маршрутизаторов по протоколу SSTP. Пусть R0 будет SSTP-сервером, а R1 SSTTP-клиентом. SSTP соединение требует RSA-сертификатов.

Перепишем папку /usr/share/doc/openvpn/examples/easy-rsa/2.0 в свою папку. Рапакуем в ней 2 архива. Определим в файле Makefile DESTDIR=/home/ВашаПапка и PREFIX=RSA. Инсталируйте: **make install**. Войдём из консоли в появившуюся папку RSA. Выполните
**. vars**

через пробел.
**./clean-all**

Исправьте под свои нужды файл vars и выполните команды для установки переменных окружения и очистки базы сертификатов
**source ./vars**
**./clean-all**

Создайте один раз корневой сертификат CA (Certificate Authority), необходимый для подписи сертификатов клиента и сервера.
**./pkitool --initca**

Создайте сертификат сервера
**./pkitool --server server**

Создайте сертификат клиента
**./pkitool client**

Здесь server и client – любые имена.

Сертификаты появятся в папке keys. Перейдите в неё. Посмотрите файл базы сертификатов index.txt.

Всегда перед началом выполнения команд в этой папке установите переменные окружения командой **. vars**

или **source ./vars**.

Перепишем сертификаты и ключ из Ubuntu в SSTP-сервер маршрутизатора Mikrotik .

**ftp 10.0.0.1**
**name: admin**
**password;**
**bin**
**put ca.crt**
**put server.crt**
**put server.key**
**quit**

Аналогично перепишем сертификаты и ключ в SSTP- клиент.

В winbox R0 в позиции Files проверим наличие переданных файлов ca.crt server.crt server.key. Выберем System -> Certificates нажимая 3 раза кнопку Import последовательно импортируем файлы ca.crt server.crt server.key. После импорта server.key соответствующая строка получит метку KR. Дважды щёлкнув на этой строке, изменим имя сертификата на srv. См. Рис. 15.

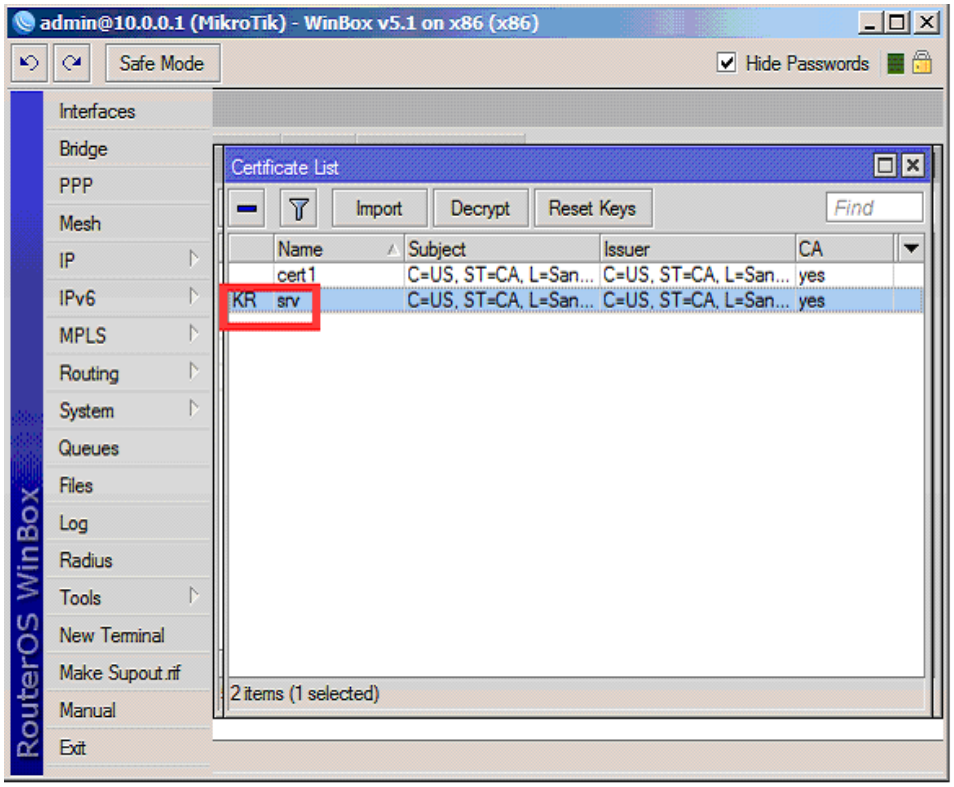



**Рис. 15. Импорт сертификатов**

В winbox R1 в позиции Files проверим наличие переданных файлов ca.crt client.crt client.key. Выберем System -> Certificates, нажимая 3 раза кнопку Import последовательно импортируем файлы ca.crt client.crt client.key. После импорта client.key соответствующая строка получит метку KR. Дважды щёлкнув на этой строке, изменим имя сертификата на cli

В winbox R0 добавим SSTP-сервер (PPP->**кнопка** SSTP-server), указав в нём имя серверного сертификата srv с проверкой сертификата клиента. См. Рис. 16.

В winbox R1 добавим SSTP-клиент (PPP->Interfaces + L2TP-client ->Dial Out), указав адрес SSTP-сервера 10.0.0.1, добавив созданного ранее PPP-пользователя q с паролем q и указав имя клиентского сертификата cli с проверкой сертификата сервера. См. Рис. 17.

У клиента и сервера видим статус connected (соединён). Соединён,но без IP-протокола. Об этом позже.

## Настройка OPENVPN

Осуществим соединение маршрутизаторов по протоколу OPENVPN. Пусть R0 будет OPENVPN -сервером, а R1 OPENVPN -клиентом. OPENVPN соединение требует RSA-сертификатов. Воспользуемся сертификатами srv и cli, импортированными при создании SSTP соединения.

В winbox R0 добавим OPENVPN-сервер (PPP->**кнопка** OVPN-server), указав в нём имя серверного сертификата srv с проверкой сертификата клиента и режим Ethernet, который понадобится для организации моста. См. Рис. 18.

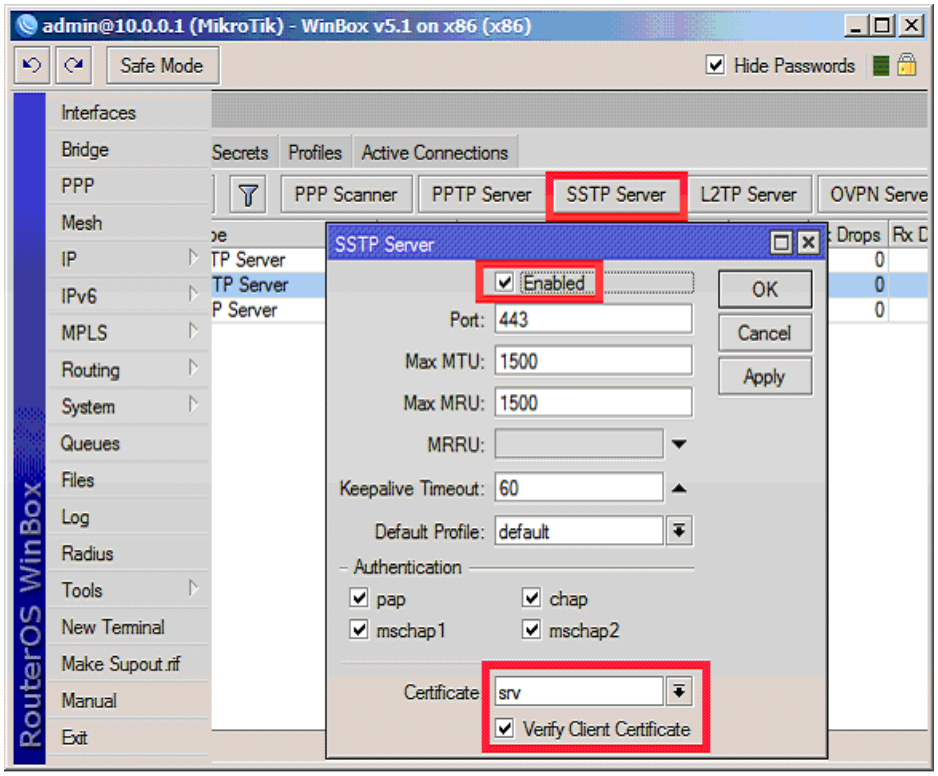



**Рис. 16. SSTP-сервер**

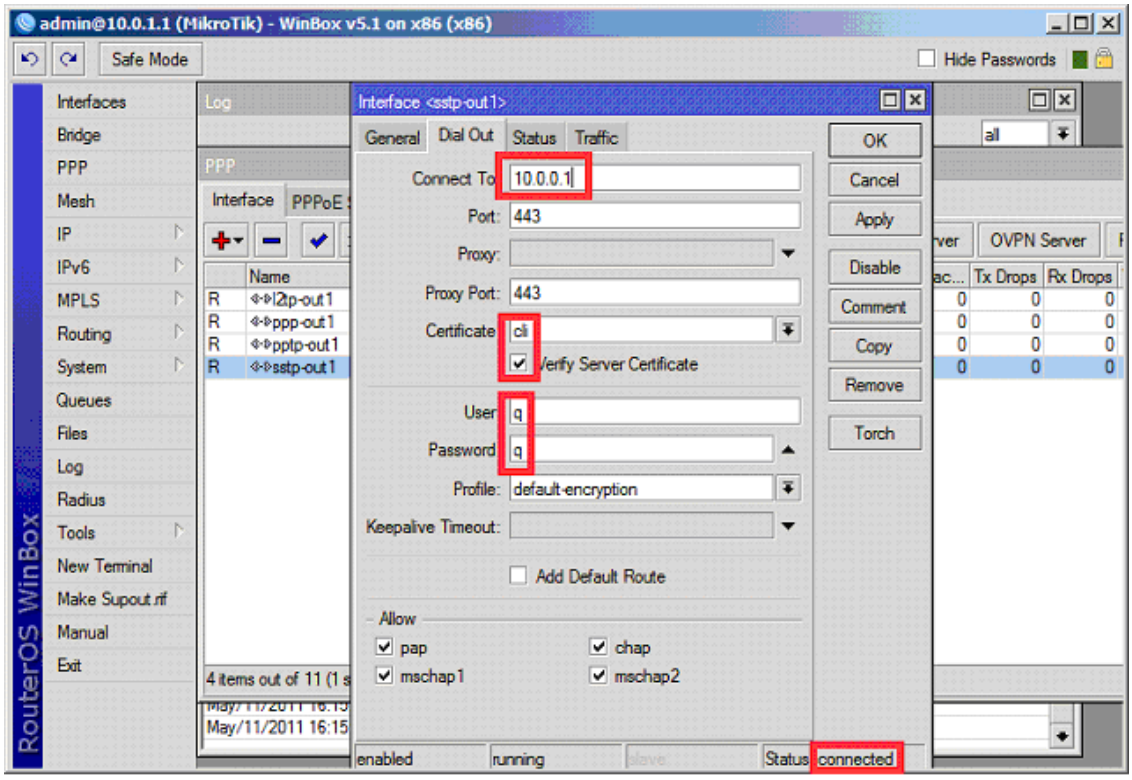



**Рис. 17. SSTP-клиент**

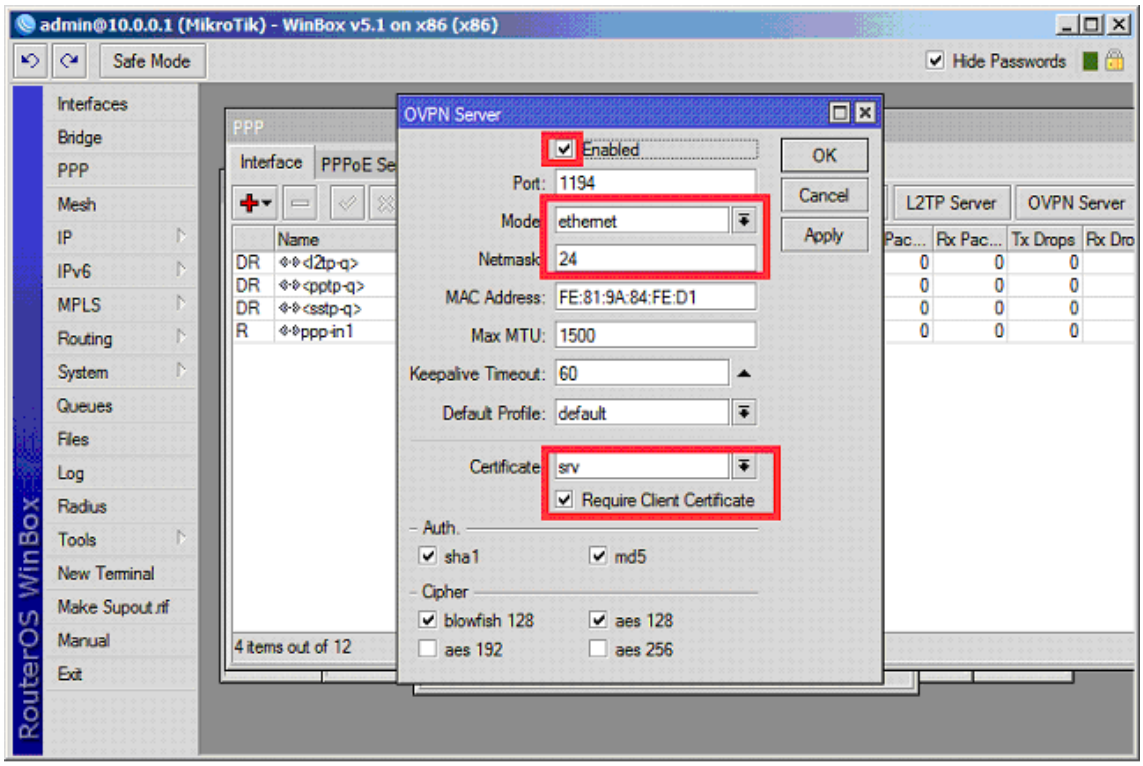



**Рис. 18. OPENVPN-сервер**

Добавим в winbox R0 мост (Bridge+).

В протоколе OPENVPN у сервера надо задать локальный и удалённый адрес, даже если мы его не будем использовать. Заметим, что OPENVPN–сервер использует профиль default. Откроем его (PPP->profiles->default) и укажем в нём мост и произвольные (пока) локальный и удалённый адрес 192.168.100.1 и 192.168.200.1. См. Рис. 19.

В winbox R1 добавим мост (Bridge+). Заметим, что OPENVPN –клиент использует профиль default. Откроем его (PPP->profiles->default) и укажем в нём мост для единообразия настройки. См. Рис. 20.

В winbox R1 добавим OPENVPN -клиент (PPP->Interfaces + OPENVPN -client ->Dial Out) режиме Ethernet (который понадобится для организации моста), указав адрес OPENVPN -сервера 10.0.0.1, добавив созданного ранее PPP-пользователя q с паролем q и указав имя клиентского сертификата cli с проверкой сертификата сервера. См. Рис. 20.

У клиента и сервера видим статус connected (соединён). В winbox видим (IP->addresses) что сервер и клиент получили назначенные адреса 192.168.100.1/32 и 192.168.200.1/32.

Итак маршрутизаторы R0 и R1 соединены по 5 каналам. См. Рис. 21 и 22.

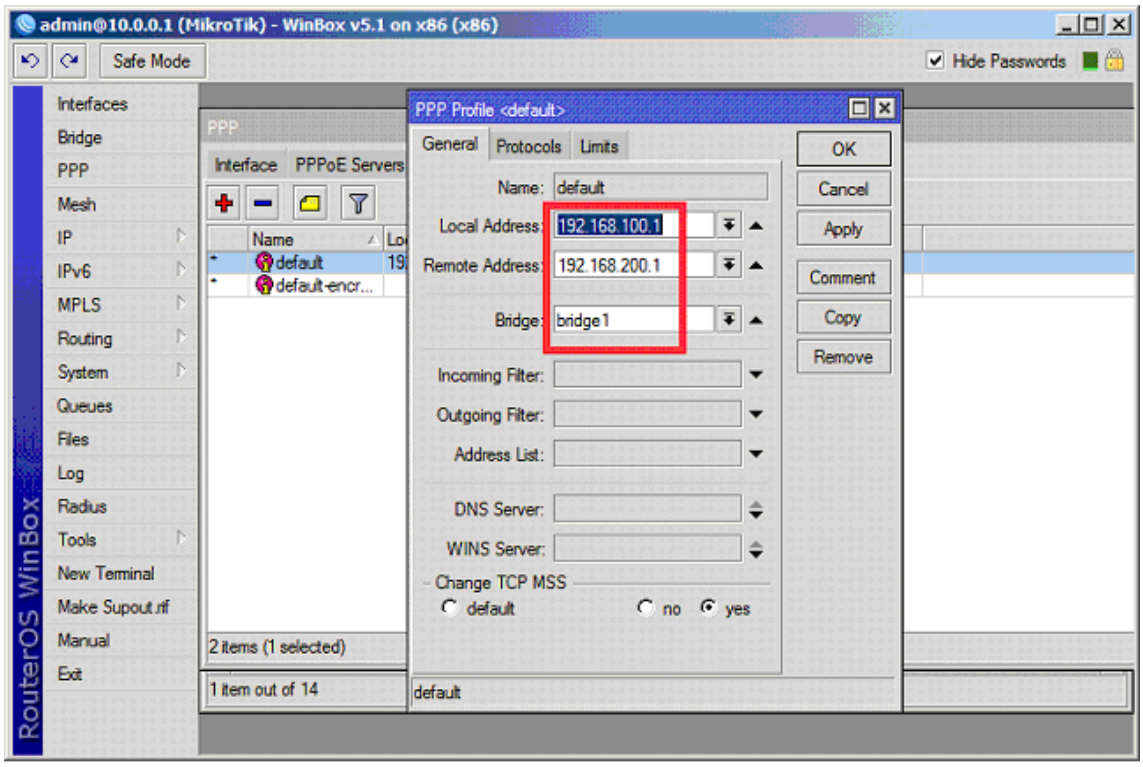



**Рис. 19. Профиль для OPENVPN-сервера**.

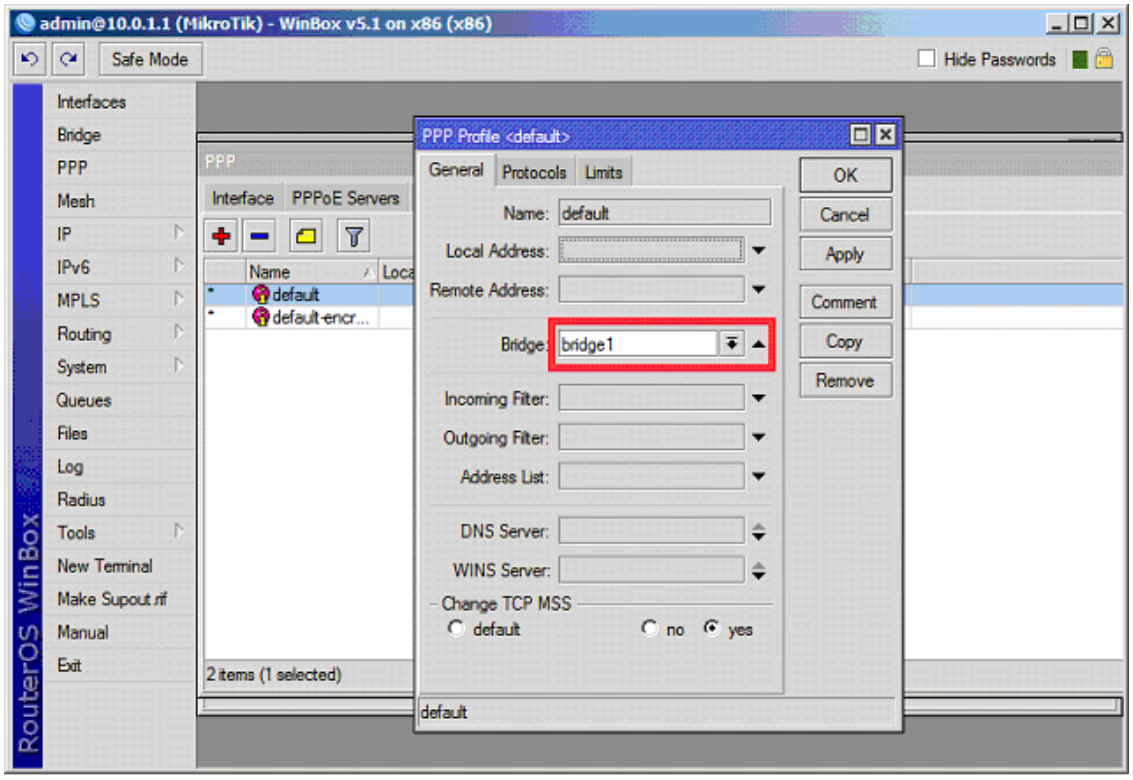



**Рис. 19. Профиль для OPENVPN-клиента**

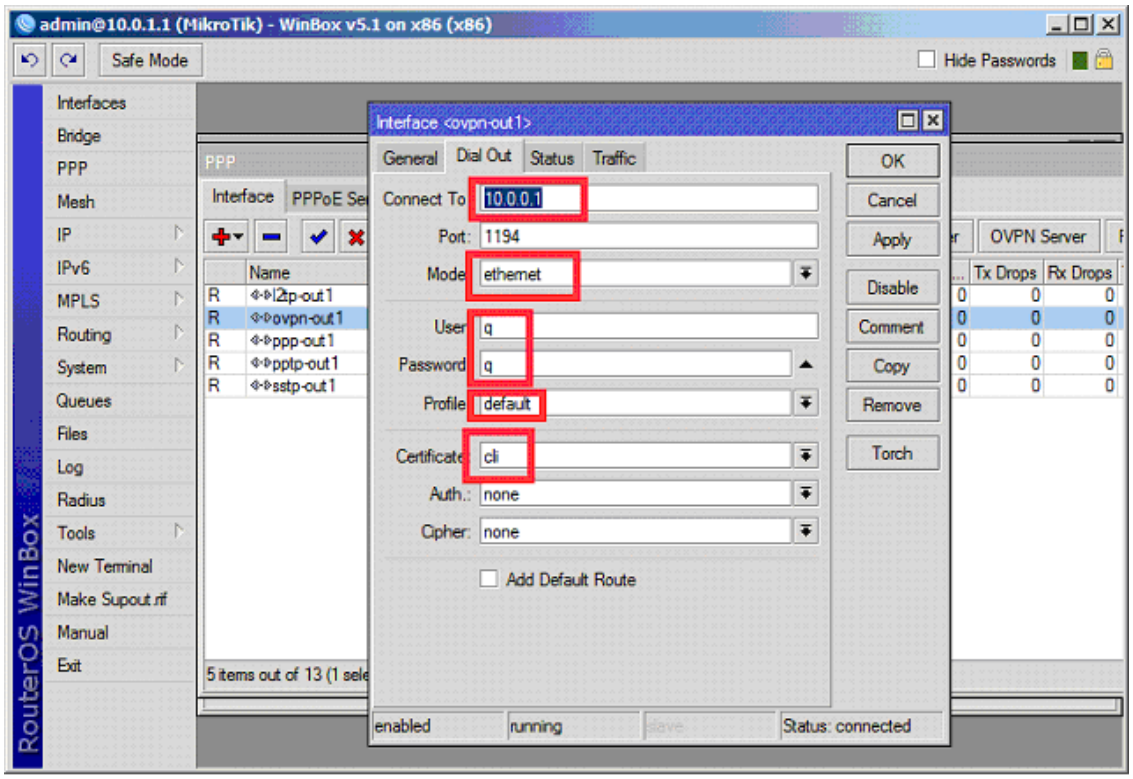



**Рис. 20. OPENVPN -клиент**

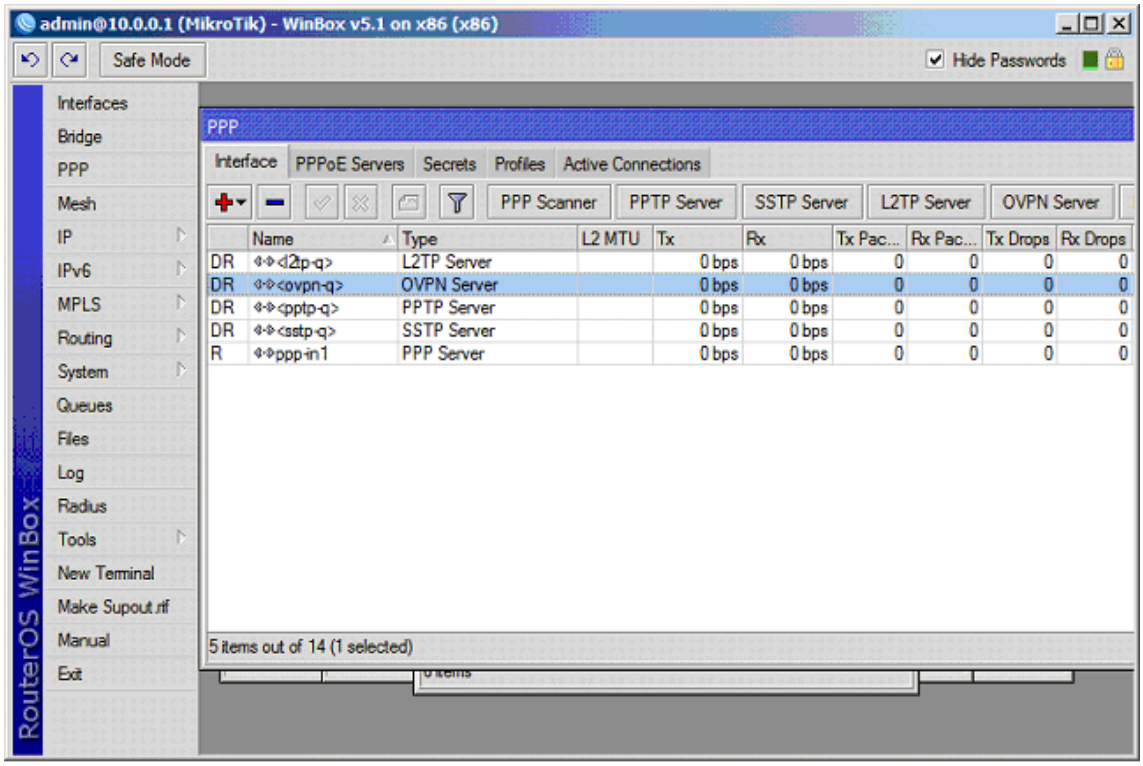



**Рис. 21. 5 соединенных серверов**

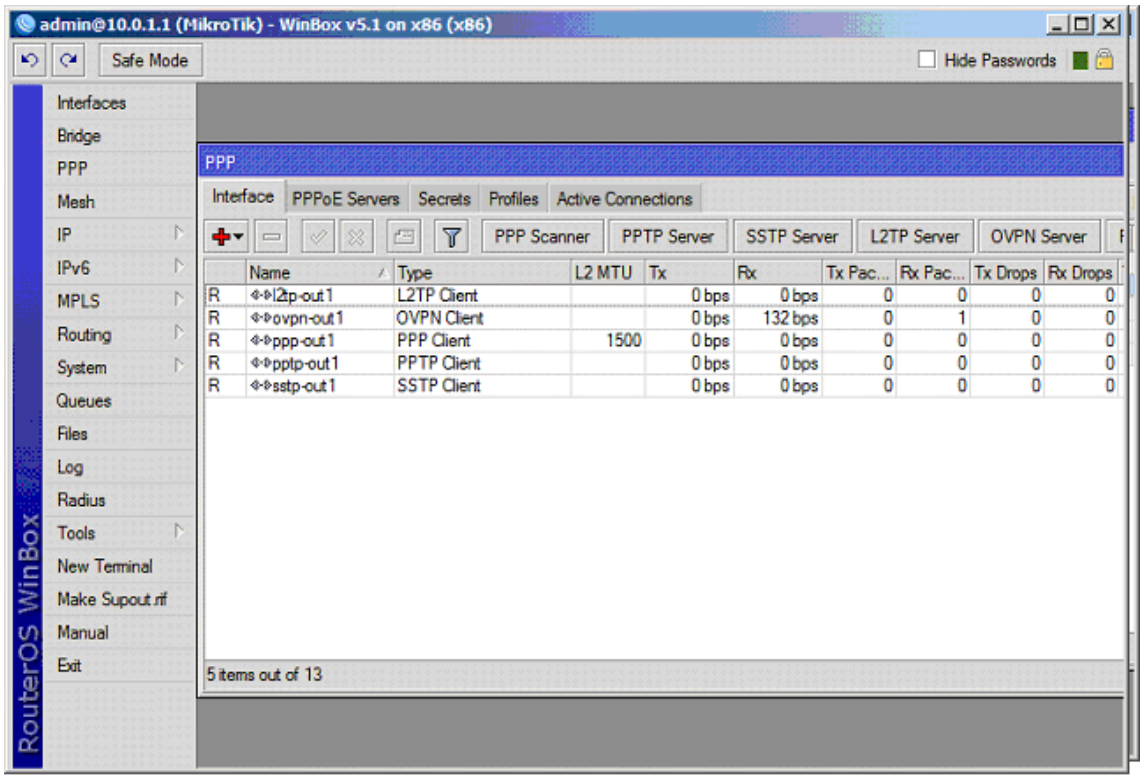



**Рис. 22. 5 соединенных клиентов**

# VPN уровня 2

Изучим организацию VPN уровня 2 с помощью мостов.

Так как у нас используется один пользователь q и два пофиля default и default encryption для всех соединений, то во избежании взаимного влияния будем строить мосты по очереди для одного типа PPP-соединения, оставляя другие неактивными.

Деактивируем всех клиентов, нажимая в winbox R1->PPP значок X на каждом клиенте. Деактивируем сервера (winbox R0->PPP), снимая галочку enable в их настройках. Сервер PPP R0 (winbox R0->PPP) деактивируется нажатием кнопки disable в его настройках.

Остановим топологию, добавим два элемента qemu host, соединив их, как указано на рисунке Рис. 23.

Последовательно используем все 5 видов PPP-соединений для организации моста между R2 и R3. При успешном создании моста R2 и R3 будут в одном Ethernet-сегменте, образуя VPN уровня 2. Адреса для R2 и R3 можно назначать из одной IP-подсети, например 172.16.1.0/24

Запустим топологию. Назначим адаптерам адреса согласно Рис. 23.

И на клиенте R1 и на сервере R0 добавим в ранее созданный мост bridge1 интерфейс ether1. (В winbox это Bridge->ports +). См. рис. 24. Проверим, чтобы этот мост фигурировал в настройках профиля default и на клиенте R1 и на сервере R0 (PPP->profiles->default). В настройках по умолчанию используется также профиль default encryption. Сделаем так, чтобы мост bridge1 фигурировал в настройках этого профиля и на клиенте R1 и на сервере R0 (PPP->profiles->default encryption). См. рис. 25.

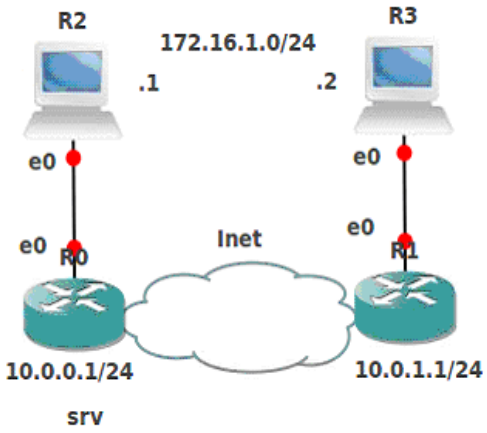



**Рис. 23. Топология для VPN уровня 2**

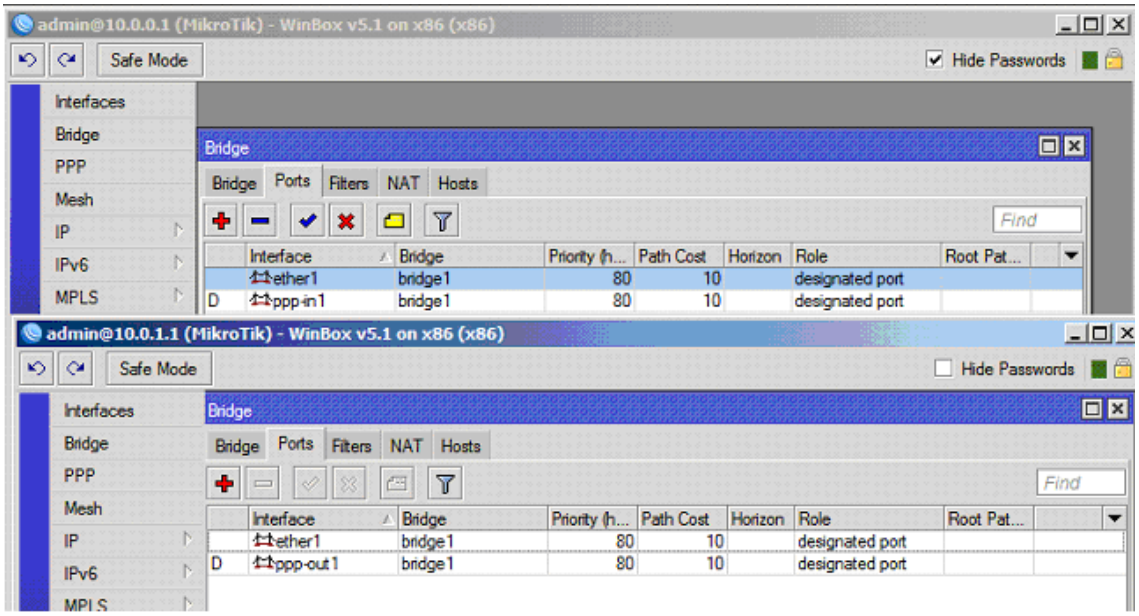



**Рис. 24. Мост в PPP**

## PPP

Активируем сервер PPP R0 (winbox R0->PPP), нажав кнопку enable в его настройках. Активируем клиент PPT R1 (winbox R0->PPP), нажав на строке ppp правой кнопкой и выбрав enable. И на клиенте R1 и на сервере R0 у PPTP -интерфейсов должна появиться метка R, а эти интерфейсы автоматически добавятся в мост bridge1 (см. рис.24). R3 видит R2 по IP и наоборот. Проверьте утилитой пинг.

Деактивируем PPP -клиента, нажимая в winbox R1->PPP значок X на соответствующей строке. Деактивируем PPPсервер (winbox R0->PPP), снимая галочку enable в его настройках.

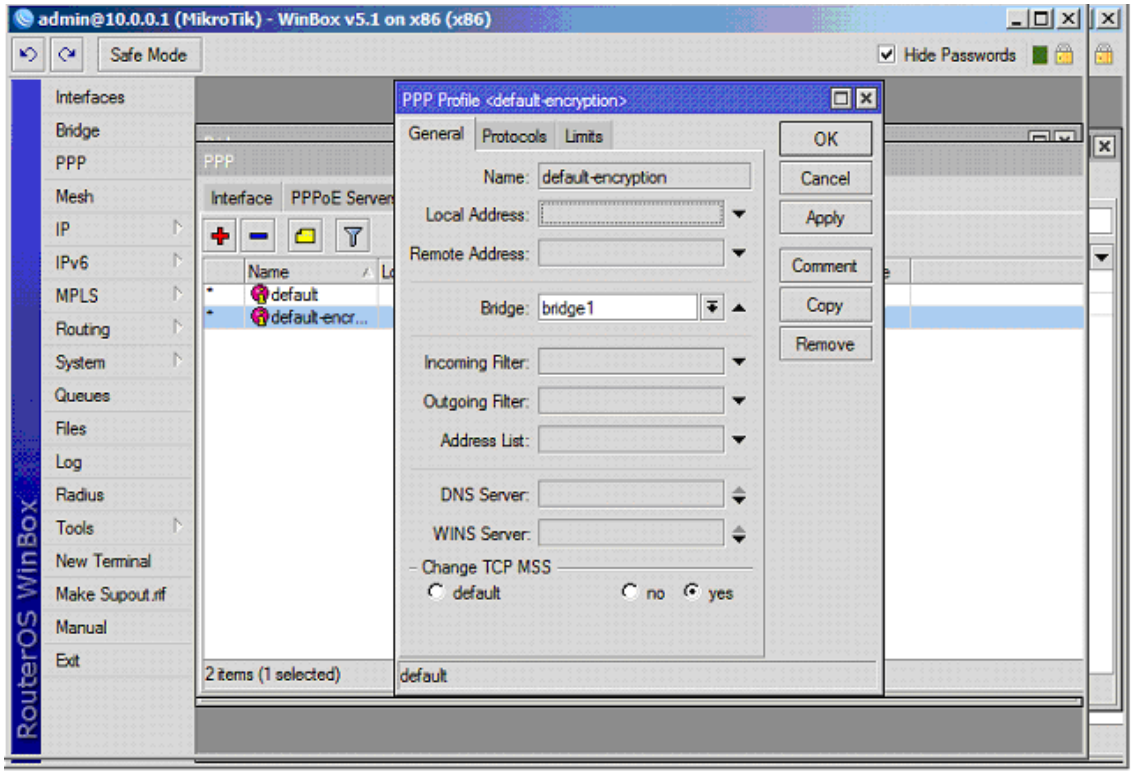



**Рис. 25. Добавление моста в профиль**

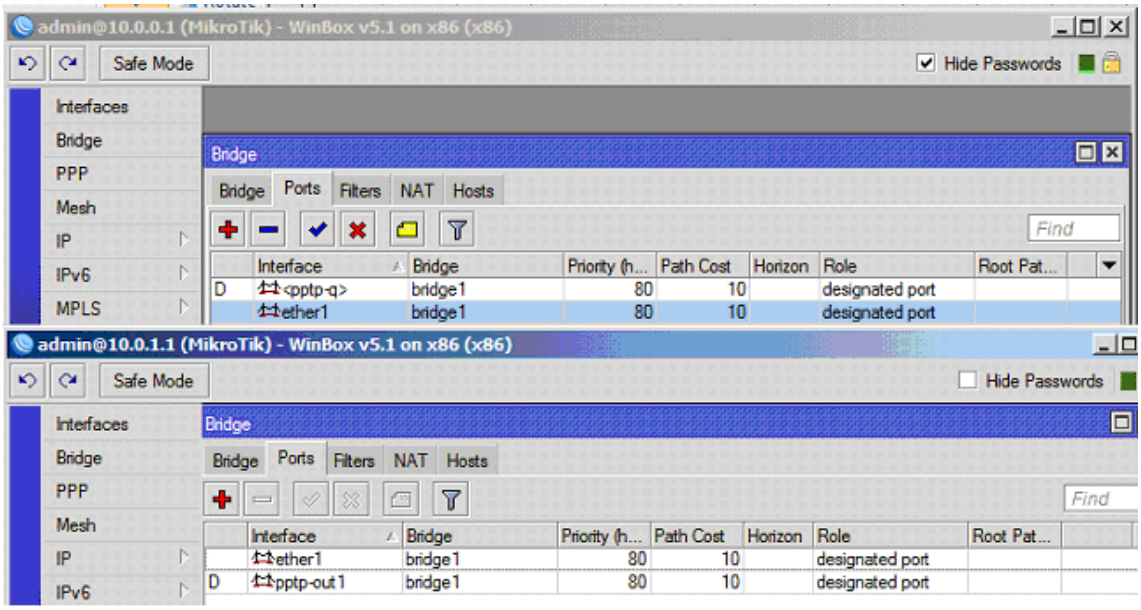



**Рис. 26. Мост в PPTP**

**PPTP**

Активируем сервер PPTP R0 (winbox R0->PPP), установив галочку enable в его настройках. По умолчанию PPTP -клиент и сервер использует профиль default encryption. Активируем клиент PPTP R1 (winbox R0->PPP), нажав на строке pptp правой кнопкой и выбрав enable. И на клиенте R1 и на сервере R0 у PPTP -интерфейсов должна появиться метка R, а эти интерфейсы автоматически добавятся в мост bridge1. См. рис. 26. R3 видит R2 по IP и наоборот. Проверьте.

В домашних условиях и при наличии прав вы можете воспользоваться анализатором пакетов Wireshark. При установлении PPTP-соединения вы увидите картину, изображённую на рис. 27.

| | |
|---|---|
| TCP | 34077 > pptp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSV=236554 TSER=0 WS=4 |
| TCP | pptp > 34077 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSV=236634 TSER |
| TCP | 34077 > pptp [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=236555 TSER=236634 |
| PPTP | Start-Control-Connection-Request |
| TCP | pptp > 34077 [ACK] Seq=1 Ack=157 Win=6864 Len=0 TSV=236635 TSER=236555 |
| PPTP | Start-Control-Connection-Reply |
| TCP | 34077 > pptp [ACK] Seq=157 Ack=157 Win=6912 Len=0 TSV=236555 TSER=236635 |
| PPTP | Outgoing-Call-Request |
| PPTP | Outgoing-Call-Reply |
| PPP LCP | Configuration Request |
| PPP LCP | Configuration Request |
| PPP LCP | Configuration Ack |
| PPP LCP | Configuration Ack |
| PPP CHAP | Challenge (NAME='MikroTik', VALUE=0x08094c6135b6e26000636e14080ee390) |
| PPP CHAP | Response (NAME='q', VALUE=0x91f135c4d0ec3c9bc1476fdcde162fba0000000000000000...) |
| PPP CHAP | Success (MESSAGE='S=473DFBE2FBC4259D50C23F8A6F1B461E00D81EA6') |
| PPP IPCP | Configuration Request |
| PPP IPCP | Configuration Request |
| PPP MPLS | Configuration Request |
| 0x8031 | PPP Bridging NCP (0x8031) |
| 0x8031 | PPP Bridging NCP (0x8031) |
| PPP IPCP | Configuration Reject |
| PPP IPCP | Configuration Reject |
| PPP LCP | Protocol Reject |
| 0x8031 | PPP Bridging NCP (0x8031) |
| PPP IPCP | Configuration Request |
| 0x8031 | PPP Bridging NCP (0x8031) |
| PPP IPCP | Configuration Request |
| PPP IPCP | Configuration Ack |
| PPP IPCP | Configuration Ack |
| TCP | 34077 > pptp [ACK] Seq=325 Ack=189 Win=6912 Len=0 TSV=236559 TSER=236635 |
| PPP IPCP | Termination Request |
| PPP IPCP | Termination Request |
| PPP IPCP | Termination Ack |
| PPP IPCP | Termination Ack |

**Рис. 27 Установление PPTP-соединения в режиме моста.**

После организации связи и установки TCP- соединения видим такой стек протоколов: IP над GRE над PPP над BCP над Ethernet над IP над TCP. См рис. 28.

```
▸ Frame 386: 122 bytes on wire (976 bits), 122 bytes captured (976 bits)
▸ Ethernet II, Src: RealtekU_12:34:5c (52:54:00:12:34:5c), Dst: 7a:7c:c1:9f:8e:75 (7a:7c:c1:9f:8e:75)
▸ Internet Protocol, Src: 10.0.1.1 (10.0.1.1), Dst: 10.0.0.1 (10.0.0.1)
▸ Generic Routing Encapsulation (PPP)
▸ Point-to-Point Protocol
▸ PPP Bridging Control Protocol
▸ Ethernet II, Src: Intel_3f:6c:00 (00:aa:00:3f:6c:00), Dst: Intel_f6:4a:00 (00:aa:00:f6:4a:00)
▸ Internet Protocol, Src: 172.16.1.2 (172.16.1.2), Dst: 172.16.1.1 (172.16.1.1)
▸ Transmission Control Protocol, Src Port: 39105 (39105), Dst Port: telnet (23), Seq: 63, Ack: 549, Len: 0
```

**Рис. 28. Стек протоколов для PPTP-соединения в режиме моста при отключенном шифровании**

Приведенные скриншоты получены при отключенном шифровании. Как правило, в PPTP вложенные в PPP пакеты шифруются. В этом случае видимые на рис. 27 адреса (172.16.1.2 172.16.1.1) будут зашифрованы (равно как и данные) и будут недоступны для наблюдения. См рис. 29.

```
▸ Frame 61: 107 bytes on wire (856 bits), 107 bytes captured (856 bits)
▸ Ethernet II, Src: 4e:d5:cc:1b:a8:75 (4e:d5:cc:1b:a8:75), Dst: RealtekU_12:34:5c (52:54:00:12:34:5c)
▸ Internet Protocol, Src: 10.0.1.1 (10.0.1.1), Dst: 10.0.0.1 (10.0.0.1)
▸ Generic Routing Encapsulation (PPP)
▸ Point-to-Point Protocol
  PPP Compressed Datagram
```

**Рис. 29. Стек протоколов для PPTP-соединения в режиме моста при включенном шифровании**

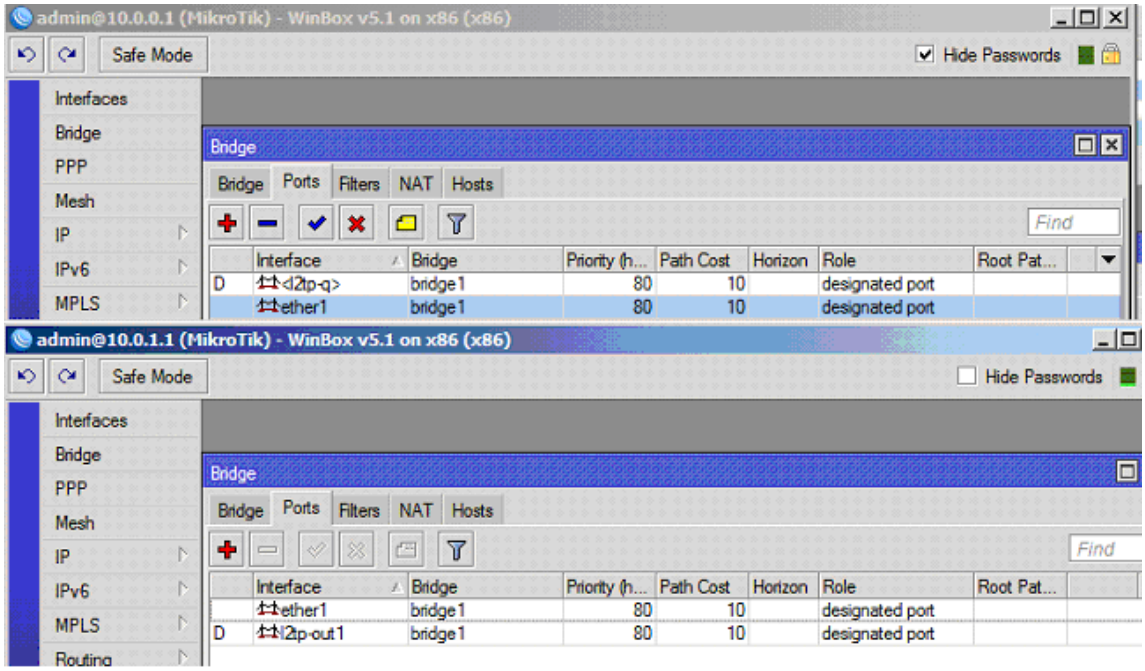



**Рис. 30. Мост в L2TP**

Деактивируем PPTP-клиента, нажимая в winbox R1->PPP значок X на соответствующей строке. Деактивируем PPTP сервер (winbox R0->PPP), снимая галочку enable в его настройках.

| | | |
|---|---|---|
| L2TP | Control Message - SCCRQ | (tunnel id=0, session id=0) |
| L2TP | Control Message - SCCRP | (tunnel id=1, session id=0) |
| L2TP | Control Message - SCCCN | (tunnel id=2, session id=0) |
| L2TP | Control Message - ZLB | (tunnel id=1, session id=0) |
| L2TP | Control Message - ICRQ | (tunnel id=2, session id=0) |
| L2TP | Control Message - ICRP | (tunnel id=1, session id=1) |
| L2TP | Control Message - ICCN | (tunnel id=2, session id=1) |
| L2TP | Control Message - ZLB | (tunnel id=1, session id=0) |
| PPP LCP | Configuration Request | |
| PPP LCP | Configuration Request | |
| PPP LCP | Configuration Ack | |
| PPP LCP | Configuration Ack | |
| PPP CHAP | Challenge (NAME='MikroTik', VALUE=0xa3fdf1eb4a9b0c889e3637f9e1e929e9) | |
| PPP CHAP | Response (NAME='q', VALUE=0x204a4a05465dcfe709e9cb52a05b9ef10000000000000000...) | |
| PPP CHAP | Success (MESSAGE='S=BF23BD328CF452588B7AF2267BC4B0B47993AF1F') | |
| PPP IPCP | Configuration Request | |
| 0x8031 | PPP Bridging NCP (0x8031) | |
| PPP IPCP | Configuration Request | |
| 0x8031 | PPP Bridging NCP (0x8031) | |
| PPP IPCP | Configuration Reject | |
| PPP IPCP | Configuration Reject | |
| 0x8031 | PPP Bridging NCP (0x8031) | |
| PPP IPCP | Configuration Request | |
| 0x8031 | PPP Bridging NCP (0x8031) | |
| PPP IPCP | Configuration Request | |
| PPP IPCP | Configuration Ack | |
| PPP IPCP | Configuration Ack | |
| PPP IPCP | Termination Request | |
| PPP IPCP | Termination Request | |
| PPP IPCP | Termination Ack | |
| PPP IPCP | Termination Ack | |

**Рис. 31. Установление L2TP-соединения в режиме моста.**

### L2TP

Активируем сервер L2TP R0 (winbox R0->PPP), установив галочку enable в его настройках. По умолчанию L2TP -клиент и сервер использует профиль default encryption. Активируем клиент L2TP R1 (winbox R0->PPP), нажав на строке l2tp правой кнопкой и выбрав enable. И на клиенте R1 и на сервере R0 у L2TP -интерфейсов должна появиться метка R, а эти интерфейсы автоматически добавятся в мост bridge1. См рис. 30. R3 видит R2 по IP и наоборот. Проверьте

```
▸ Frame 761: 121 bytes on wire (968 bits), 121 bytes captured (968 bits)
▸ Ethernet II, Src: 7a:7c:c1:9f:8e:75 (7a:7c:c1:9f:8e:75), Dst: RealtekU_12:34:5c (52:54:00:12:34:5c)
▸ Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.1.1 (10.0.1.1)
▸ User Datagram Protocol, Src Port: l2f (1701), Dst Port: l2f (1701)
▸ Layer 2 Tunneling Protocol
▸ Point-to-Point Protocol
▸ PPP Bridging Control Protocol
▸ Ethernet II, Src: Intel_f6:4a:00 (00:aa:00:f6:4a:00), Dst: Intel_3f:6c:00 (00:aa:00:3f:6c:00)
▸ Internet Protocol, Src: 172.16.1.1 (172.16.1.1), Dst: 172.16.1.2 (172.16.1.2)
▸ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 45485 (45485), Seq: 1664, Ack: 223, Len: 1
```

**Рис. 32. Стек протоколов для L2TP-соединения в режиме моста при отключенном шифровании**

В домашних условиях и при наличии прав вы можете воспользоваться анализатором пакетов Wireshark. При установлении L2TP-соединеия вы увидите картину, изображённую на рис. 31.

После организации связи и установки TCP- соединения видим такой стек протоколов: IP над UDP над L2TP над PPP над BCP над Ethernet над IP над TCP. См рис. 32.

Приведенные скриншоты получены при отключенном шифровании. Как правило в L2TP вложенные в PPP пакеты шифруются и реально вы ничего не увидите. См. рис. 33.

```
Frame 66: 112 bytes on wire (896 bits), 112 bytes captured (896 bits)
Ethernet II, Src: RealtekU_12:34:5c (52:54:00:12:34:5c), Dst: 4e:d5:cc:1b:a8:75 (4e:d5:cc:1b:a8:75)
Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.1.1 (10.0.1.1)
User Datagram Protocol, Src Port: l2f (1701), Dst Port: l2f (1701)
Layer 2 Tunneling Protocol
Point-to-Point Protocol
PPP Compressed Datagram
```

**Рис. 33. Стек протоколов для L2TP-соединения в режиме моста при включенном шифровании**

Деактивируем L2TP -клиента, нажимая в winbox R1->PPP значок X на соответствующей строке. Деактивируем L2TP сервер (winbox R0->PPP), снимая галочку enable в его настройках.

**SSTP**

Активируем сервер SSTP R0 (winbox R0->PPP), установив галочку enable в его настройках. По умолчанию SSTP-клиент использует профиль default encryption. Активируем клиент SSTP R1 (winbox R0->PPP), нажав на строке sstp правой кнопкой и выбрав enable. И на клиенте R1 и на сервере R0 у SSTP -интерфейсов должна появиться метка R, а эти интерфейсы автоматически добавятся в мост bridge1. См. рис. 34.

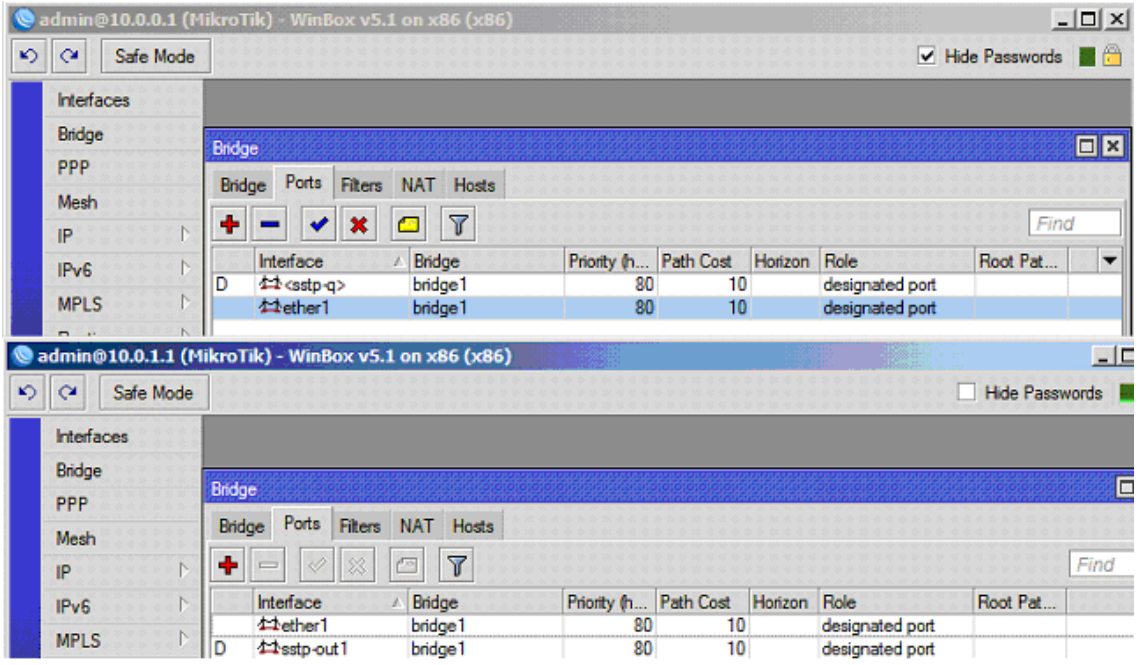
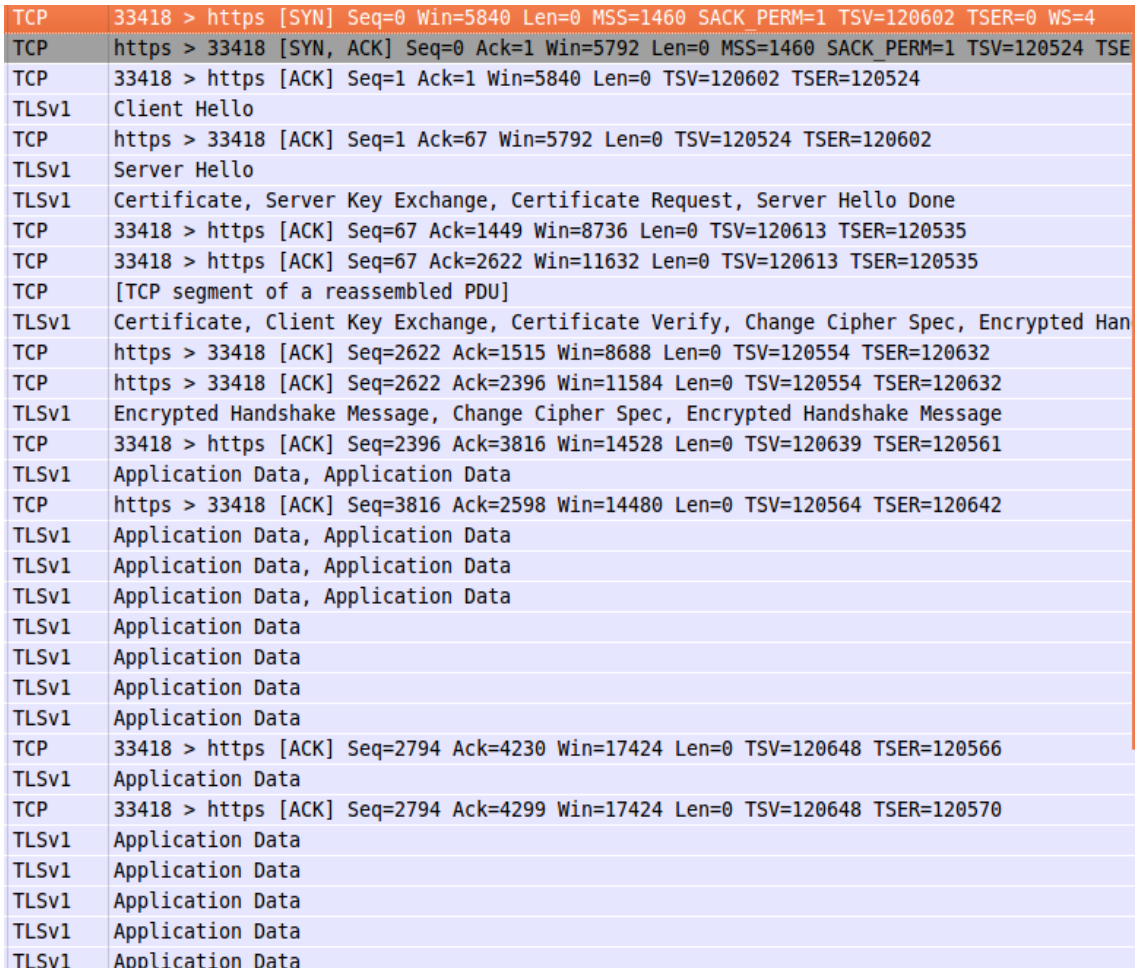



**Рис. 34. Мост в SSTP**

R3 видит R2 по IP и наоборот. Проверьте.

В протоколе SSTP для организации канала используется шифрование по протоколу tls v1 ssl. В sstp установка соединения и обмен данными происходит в зашифрованом виде, и в Wireshark мы ничего не увидим. См. рис. 35.



| | |
|---|---|
| TCP | 33418 > https [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSV=120602 TSER=0 WS=4 |
| TCP | https > 33418 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSV=120524 TSE |
| TCP | 33418 > https [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=120602 TSER=120524 |
| TLSv1 | Client Hello |
| TCP | https > 33418 [ACK] Seq=1 Ack=67 Win=5792 Len=0 TSV=120524 TSER=120602 |
| TLSv1 | Server Hello |
| TLSv1 | Certificate, Server Key Exchange, Certificate Request, Server Hello Done |
| TCP | 33418 > https [ACK] Seq=67 Ack=1449 Win=8736 Len=0 TSV=120613 TSER=120535 |
| TCP | 33418 > https [ACK] Seq=67 Ack=2622 Win=11632 Len=0 TSV=120613 TSER=120535 |
| TCP | [TCP segment of a reassembled PDU] |
| TLSv1 | Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Han |
| TCP | https > 33418 [ACK] Seq=2622 Ack=1515 Win=8688 Len=0 TSV=120554 TSER=120632 |
| TCP | https > 33418 [ACK] Seq=2622 Ack=2396 Win=11584 Len=0 TSV=120554 TSER=120632 |
| TLSv1 | Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message |
| TCP | 33418 > https [ACK] Seq=2396 Ack=3816 Win=14528 Len=0 TSV=120639 TSER=120561 |
| TLSv1 | Application Data, Application Data |
| TCP | https > 33418 [ACK] Seq=3816 Ack=2598 Win=14480 Len=0 TSV=120564 TSER=120642 |
| TLSv1 | Application Data, Application Data |
| TLSv1 | Application Data, Application Data |
| TLSv1 | Application Data, Application Data |
| TLSv1 | Application Data |
| TLSv1 | Application Data |
| TLSv1 | Application Data |
| TLSv1 | Application Data |
| TCP | 33418 > https [ACK] Seq=2794 Ack=4230 Win=17424 Len=0 TSV=120648 TSER=120566 |
| TLSv1 | Application Data |
| TCP | 33418 > https [ACK] Seq=2794 Ack=4299 Win=17424 Len=0 TSV=120648 TSER=120570 |
| TLSv1 | Application Data |
| TLSv1 | Application Data |
| TLSv1 | Application Data |
| TLSv1 | Application Data |
| TLSv1 | Application Data |

**Рис. 35. Установление SSTP-соединения в режиме моста.**

Деактивируем SSTP -клиента, нажимая в winbox R1->PPP значок X на соответствующей строке. Деактивируем SSTP сервер (winbox R0->PPP), снимая галочку enable в его настройках.

**OPENVPN**

Активируем сервер OPENVPN R0 (winbox R0->PPP), установив галочку enable в его настройках. Активируем клиент OPENVPN R1 (winbox R0->PPP), нажав на строке Ovpn правой кнопкой и выбрав enable. И на клиенте R1 и на сервере R0 у OPENVPN-интерфейсов должна появиться метка R, а эти интерфейсы автоматически добавятся в мост bridge1. См. рис. 36. R3 видит R2 по IP и наоборот. Проверьте.

OPENVPN–сервер использует профиль default в котором для него необходимо указать локальный и удалённый адреса. Мы взяли адреса 192.168.100.1 и 192.168.200.1, которые никак не влияют на созданный мост. Эти адреса OPENVPN назначил на OPENVPN-интерфейсы. Посмотрите в OPENVPN –сервере R0 (winbox IP->adresses). См. рис. 37.

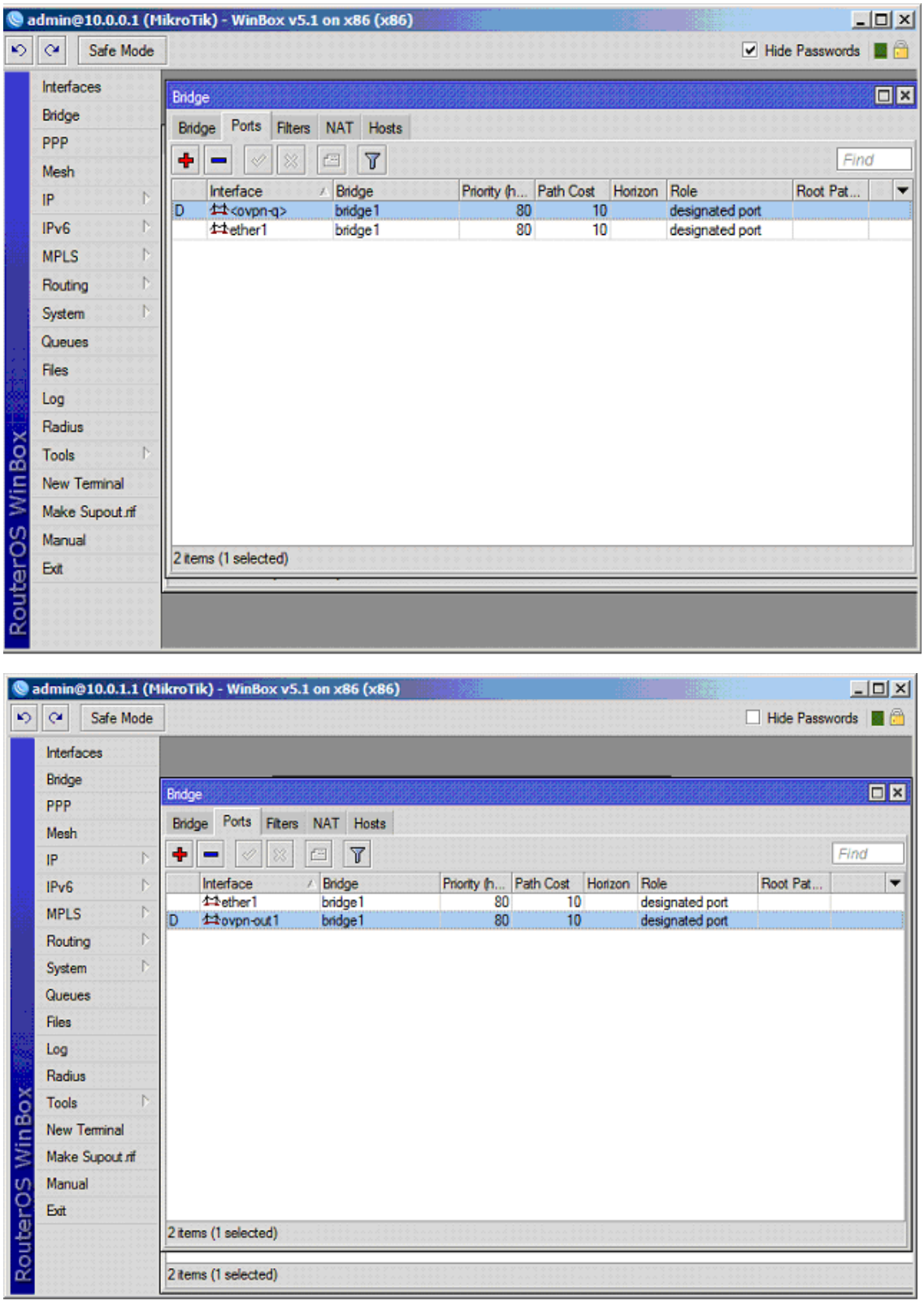



**Рис. 36. Мосты в OPENVPN**

OPENVPN–сервер использует профиль default в котором для него необходимо указать локальный и удалённый адреса. Мы взяли адреса 192.168.100.1 и 192.168.200.1, которые

никак не влияют на созданный мост. Эти адреса OPENVPN-сервер назначил на OPENVPN-интерфейсы. Посмотрите в OPENVPN –сервере R0 (winbox IP->adresses). См. рис. 37. Пропингуйте из R0 R1 и из этим адресам.

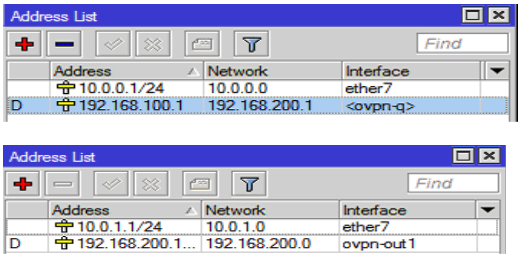



**Рис. 37. OPENVPN назначил адреса для клиента и сервера**

| | |
|---|---|
| TCP | 49779 > openvpn [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSV=132074 TSER=0 WS=4 |
| TCP | openvpn > 49779 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSV=131996 T |
| TCP | 49779 > openvpn [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=132074 TSER=131996 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=16 TSV=132074 TSER=131996 |
| TCP | openvpn > 49779 [ACK] Seq=1 Ack=17 Win=5792 Len=0 TSV=131997 TSER=132074 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=1 Ack=17 Win=5792 Len=16 TSV=131997 TSER=132074 |
| TCP | 49779 > openvpn [ACK] Seq=17 Ack=17 Win=5840 Len=0 TSV=132075 TSER=131997 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=17 Ack=17 Win=5840 Len=28 TSV=132075 TSER=131997 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=17 Ack=17 Win=5792 Len=24 TSV=131997 TSER=132075 |
| TCP | openvpn > 49779 [ACK] Seq=41 Ack=45 Win=5792 Len=0 TSV=132001 TSER=132075 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=45 Ack=41 Win=5840 Len=118 TSV=132078 TSER=131997 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=41 Ack=163 Win=5792 Len=24 TSV=132001 TSER=132078 |
| TCP | 49779 > openvpn [ACK] Seq=163 Ack=65 Win=5840 Len=0 TSV=132083 TSER=132001 |
| TCP | openvpn > 49779 [ACK] Seq=65 Ack=163 Win=5792 Len=1448 TSV=132005 TSER=132078 |
| TCP | 49779 > openvpn [ACK] Seq=163 Ack=1513 Win=8736 Len=0 TSV=132083 TSER=132005 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=1513 Ack=163 Win=5792 Len=1229 TSV=132005 TSER=132083 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=163 Ack=1513 Win=8736 Len=24 TSV=132083 TSER=132005 |
| TCP | openvpn > 49779 [ACK] Seq=2742 Ack=187 Win=5792 Len=0 TSV=132009 TSER=132083 |
| TCP | 49779 > openvpn [ACK] Seq=187 Ack=2742 Win=11632 Len=0 TSV=132087 TSER=132005 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=187 Ack=2742 Win=11632 Len=24 TSV=132087 TSER=132009 |
| TCP | openvpn > 49779 [ACK] Seq=2742 Ack=211 Win=5792 Len=0 TSV=132009 TSER=132087 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=211 Ack=2742 Win=11632 Len=1416 TSV=132094 TSER=132009 |
| TCP | openvpn > 49779 [ACK] Seq=2742 Ack=1627 Win=8688 Len=0 TSV=132016 TSER=132094 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=1627 Ack=2742 Win=11632 Len=945 TSV=132094 TSER=132016 |
| TCP | openvpn > 49779 [ACK] Seq=2742 Ack=2572 Win=11520 Len=0 TSV=132017 TSER=132094 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=2742 Ack=2572 Win=11520 Len=24 TSV=132017 TSER=132094 |
| TCP | 49779 > openvpn [ACK] Seq=2572 Ack=2766 Win=11632 Len=0 TSV=132095 TSER=132017 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=2766 Ack=2572 Win=11520 Len=24 TSV=132017 TSER=132095 |
| TCP | 49779 > openvpn [ACK] Seq=2572 Ack=2790 Win=11632 Len=0 TSV=132095 TSER=132017 |
| TCP | openvpn > 49779 [PSH, ACK] Seq=2790 Ack=2572 Win=11520 Len=1210 TSV=132022 TSER=132095 |
| TCP | 49779 > openvpn [ACK] Seq=2572 Ack=4000 Win=14528 Len=0 TSV=132100 TSER=132022 |
| TCP | 49779 > openvpn [PSH, ACK] Seq=2572 Ack=4000 Win=14528 Len=24 TSV=132100 TSER=132022 |

**Рис. 38. Установка соединения и обмен данными в OPENVPN**

Для остальных 4-х соединений PPP, PPTP, L2TP и SSTP адреса 192.168.100.1 и 192.168.200.1 в профиле default не используется.

В OPENVPN установка соединения и обмен данными происходит в зашифрованном виде и мы ничего не увидим в анализаторе пакетов. См. рис. 38.

Деактивируем OPENVPN -клиента, нажимая в winbox R1->PPP значок X на соответствующей строке. Деактивируем OPENVPN сервер (winbox R0->PPP), снимая галочку enable в его настройках.

## Особенности работы из командной строки

База данных профилей управляется из подменю

**/ppp profile**

Профили PPP используются для определения значений по умолчанию для записи пользователя, определённой с помощью подменю /ppp secret. Установки в /ppp secret подавляют соответствующие установки, сделанные па подменю /ppp profile, за исключением того, что назначение единственного IP-адреса всегда имеет приоритет над адресом, взятым из назначенного пула IP-адресов. Значения свойств профиля PPP приведены в табл. 1.

Таблица 1.

Значения свойств профиля PPP

| Свойство | Значения | Описание |
|---|---|---|
| **address-list** | строка | Имя списка адресов к которому будет добавлен адрес, назначенный PPP |
| **bridge** | строка | Имя моста к которому будет добавлен PPP-интерфейс как порт. |
| **change-tcp-mss** | yes \| no \| default | Изменить MSS (Maximum Segment Size) соединения<br>yes– подогнать MSS соединения<br>no – не подгонять MSS соединения<br>default – взять это значение из профиля по умолчанию для интерфейса; no если это и есть профиль по умолчанию<br>Используется для избежания фрагментации пакетов. |
| **dns-server** | IP | IP-адрес DNS-сервера, назначаемый клиенту |
| **idle-timeout** | | Допустимое время неактивности, после которого линк будет разорван |
| **incoming-filter** | строка | Имя цепочки фаервола для входящих пакетов, которая берёт контроль над каждым пакетом, приходящим от клиента. Цепочка должна быть добавлена вручную |
| **local-address** | IP | IP-адрес или имя пула IP-адресов для PPP-сервера |
| **name** | строка | Имя этого профиля |
| **only-one** | yes \| no \| default | Определяет разрешено ли пользователю иметь боле одного соединения<br>yes - нельзя<br>no - можно<br>default - взять это значение из профиля по умолчанию для интерфейса |
| **outgoing-filter** | строка | Имя цепочки фаервола для исходящих пакетов, которая берёт контроль над каждым пакетом, идущим к клиенту. Цепочка должна быть |

| | | |
|---|---|---|
| | | добавлена вручную. |
| **rate-limit** | строка | Ограничение скорости |
| **remote-address** | IP | IP-адрес или имя пула IP-адресов для PPP-клиента |
| **session-timeout** | | Максимальное время, которое соединения может оставаться активным |
| **use-compression** | yes \| no \| default | Определяет, используется ли сжатие данных<br>yes - да<br>no - нет<br>default - взять это значение из профиля по умолчанию для интерфейса; no если это и есть профиль по умолчанию |
| **use-encryption** | yes \| no \| default \| required | Определяет, используется ли шифрование данных<br>yes - да<br>no - нет<br>default - взять это значение из профиля по умолчанию для интерфейса; no если это и есть профиль по умолчанию<br>require - явно требовать шифрования |
| **use-ipv6** | yes \| no \| default \| required | Определяет, разрешён ли IPv6. По умолчанию разрешён, если IPv6 установлен.<br>yes - да<br>no - нет<br>default - взять это значение из профиля по умолчанию для интерфейса; no если это и есть профиль по умолчанию<br>require - явно требовать поддержки IPv6 |
| **use-mpls** | yes \| no \| default \| required | Определяет, разрешён ли MPLS через PPP.<br>yes - да<br>no - нет<br>default - взять это значение из профиля по умолчанию для интерфейса; no если это и есть профиль по умолчанию<br>require - явно требовать поддержки MPLS |
| **use-vj-compression** | yes \| no \| default | Определяет, используется ли алгоритм сжатия заголовков Van Jacobson<br>yes - да<br>no - нет<br>default - взять это значение из профиля по умолчанию для интерфейса; no если это и есть профиль по умолчанию |
| **wins-server** | IP | IP-адрес WINS-сервера, назначаемый клиенту |

Есть два профиля по умолчанию, которые нельзя удалить
**/ppp profile print**
Flags: * - default

0 * name="default" use-compression=no use-vj-compression=no use-encryption=no only-one=no change-tcp-mss=yes

1 * name="default-encryption" use-compression=default use-vj-ompression=default use-encryption=yes only-one=default change-tcp-mss=default

База данных пользователей управляется из подменю

**/ppp secret**

Значения свойств пользователей PPP приведены в табл. 2.

Таблица 2.

Значения свойств пользователей PPP

| **Свойство** | Значения | Описание |
|---|---|---|
| **caller-id** | строка | Для PPTP и L2TPF это IP-адрес из которого клиент должен соединяться. Для PPPoE это MAC-адрес из которого клиент должен соединяться. Используется только при необходимости |
| **limit-bytes-in** | целое | Максимальное число байт, которые может выгрузить клиент |
| **limit-bytes-out** | целое | Максимальное число байт, которые может загрузить клиент |
| **local-address** | IP | IP-адрес или имя пула IP-адресов для PPP-сервера |
| **name** | строка | Имя пользователя для аутентификации |
| **password** | строка | Пароль для аутентификации |
| **profile** | строка | Какой профиль пользователя использовать |
| **remote-address** | IP | IP-адрес или имя пула IP-адресов для PPP-клиента |
| **remote-ipv6-prefix** | IPv6 | Префикс *IPv6* для PPP-клиента |
| **routes** | строка | Маршруты,которые появятся на сервере, когда клиент соединится. Формат маршрута: dst-address gateway metric (например, 10.1.0.0/ 24 10.0.0.1 1). Несколько маршрутов разделяются запятыми. Игнорируется для OpenVPN |
| **service** | any \| async \| isdn \| l2tp \| pppoe \| pptp \| ovpn | Оределяет сервисы,доступные этому пользователю |

Проанализируем общие параметры настройки PPP, PPTP, L2TP, SSTP, PPOE и OPENVPN серверов и клиентов. Общие параметры серверов приведены в табл. 3.

Общие параметры серверов Таблица 3.

| | **l2tp** | **ovpn** | **ppp** | **pppoe** | **pptp** | **sstp** |
|---|---|---|---|---|---|---|
| **max-mtu** | + | + | + | + | + | + |
| **max-mru** | + | | + | + | + | + |
| **mrru** | + | | + | + | + | + |
| **authentication** | + | | + | + | + | + |
| **keepalive-timeout** | | + | | + | + | + |
| **certificate** | | + | | | | + |
| **verify-client-certificate** | | + | | | | + |

| | | | | | | |
|---|---|---|---|---|---|---|
| **port** | | + | | | | + |

Общие параметры клиентов приведены в табл. 4.

Общие параметры клиентов Таблица 4.

| | **l2tp** | **ovpn** | **ppp** | **pppoe** | **pptp** | **sstp** |
|---|---|---|---|---|---|---|
| **max-mtu** | + | + | + | + | + | + |
| **max-mru** | + | | + | + | + | + |
| **mrru** | + | | + | + | + | + |
| **connect-to** | + | + | | | + | + |
| **User,password** | + | + | + | + | + | + |
| **add-default-route** | + | + | + | + | + | + |
| **dial-on-demand** | + | | + | + | + | + |
| **authentication** | + | | + | + | + | + |
| **port** | | + | | | | + |
| **certificate** | | + | | | | + |
| **use-peer-dns** | | | + | + | | |
| **keepalive-timeout** | + | | | | | + |

Объяснение значений свойств из табл. 3 и табл. 4 приведены в табл. 5.

Объяснение значений свойств клиентов и серверов Таблица. 5.

| **Свойство** | **Пояснение** |
|---|---|
| **max-mtu** | Максимальный размер пакета, передаваемый без фрагментации. Для эсернет-1500 |
| **max-mru** | Максимальный размер пакета, принимаемый без фрагментации Для эсернет-1500 |
| **mrru** | Максимальный размер принимаемого пакета. Позволяет передавать эсернет-фреймы через туннель. Если пакет больше, чем **max-mru** он будет разбит. |
| **authentication** | Использовать протокол проверки подлинности mschap2, mschap1, chap или pap |
| **keepalive-timeout** | Время активности |
| **certificate** | Имя сертификата |
| **verify-client-certificate** | Проверять ли сертификат клиента |
| **port** | TCP-порт |
| **connect-to** | Адрес для соединения |
| **User,password** | Имя и пароль |
| **add-default-route** | Добавить маршрут по умолчанию |
| **dial-on-demand** | Обратный вызов по требованию |
| **use-peer-dns** | Испоьзовать DNS пира |

Значения max-mtu и max-mtu могут быть меньше, чем 1500. Например, PPPoE требует для себя дополнительные 6 байт. 2 байта добавляет само PPP. Остаётся 1492 байта для IP-датаграмы. Следовательно, max-mtu и max-mtu для PPPoE не могут превышать 1492.

Сейчас мы поэкспериментируем с настройками через командную строку. Сделайте топологию из одного маршрутизатора. Добавим профиль q

[admin@MikroTik] > **ppp profile add**
name: **q**

Имеем

[admin@MikroTik] > **ppp profile print detail**
name="q" use-ipv6=yes use-mpls=default use-compression=default use-vj-compression=default use-encryption=default only-one=default change-tcp-mss=default

Пример добавления пользователя q

[admin@MikroTik] > **ppp secret add**
name: **q**

Имеем

[admin@MikroTik] > **ppp secret print detail**
name="q" service=any caller-id="" password="" profile=default routes="" limit-bytes-in=0 limit-bytes-out=0

Есть встроенные сервера PPTP, L2TP, SSTP и OPENVPN

[admin@MikroTik] > **interface pptp-server server print**
enabled: yes
max-mtu: 1460
max-mru: 1460
mrru: disabled
authentication: mschap1,mschap2
keepalive-timeout: 30
default-profile: default
[admin@MikroTik] > **interface l2tp-server server print**
enabled: no
max-mtu: 1460
max-mru: 1460
mrru: disabled
authentication: pap,chap,mschap1,mschap2
default-profile: default-encryption
[admin@MikroTik] > **interface sstp-server server print**
enabled: no
port: 443
max-mtu: 1500
max-mru: 1500
mrru: disabled
keepalive-timeout: 60
default-profile: default
authentication: pap,chap,mschap1,mschap2
certificate: none
verify-client-certificate: no
[admin@MikroTik] > **interface ovpn-server server print**
enabled: no
port: 1194
mode: ip
netmask: 24

```
            mac-address: FE:8C:42:2D:6D:A2
                max-mtu: 1500
        keepalive-timeout: 60
          default-profile: default
              certificate: none
 require-client-certificate: no
                    auth: sha1,md5
                  cipher: blowfish128,aes128
```

Активация серверов. Активация сервера PPP

```
[admin@MikroTik] > interface ppp-server add
```

Активация серверов l PPTP, L2TP, SSTP и OPENVPN

```
[admin@MikroTik] > interface  l2tp-server server set enabled=yes
[admin@MikroTik] > interface  ovpn-server server set enabled=yes
[admin@MikroTik] > interface  pptp-server server set enabled=yes
[admin@MikroTik] > interface  sstp-server server set enabled=yes
```

Активация сервера PPPOE

```
[admin@MikroTik] > interface pppoe-server server add interface=ether1 disabled=no
[admin@MikroTik] > interface pppoe-server server print
Flags: X - disabled
 0 X service-name="" interface=ether1 max-mtu=1480 max-mru=1480 mrru=disabled
     authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10
     one-session-per-host=no max-sessions=0 default-profile=default
```

После активации и соединении клиентов автоматически создаются соответствующие интерфейсы.

Для ручного создания серверов служат команды

**interface l2tp-server add**
**interface ovpn-server add**
**interface pppoe-server add**
**interface pptp-server add**
**interface sstp-server add**

при этом запрашивается для какого пользователя создаётся интерфейс, а для PPPOE ещё и имя службы.

Перейдём к выяснению параметров необходимых для добавления клиентов

```
[admin@MikroTik] > int l2tp-client add
connect-to: 10.0.0.1
user: q
[admin@MikroTik] > int l2tp-client pr det
Flags: X - disabled, R - running
 0 X  name="l2tp-out1" max-mtu=1460 max-mru=1460 mrru=disabled
      connect-to=10.0.0.1 user="q" password="" profile=default-encryption
      add-default-route=no dial-on-demand=no allow=pap,chap,mschap1,mschap2
[admin@MikroTik] > int ovpn-client add
connect-to: 10.0.0.1
user: q
[admin@MikroTik] > int ovpn-client pr det
Flags: X - disabled, R - running
```

```
 0   name="ovpn-out1" mac-address=FE:F8:51:31:89:E0 max-mtu=1500
     connect-to=10.0.0.1 port=1194 mode=ip user="q" password=""
     profile=default certificate=none auth=sha1 cipher=blowfish128
     add-default-route=no
[admin@MikroTik] > int ppp-client add
[admin@MikroTik] > int ppp-client pr det
Flags: X - disabled, R - running
 0  R name="ppp-out1" max-mtu=1500 max-mru=1500 mrru=disabled port=serial1
     data-channel=0 info-channel=0 pin="" user="0" password="0"
     profile=default phone="" dial-command="ATDT" modem-init=""
     null-modem=yes dial-on-demand=no add-default-route=no use-peer-dns=no
     keepalive-timeout=10 allow=pap,chap,mschap1,mschap2
[admin@MikroTik] > int pppoe-client add interface=ether1
[admin@MikroTik] > int pppoe-client  pr detail
Flags: X - disabled, R - running
 0 X  name="pppoe-out1" max-mtu=1480 max-mru=1480 mrru=disabled
     interface=ether1 user="" password="" profile=default service-name=""
     ac-name="" add-default-route=no dial-on-demand=no use-peer-dns=no
     allow=pap,chap,mschap1,mschap2
[admin@MikroTik] > int pptp-client add
connect-to: 10.0.0.1
user: q
[admin@MikroTik] > int pptp-client pr det
Flags: X - disabled, R - running
 0 X  name="pptp-out1" max-mtu=1460 max-mru=1460 mrru=disabled
     connect-to=10.0.0.1 user="q" password="" profile=default-encryption
     add-default-route=no dial-on-demand=no allow=pap,chap,mschap1,mschap2
[admin@MikroTik] > int sstp-client add
connect-to: 10.0.0.1
user: q
[admin@MikroTik] > int sstp-client print detail
Flags: X - disabled, R - running
 0 X  name="sstp-out1" max-mtu=1500 max-mru=1500 mrru=disabled
     connect-to=10.0.0.1 :443 http-proxy=0.0.0.0:443 certificate=none
     verify-server-certificate=no user="q" password="" profile=default
     keepalive-timeout=60 add-default-route=no dial-on-demand=no
     authentication=pap,chap,mschap1,mschap2
```

Смена параметров производится так

```
[admin@MikroTik] > int sstp-client set 0 profile=?
Default default-encription
[admin@MikroTik] > int sstp-client set 0 profile= default-encription
```

Здесь 0 –это номер порции информации, которую выдала команда

```
[admin@MikroTik] >  int sstp-client print detail
```

Применим полученные навыки. Добавьте в топологию на рис. 23 новые 4-е устройства qemu host. Назначьте адреса и имена согласно рис. 39. Обеспечьте подсоединение к псевдоинтернету (студент D=0)

```
[admin@ R4] >ip add addr add address=10.0.4.1/24 int=ether7
```

[admin@ R4] >**ip ro add dst-address=10.0.0.0/16 gateway=10.0.4.2**
[admin@ R5] >**ip add addr add address=10.0.5.1/24 int=ether7**
[admin@ R5] >**ip ro add dst-address=10.0.0.0/16 gateway=10.0.5.2**
[admin@ R6] >**ip add addr add address=10.0.6.1/24 int=ether7**
[admin@ R6] >**ip add addr add address=172.16.2.1/24int=ether1**
[admin@ R7] >**ip add addr add address=10.0.7.1/24 int=ether7**
[admin@ R7] >**ip add addr add address=172.16.2.2/24 int=ether1**

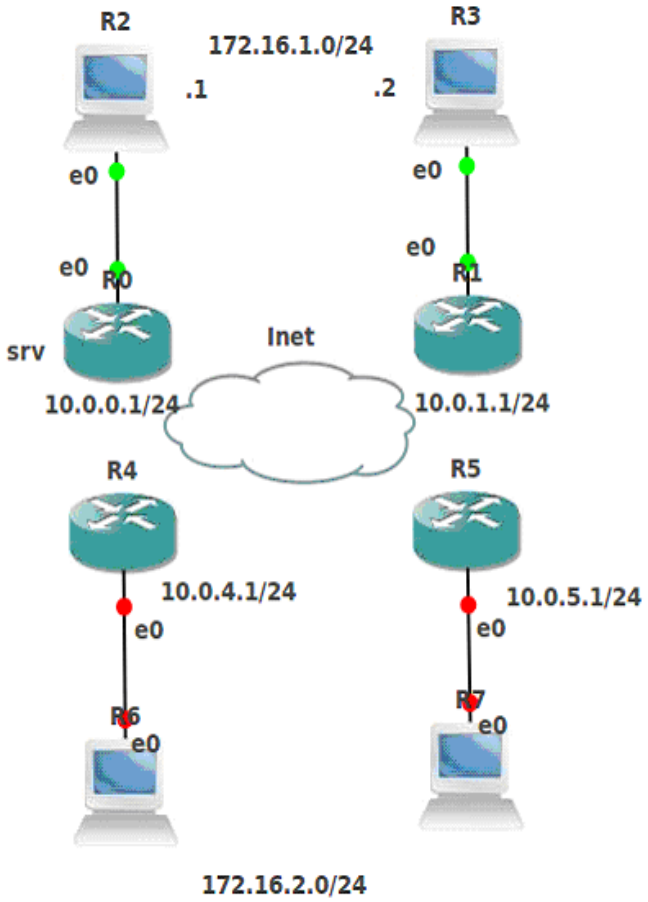



**Рис. 39. Топология l2vpn**

Повторим настройки из командной строки и без помощи winbox для новых маршрутизаторов. Очень полезна техника copy-paste.

На R4 и R5 добавим мост в профили по-умолчанию

**int br add**
**int bridge port add bridge=bridge1 interface=ether1**
**ppp profile print**
**ppp profile set 0,1 bridge=bridge1**

Добавим пользователя с паролем

[admin@ R4] >**ppp secret add name=q password=q**

**PPP**

Запускаем PPP сервер

[admin@ R4] >**int ppp-server add port=serial1 null-modem=yes disabled=no**

Запускаем PPP клиент

[admin@ R5] >**interface ppp-client add user=q password=q disabled=no dial-on-demand=no null-modem=yes port=serial1 use-peer-dns=no add-default-route=no**

Видим поднявшиеся PPP – интерфейсы (буква R). Клиент

[admin@ R5] > **interface ppp-clien pr**

```
 Flags: X - disabled, R - running
 0  R name="ppp-out1" max-mtu=1500 max-mru=1500 mrru=disabled port=serial1
     data-channel=0 info-channel=0 pin="" user="q" password="q"
     profile=default phone="" dial-command="ATDT" modem-init=""
     null-modem=yes dial-on-demand=no add-default-route=no use-peer-dns=no
     keepalive-timeout=10 allow=pap,chap,mschap1,mschap2
```

Сервер

[admin@ R4] > **int ppp-serve pr det**

```
Flags: X - disabled, R - running
 0  R name="ppp-in1" max-mtu=1500 max-mru=1500 mrru=disabled port=serial1
     data-channel=0 authentication=pap,chap,mschap1,mschap2 profile=default
     modem-init="" ring-count=1 null-modem=yes
```

И в R4 и в R5 в мост добавился интерфейс, хотя и с неизвестным именем, например

[admin@ R4] **> interface bridge port print**

```
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE           BRIDGE           PRIORITY PATH-COST  HORIZON
 0    ether1              bridge1          0x80     10         none
 1  D (unknown)           bridge1          0x80     10         none
```

R6 и R7 видят друг друга по IP. Проверьте это утилитой ping.

Остановим сервер и клиент

[admin@ R4] **>int ppp-server set 0 disabled=yes**

[admin@ R5] **>int ppp-client set 0 disabled=yes**

**PPTP**

Запускаем PPTP сервер R4

[admin@ R4] **>interface pptp-server server set enabled=yes**

Запускаем PPTP клиент R5

[admin@ R5] **>int pptp-client add user=q password=q connect-to=10.0.4.1 disabled=no**

Видим поднявшиеся PPP – интерфейсы (буква R). Сервер

[admin@ R4] **>interface pptp-serve pr detail**

```
Flags: X - disabled, D - dynamic, R - running
 0  DR name="<pptp-q>" user="q" mtu=1460 mru=1460 client-address="10.0.5.1"
      uptime=4m29s encoding="MPPE128 stateless"
```

Клиент

[admin@ R5] > **int pptp-client print detail**

```
Flags: X - disabled, R - running
 0  R name="pptp-out1" max-mtu=1460 max-mru=1460 mrru=disabled
     connect-to=10.0.4.1 user="q" password="q" profile=default-encryption
```

add-default-route=no dial-on-demand=no allow=pap,chap,mschap1,mschap2

Команда **interface bridge port print покажет, что** и в R4 и в R5 в мост добавился интерфейс. R6 и R7 видят друг друга по IP. Проверьте это утилитой ping.

Остановим сервер и клиент

[admin@ R4] **>int pptp-server set 0 disabled=yes**
[admin@ R5] **>int pptp-client set 0 disabled=yes**

**L2TP**

L2TP полностью аналогичен PPTP (поменяйте в командах pptp на l2tp)

**SSTP**

Надо импортировать сертификаты. Сертификаты находятся в папке easy-rsa/keys. Перейдите в неё. Перепишем сертификаты и ключ из Ubuntuв SSTP-сервер.

ftp 10.0.4.1
name: admin
password;
bin
put ca.crt
put server.crt
put server.key
quit

Перепишем сертификаты и ключ в SSTP- клиент.

ftp 10.0.5.1
name: admin
password;
bin
put ca.crt
put client.crt
put client.key
quit

Импортируем сертификаты в сервере

[admin@ R4] **> certificate import file-name=ca.crt**
[admin@ R4] **> certificate import file-name=server.crt**
[admin@ R4] **> certificate import file-name=server.key**

На запрос passphrase – просто жмём enter

Проверим

[admin@ R4] **> certificate pr**

```
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
 0    name="cert1" subject=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=Fort-
       Funston CA,emailAddress=me@myhost.mydomain
      issuer=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=Fort-Funston CA,
       emailAddress=me@myhost.mydomain
      serial-number="C8A494A29F4DC49F" email=me@myhost.mydomain
```

invalid-before=may/11/2011 15:34:56 invalid-after=may/08/2021 15:34:56
ca=yes
1 KR name="cert2" subject=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=server,
emailAddress=me@myhost.mydomain
issuer=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=Fort-Funston CA,
emailAddress=me@myhost.mydomain
serial-number="01" email=me@myhost.mydomain
invalid-before=may/11/2011 15:35:13 invalid-after=may/08/2021 15:35:13
ca=yes

Переименовываем KR сертификат

[admin@ R4] **> syscertificate set 1 name=srv**

Импортируем сертификаты у клиента

[admin@ R5] **>certificate import file-name=ca.crt**
[admin@ R5] **>certificate import file-name=client.crt**
[admin@ R5] **>certificate import file-name=client.key**

На запрос passphrase – просто жмём enter

Проверим

[admin@ R5] **>certificate print detail**
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
0 name="cert1" subject=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=Fort-
Funston CA,emailAddress=me@myhost.mydomain
issuer=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=Fort-Funston CA,
emailAddress=me@myhost.mydomain
serial-number="C8A494A29F4DC49F" email=me@myhost.mydomain
invalid-before=may/11/2011 15:34:56 invalid-after=may/08/2021 15:34:56
ca=yes
1 KR name="cert2" subject=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=client,
emailAddress=me@myhost.mydomain
issuer=C=US,ST=CA,L=SanFrancisco,O=Fort-Funston,CN=Fort-Funston CA,
emailAddress=me@myhost.mydomain
serial-number="02" email=me@myhost.mydomain
invalid-before=may/11/2011 15:35:33 invalid-after=may/08/2021 15:35:33
ca=yes

Переименовываем KR сертификат клиента

[admin@ R5] >**certificate set 1 name=cli**

Запускаем SSTP сервер R4

[admin@R4]**>interface sstp-server server set enabled=yes certificate=srv verify-client-certificate=yes**

Запускаем SSTP клиент R5

[admin@R4]**>int sstp-client add user=q password=q connect-to=10.0.4.1 disabled=no certificate=cli verify-server-certificate=yes**

Интерфейсы поднялись. И в R4 и в R4 в мост добавился интерфейс. R6 R7 видят друг друга по IP. Проверьте всё это.

Остановим сервер и клиент

[admin@ R4] **>int sstp-server set 0 disabled=yes**
[admin@ R5] **>int sstp-client set 0 disabled=yes**

**OPENVPN**

Пропишем в профиле сервера локальный и удалённый адреса

```
[admin@ R4] >ppp profile pr
Flags: * - default
 0 * name="default" bridge=bridge1 use-ipv6=yes use-mpls=default
     use-compression=default use-vj-compression=default
     use-encryption=default only-one=default change-tcp-mss=yes
1 * name="default-encryption" bridge=bridge1 use-ipv6=yes use-mpls=default
     use-compression=default use-vj-compression=default use-encryption=yes
     only-one=default change-tcp-mss=yes
```

**[admin@ R4] >ppp profile set 0 local-address=192.168.100.1 remote-address= 192.168.200.1**

Запускаем OPENVPN сервер R4

**[admin@ R4] >interface ovpn-server server set enabled=yes certificate=srv require-client-certificate=yes mode=Ethernet**

Запускаем OPENVPN клиент R5

**[admin@ R5] >interface ovpn-client add user=q password=q connect-to=10.0.4.1 disabled=no certificate=cli mode=Ethernet**

Интерфейсы поднялись. И в R4 и в R4 в мост добавился интерфейс

```
[admin@ R4] >interface bridge port print  detail
Flags: X - disabled, I - inactive, D - dynamic
 0    interface=ether1 bridge=bridge1 priority=0x80 path-cost=10 edge=auto
      point-to-point=auto external-fdb=auto horizon=none
 1  D interface=<ovpn-q> bridge=bridge1 priority=0x80 path-cost=10 edge=no
      point-to-point=yes external-fdb=no horizon=none
[admin@ R5] >interface bridge port print detail
Flags: X - disabled, I - inactive, D - dynamic
 0    interface=ether1 bridge=bridge1 priority=0x80 path-cost=10 edge=auto
      point-to-point=auto external-fdb=auto horizon=none
1  D interface=ovpn-out1 bridge=bridge1 priority=0x80 path-cost=10 edge=no
      point-to-point=yes external-fdb=no horizon=none
```

R6 R7 видят друг друга по IP. Проверьте всё это.

Остановим сервер и клиент

**[admin@ R4] >int ovpn-server set 0 disabled=yes**
**[admin@ R5] >int ovpn-client set 0 disabled=yes**

## Распределённый мост

Рассмотрим более сложные варианты использования мостов. Возьмём протокол PPTP. Для остальных конфигурация аналогична. Соберём топологию, изображённую на Рис. 40. Назначьте адреса согласно рисунку Рис. 40. Во всех маршрутизаторах R0, R1, R4, R5 добавим мосты и в них эсернет-интерфейс, идущий к подсоединённому компьютеру R2, R3, R6, R7.

**interface bridge add**
**interface bridge port add**
**bridge: bridge1**

**interface: ether1**

Маршрутизаторы R0, R1, R4 будут PPTP серверами, а маршрутизаторы R1, R4, R5 - PPTP клиентами.

Определим в серверах пользователя q с паролем q.

**ppp secret add name=q password=q**

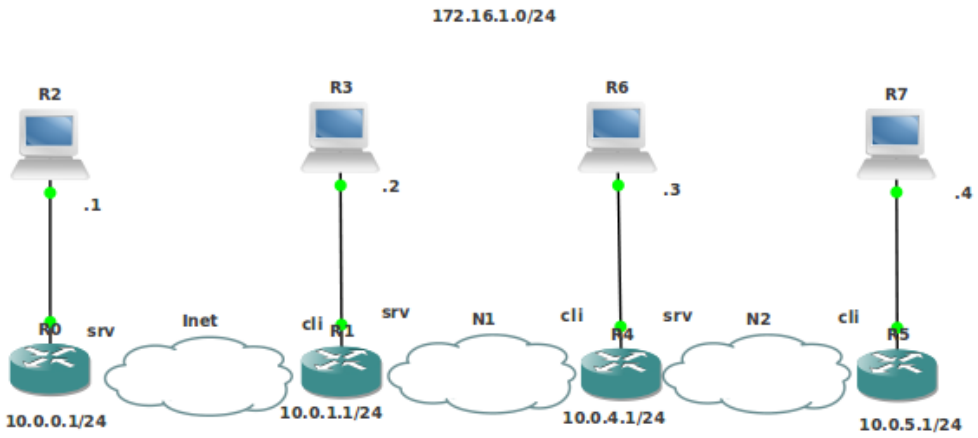



**Рис. 40. Топология l2vpn1**

По умолчанию сервера и клиенты имеют профиль default encription. Пользователь по умолчанию имеет профиль default. Профиль пользователя подавляет профиль сервера. В каком профиле определить мост? И на серверах и на клиентах определим мост в профиле default.

**ppp profile set 0 bridge=bridge1**

Здесь 0 - номер профиля default, который можно увидеть из команды **ppp profile print**

В самих клиентах заменим профиль default encription на профиль default, зададим пользователя q с паролем, зададим адреса серверов и активируем их

для R1 на 10.0.0.1 (R0)

[admin@R1]**>interface pptp-client add profile=default user=q password=q connect-to=10.0.0.1 disabled=no**

[admin@R4]**>interface pptp-client add profile=default user=q password=q connect-to=10.0.1.1 disabled=no**

[admin@R5]**>interface pptp-client add profile=default user=q password=q connect-to=10.0.4.1 disabled=no**

Активируем сервера.

**interface pptp-server server set enabled=yes**

Во всех маршрутизаторах R0, R1, R4, R5 в мостах появятся новые динамические PPTP интерфейсы. Причем в маршрутизаторах R1, R4 их будет два.

**interface bridge port print**

Получился распределённый мост (или свич): все компьютеры R2, R3, R6, R7 лежат в одной эсернет-сети.

Пропингуйте крайние компьютеры R2, R7.

Повторите всё для протоколов L2TP, SSTP и OPENVPN.

## Использование профилей пользователя.

Соберём топологию, изображённую на рис. 41. В ней адреса компьютеров R2 и R3 лежат в сети 172.16.1.0/24. Адреса компьютеров R6 и R7 лежат в той же сети 172.16.1.0/24. Назначьте адреса согласно рисунку. Назначьте командой **system identity set name=** имена согласно рисунку.

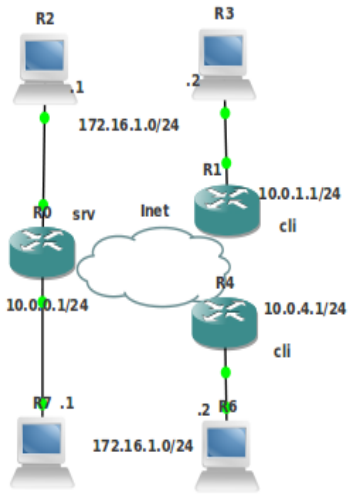



**Рис. 41. Топология l2vpn2**

Маршрутизатор R0 будет PPTP-сервером, а маршрутизаторы R1, R4- PPTP клиентами. Во всех клиентах R1, R4 добавьте мосты и в них эсернет- интерфейс, идущий к подсоединённому компьютеру. На клиентах в профиле default определим мост

**ppp profile set 0 bridge=bridge1**

Здесь 0 - номер профиля default, который можно увидеть из команды ppp profile print

На сервере добавим два моста и в каждый из них добавим по одному эсернет интерфейсу идущему к разным компьютерам. Пусть ether1 идёт к R2, а ether2 идёт к R7.

[admin@R0]**>interface bridge add**
[admin@R0]**>interface bridge add**
(2 раза)
[admin@R0]**>interface bridge port add**
bridge: bridge1
interface: ether1
[admin@R0]**>interface bridge port add**
bridge: bridge2
interface: ether2

Создадим для моста bridge1 профиль 1, а для моста bridge2 профиль 2

[admin@R0]**>ppp profile add name=1 bridge=bridge1**
[admin@R0]**>ppp profile add name=2 bridge=bridge2**

Создадим пользователей 1 и 2 с профилями 1 и 2, соответственно

[admin@R0]**>ppp secret add name=1 password=1 profile=1**
[admin@R0]**>ppp secret add name=2 password=1 profile=2**

Активируем сервер

**interface pptp-server server set enabled=yes**

В самих клиентах заменим профиль default encription на профиль default, зададим разных пользователей с паролем, зададим адрес сервера и активируем их

[admin@R1]**>interface pptp-client add profile=default user=1 password=1 connect-to=10.0.0.1 disabled=no**

[admin@R4]**>interface pptp-client add profile=default user=2 password=2 connect-to=10.0.0.1 disabled=no**

Во всех маршрутизаторах R0, R1, R4 в мостах появятся новые динамические PPTP интерфейсы. Причем в маршрутизаторе R0 их будет 2 в разных мостах. Проверьте это командой

**interface bridge port print**

Получили два независимых распределённых виртуальных моста (свича): компьютеры R2, R3 лежат в одной эсернет-сети, а компьютеры R6, R7 лежат в другой эсернет-сети. Эти эсернет-сети никак не связаны, и в них можно назначать одинаковые адреса, например из сети 172.16.1.0.24. Проверим

[admin@R3]**>system telnet 172.16.1.1**

Попадём в R2. Выход ctrl-d.

[admin@R6]**>system telnet 172.16.1.1**

Попадём в R7. Выход ctrl-d.

Повторите всё для протоколов L2TP, SSTP и OPENVPN.

# VPN уровня 3

## Маршрутизация RIP

Соберём топологию, изображённую Рис. 42. Назначьте адреса согласно рисунку.

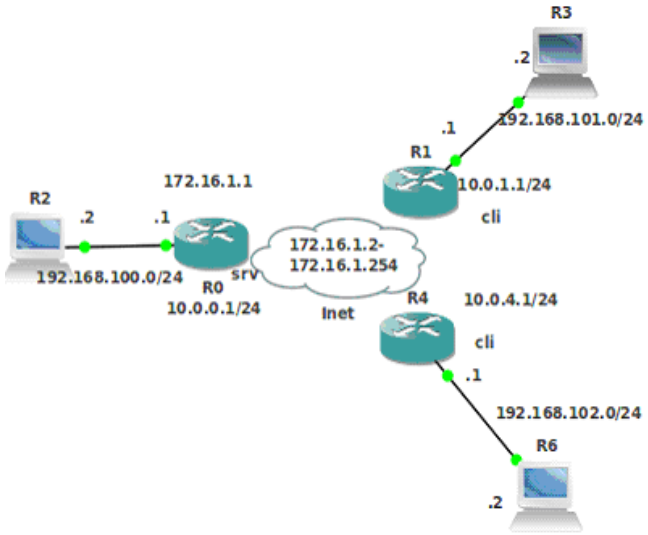



**Рис. 42. Топологии l3vpnrip, l3vpnrip2, l3vpnospf**

R0 является PPPT-сервером, а R3 и R6 - PPPT-клиентами. Поставим задачу так организовать маршрутизацию, чтобы компьютеры R2, R3 и R6 видели бы друг друга по IP.

Если мы не используем мосты то надо определится с адресами, назначаемыми на PPPTP-интерфейсы после установки PPPT-соединения.

Добавим в PPPT-сервере R0 пул адресов, назначаемых подсоединившимся PPPTP-клиентам

[admin@R0]**>ip pool add ranges=172.16.1.2-172.16.1.254 name=pool**

Пропишем это в профиле default. Мы будем назначать адрес 172.16.1.1 на интерфейс PPPT-сервера

[admin@R0]**>ppp profile set 0 local-address=172.16.1.1 remote-address=pool**

Не забудьте убрать из профиля мост (если он там остался). Создадим пользователя

[admin@R0]**>ppp secret add name=q password=q**

Активируем сервер

**interface pptp-server server set enabled=yes**

В самих клиентах R1  R4 зададим пользователя q с паролем и зададим адрес сервера

**interface pptp-client add profile=default user=q password=q connect-to=10.0.0.1 disabled=no**

На сервере появилось два  одинаковых адреса, но в разных сетях

[admin@R0]**>ip ad pr**

```
0   10.0.0.1/24       10.0.0.0       ether7
 1   192.168.100.1/24  192.168.100.0  ether1
 2 D 172.16.1.1/32     172.16.1.253   <pptp-q-1>
 3 D 172.16.1.1/32     172.16.1.254   <pptp-q>
```

На клиенте R1 появился адрес

[admin@ R1] **> ip ad pr**

```
0   10.0.1.1/24       10.0.1.0       ether7
 1   192.168.101.1/24  192.168.101.0  ether1
 2 D 172.16.1.254/32   172.16.1.1     pptp-out1
```

На клиенте R4 появился адрес

[admin@ R4] **> ip ad pr**

```
0   10.0.4.1/24       10.0.4.0       ether7
 1   192.168.102.1/24  192.168.102.0  ether1
 2 D 172.16.1.253/32   172.16.1.1     pptp-out2
```

Обратите внимание на маску назначенных адресов и сети. В нашей топологии фигурирует 3 сети с маской /24 192.168.100.0/24 192.168.101.0/24 192.168.102.0/24 и в общем случае переменное число сетей с маской /32. Это число зависит от количества клиентов. В нашем случае имеем 3 сети  172.16.1.1  172.16.1.253  172.16.1.254

Можно прописать маршрутизацию статически (сделайте это).

Воспользуемся протоколом RIP. Так как нельзя предсказать, какие адреса  будут назначены  клиентам будем оперировать сетью 172.16.1.0/24.

[admin@R0]**>routing rip network add network=172.16.1.0/24**
[admin@R0]**>routing rip network add network=192.168.100.0/24**
[admin@R1]**>routing rip network add network=172.16.1.0/24**
[admin@R1]**>routing rip network add network=192.168.101.0/24**
[admin@R4]**>routing rip network add network=172.16.1.0/24**
[admin@R4]**>routing rip network add network=192.168.102.0/24**

Посмотрим созданные RIP-маршруты

[admin@R4]**> ip ro pr**

```
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS       PREF-SRC       GATEWAY          DISTANCE
```

```
 0 A S  10.0.0.0/16                  10.0.4.2         1
 1 ADC  10.0.4.0/24       10.0.4.1       ether7           0
 2 ADC  172.16.1.1/32     172.16.1.253   pptp-out2        0
 3 ADr  172.16.1.254/32                  172.16.1.1       120
 4 ADr  192.168.100.0/24                 172.16.1.1       120
 5 ADr  192.168.101.0/24                 172.16.1.1       120
 6 ADC  192.168.102.0/24  192.168.102.1  ether1           0
```

Есть маршрут на сеть 192.168.101.0/24 компьютера R3

```
[admin@R1]> ip ro pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS       PREF-SRC       GATEWAY          DISTANCE
 0 A S  10.0.0.0/16                  10.0.1.2         1
 1 ADC  10.0.1.0/24       10.0.1.1       ether7           0
 2 ADC  172.16.1.1/32     172.16.1.254   pptp-out1        0
 3 ADr  172.16.1.253/32                  172.16.1.1       120
 4 ADr  192.168.100.0/24                 172.16.1.1       120
 5 ADC  192.168.101.0/24  192.168.101.1  ether1           0
 6 ADr  192.168.102.0/24                 172.16.1.1       120
```

Есть маршрут на сеть 192.168.102.0/24 компьютера R6.

Пропишем маршрут по умолчанию

[admin@R2]**>ip route add gateway = 192.168.100.1**
[admin@R3]**>ip route add gateway = 192.168.101.1**
[admin@R6]**>ip route add gateway = 192.168.102.1**

Теперь R3 увидит по IP R6 и наоборот.

Такой трюк с сетью 172.16.1.0/24 для OSPF не проходит

## Маршрутизация OSPF

Теперь переделаем конфигурацию для маршрутизации путём назначения каждому клиенту определённого адреса. Этого добьемся путем назначения каждому клиенту отдельного имени со своим профилем. Сделайте копию l3vpnospf топологии l3vpnrip. На сервере R0 создадим 2 профиля

[admin@R0]**>ppp profile add name=1 local-address=172.16.1.1 remote-address=172.16.1.2**
[admin@R0]**>ppp profile add name=2 local-address=172.16.1.1 remote-address=172.16.1.3**

Создадим пользователей 1 и 2 с профилями 1 и 2, соответственно

[admin@R0]**>ppp secret add name=1 password=1 profile=1**
[admin@R0]**>ppp secret add name=2 password=2 profile=2**

Активируем сервер

[admin@R0]**>interface pptp-server server set enabled=yes**

Убъём старых pptp клиентов на R1 R4

**interface pptp-client pr**
**interface pptp-client rem 0**

Добавим новых

[admin@R1]**>interface pptp-client add user=1 password=1 connect-to=10.0.0.1 disabled=no**
[admin@R4]**>interface pptp-client add user=2 password=2 connect-to=10.0.0.1 disabled=no**

На сервере появилось два одинаковых адреса, но в разных сетях

```
[admin@ R0] > ip ad pr
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   10.0.0.1/24        10.0.0.0        ether7
 1   192.168.100.1/24   192.168.100.0   ether1
 2 D 172.16.1.1/32      172.16.1.3      <pptp-2>
 3 D 172.16.1.1/32      172.16.1.2      <pptp-1>
```

На клиентах появились адреса

```
[admin@ R1] > ip ad pr
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   10.0.1.1/24        10.0.1.0        ether7
 1   192.168.101.1/24   192.168.101.0   ether1
 2 D 172.16.1.2/32      172.16.1.1      pptp-out1
[admin@ R4] > ip ad pr
Flags: X - disabled, I - invalid, D - dynamic
 #   ADDRESS            NETWORK         INTERFACE
 0   10.0.4.1/24        10.0.4.0        ether7
 1   192.168.102.1/24   192.168.102.0   ether1
 2 D 172.16.1.3/32      172.16.1.1      pptp-out1
```

Воспользуемся протоколом OSPF

```
[admin@ R0] > routing ospf network add network=172.16.1.2 area=backbone
[admin@ R0] > routing ospf network add network=172.16.1.3 area=backbone
[admin@ R0] > routing ospf network add network=192.168.100.0/24 area=backbone
[admin@ R1] > routing ospf network add network=172.16.1.1 area=backbone
[admin@ R1] > routing ospf network add network=192.168.101.0/24 area=backbone
[admin@ R4] > routing ospf network add network=172.16.1.1 area=backbone
[admin@ R4] > routing ospf network add network=192.168.102.0/24 area=backbone
```

Обратите внимание, что в настройках сетей для OSPF (как и в RIP) экспортируется сеть, а не адрес. Посмотрим созданные OSPF маршруты

```
[admin@ R4] > ip ro pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS        PREF-SRC        GATEWAY           DISTANCE
 0 A S  10.0.0.0/16                       10.0.4.2          1
 1 ADC  10.0.4.0/24       10.0.4.1        ether7            0
 2 ADC  172.16.1.1/32     172.16.1.3      pptp-out1         0
 3 ADo  172.16.1.2/32                     172.16.1.1        110
 4 ADo  172.16.1.3/32                     172.16.1.1        110
 5 ADo  192.168.100.0/24                  172.16.1.1        110
 6 ADo  192.168.101.0/24                  172.16.1.1        110
 7 ADC  192.168.102.0/24  192.168.102.1   ether1            0
```

Есть маршрут на сеть 192.168.101.0/24 компьютера R3

```
[admin@ R1] > ip ro pr
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
```

```
B - blackhole, U - unreachable, P - prohibit
 #     DST-ADDRESS       PREF-SRC        GATEWAY          DISTANCE
 0 A S  10.0.0.0/16                      10.0.1.2         1
 1 ADC  10.0.1.0/24       10.0.1.1       ether7           0
 2 ADC  172.16.1.1/32     172.16.1.2     pptp-out1        0
 3 ADo  172.16.1.2/32                    172.16.1.1       110
 4 ADo  172.16.1.3/32                    172.16.1.1       110
 5 ADo  192.168.100.0/24                 172.16.1.1       110
 6 ADC  192.168.101.0/24  192.168.101.1  ether1           0
 7 ADo  192.168.102.0/24                 172.16.1.1       110
```

Есть маршрут на сеть 192.168.102.0/24 компьютера R6. Теперь R3 увидит по IP R6 и наоборот.

Повторите настройки OSPF для L2TP, SSTP и OPENVPN.

Задание. Сделайте копию l3vpnrip2 топологии l3vpnrip и настройте маршрутизацию путём назначения каждому клиенту определённого адреса для PPTP, L2TP, SSTP и OPENVPN (подобно топологии l3vpnospf).

## VPN 3 уровня через NAT

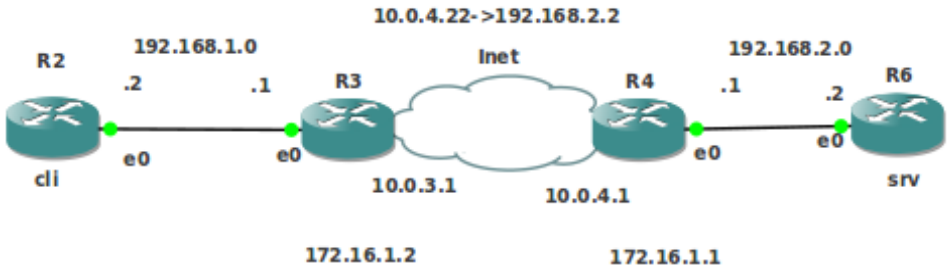



**Рис. 42. Топологии l3vpnnat**

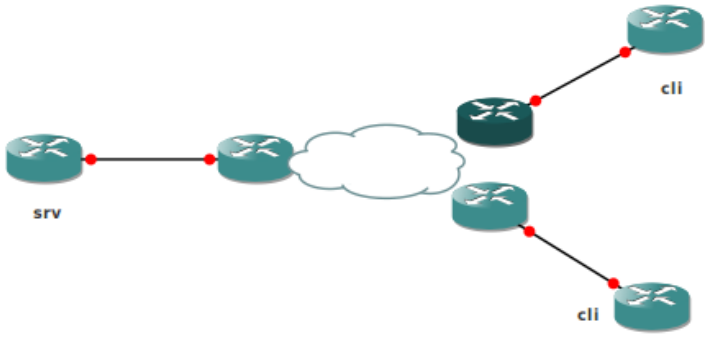



**Рис. 43. Топологии l3vpnnat3pptp l3vpnnat3l2tp l3vpnnat3sstp l3vpnnat3ovpn**

Соберём топологию, изображённую Рис. 43. Проверьте соседей. Здесь R3 и R4 – маршрутизаторы Интернет провайдера. Согласно рисунку назначьте для R2 и R3 адреса из сети 192.168.1.0/24. Для R2 назначьте щлюз 192.168.1.1. Назначьте для R4 и R6 адреса из сети 192.168.2.0/24. Для R2 назначьте щлюз 192.168.2.1. Настройте NAT для исходящих и приходящих адресов

[admin@R3] > ip **firewall nat add chain=srcnat action=masquerade out-interface=ether7**
[admin@R4]>**ip firewall nat add chain=dstnat action=dst-nat to-addresses=192.168.2.2 dst-address=10.0.4.22**

Набрав на R2

[admin@R2] > **sys telnet 10.0.4.22**
должны попасть в 192.168.2.2 (R6). Выход CtrlD

Настроим PPTP, полагая, что профиль default имеет номер 0

[admin@R6] > **ppp profile set 0 local-address=172.16.1.1 remote-address=172.16.1.2**
[admin@R6] > **ppp secret add name="q" service=pptp password="q" profile=default**

Запускаем PPTP сервер R6

[admin@R6] >interface **pptp-server server set enabled=yes**

Настроим PPTP-клиент R2. Соединяемся к PPTP-серверу через NAT T.e. через адрес 10.0.4.22, а не через 192.168.2.2.

[admin@R2] > **interface pptp-client add connect-to=10.0.4.22 user="q" password="q"** disabled=no

Проверим доступность R6 из R2

[admin@R2] ping 172.16.1.1

**Задание.**

Для рис 43 настройте VPN 3 уровня через NAT для протоколов PPTP, L2TP, SSTP и OPENVPN. Это 4 отдельные топологии. Протокол маршрутизации выбрать самостоятельно. Проверку осуществляйте расширенным пингом от клиента cli к клиенту cli
**ping a1 src=a2,**
где a1 и a1 -динамически назначенные адреса

## Настройка PPPoE

Соберите топологию, приведенную на рисунке Рис. 44. Назначьте адреса для R7, R6 и R4 согласно рисунку. Назначьте имена, согласно рисунку, например **sy id se name=R2.**

Существует локальная сеть, образованная компьютерами R7, R6 и маршрутизатором R4 с адресами 192.168.1.1/24, 192.168.1.2/24 192.168.1.3/24 соответственно. Мы хотим присоединить к этой сети новые компьютеры R2 и R3 по протоколу PPPoE. R4 будет . R4 будет PPPoE-сервером.

Добавим пользователей PPPoE

[admin@R4] > **/ppp secret add name="1" service=pppoe password="1" profile=defaul**
[admin@R4] > **/ppp secret add name="2" service=pppoe password="2" profile=default**

Настроим сервер PPPoE на интерфейсе ether1

[admin@R4] > **interface pppoe-server server add interface=ether1default-profile=default**

Настроим клиентов PPPoE, указывая им эсернет-интерфейсы

[admin@R2]>**interface pppoe-client add interface=ether2 user="2" password="2" profile=default**
[admin@R3] >**interface pppoe-client add interface=ether1 user="1" password="1" profile=default**

Установится два PPPoE соединения. На R4 видим два, а на R2 и R3 по одному новых PPPoE-интерфейса, которые можно увидеть командой **interface print detail**

Настроим в сервере PPPoE пул адресов. Адреса пула должны лежать в уже существующей сети 192.168.1.0/24

[admin@R4] >/**ip pool add name="pool1" ranges=192.168.1.100-192.168.1.200**

Изменим профиль по умолчанию, указав в нём созданный пул адресов, а в качестве локального адреса PPPoE-сервера укажем уже существующий адрес 192.168.1.3

[admin@R4] > **/ppp profile pr det**

0 * name="default" local-address=192.168.1.3 remote-address=pool1

Перезапусти PPPoE-сервер

[admin@R4] > **interface pppoe-server server> set 0 disabled=yes**

[admin@R4] > **interface pppoe-server server> set 0 disabled=no**

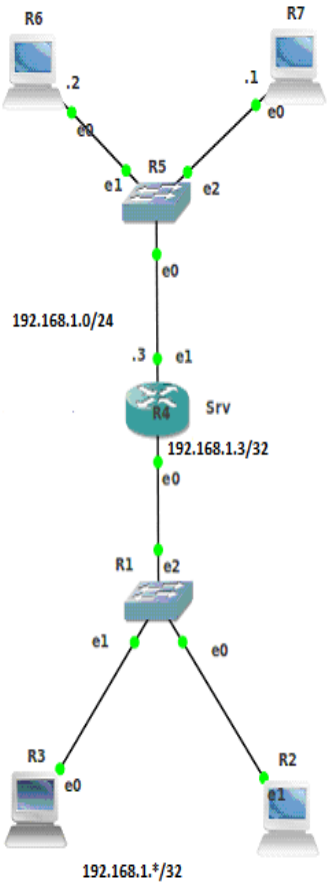



**Рис. 44. Топология pppoe**

На R4 видим новые адреса

[admin@R4] > **ip ad print detail**

Flags: X - disabled, I - invalid, D - dynamic

0 address=10.0.4.1/24 network=10.0.4.0 interface=ether7

1 address=192.168.1.3/24 network=192.168.1.0 interface=ether2

2 D address=192.168.1.3/32 network=192.168.1.200 interface=<pppoe-2>

3 D address=192.168.1.3/32 network=192.168.1.199 interface=<pppoe-1>

Адрес 192.168.1.3 повторяется три раза.

Аналогично видим адреса и у клиентов, назначенные им PPPoE-сервером из пула

```
[admin@R2] > ip address  print detail
 0   address=10.0.2.1/24 network=10.0.2.0 interface=ether7
 1 D address=192.168.1.200/32 network=192.168.1.3 interface=pppoe-out1
[admin@R3] > ip address  print detail
 0   address=10.0.3.1/24 network=10.0.3.0 interface=ether7
 1 D address=192.168.1.199/32 network=192.168.1.3 interface=pppoe-out1
```

Чтобы PPPoE клиенты R2 и R3 могли связаться по IP с компьютерами R6 и R7 они должны быть в состоянии послать им ARP-запрос и получить от них ARP-ответ. Но между ними лежит устройство R4. Нужно сделать так, чтобы R4 транслировало ARP-пакеты. Поэтому на нём организуем прокси-ARP для интерфейса ether2(e1), так как e1 смотрит в сторону R6 и R7

```
[admin@R4] > interface ethernet print
 0 R  ether1                    1500  00:AA:00:E5:69:00 enabled
 1 R  ether2                    1500  00:AA:00:E5:69:01  enabled
[admin@R4] > interface ethernet set 1 arp=proxy-arp
```

Теперь каждое устройство пингует каждое. Проверьте.

Для сдачи этой работы необходимо предоставить следующие работающие топологии для PPTP, L2TP, SSTP и OPENVPN: l2vpn1, l2vpn2, l3vpnospf**,** l3vpnrip2 – 16 шт.**l3vpnnat3pptp l3vpnnat3l2tp l3vpnnat3sstp l3vpnnat3ovpn**

# 7. IPsec (Internet Protocol Security)

**Немного теории**

- **Заголовок аутентификации** (AH)
- **Инкапсуляция зашифрованных данных (ESP)**
- **IKE**
- **Политики IPsec**

**Фазы IKE**
**Конфигурация IPsec в Mikrotik RouterOS**
**Шифрация соединения точка-точка.**
**Организация VPN типа сайт-сайт с помощью IPsec**
**Немного теории**

IPsec представляет собой набор протоколов для безопасного обмена пакетами по сети IP. Семейство IPSec протокол можно разделить на следующие группы:

*заголовок аутентификации (Authentication Header (AH));
* инкапсуляция зашифрованных данных (Encapsulating Security Payload (ESP)) ;
* протоколы обмена Интернет-ключей (Internet Key Exchange (IKE)).

**Заголовок аутентификации** (AH)

AH это протокол, который обеспечивает проверку подлинности содержания IP-датаграммы путем добавления в датаграмму контрольного хеш-значения, которое рассчитывается на основе этой датаграммы. Какие части датаграммы используются для расчета, а также размещение полей датаграммы, зависит от того, какой режим используется: туннельный или транспортный.

AH не шифрует датаграмму. Для шифрования используется другой протокол ESP. Основным и единственным назначением AH является обеспечение защиты от атак, связанных с несанкционированным изменением содержимого пакета, и в том числе от подмены исходного адреса сетевого уровня.

В **транспортном** режиме AH-заголовок вставляется после заголовка IP. AH-заголовок содержит контрольное хеш-значение. Для его расчета используется IP-данные и заголовки. IP-поля, которые могут меняться во время транзита, какие как TTL , приводятся к нулевым значениям до аутентификации.

В **туннельном** режиме исходный пакет IP инкапсулируется в новый пакет IP. Весь исходный пакет IP проходит проверки подлинности. AH-заголовок вставляется после нового заголовка IP

**Инкапсуляция зашифрованных данных (ESP)**

ESP использует общий ключ шифрования для обеспечения конфиденциальности данных. ESP также поддерживает свою собственную схему аутентификации, подобную той, которая используется в AH, или может быть использован в сочетании с AH.

В **транспортном** режиме IP-заголовки не подвергаются аутентификации и шифрованию, а IP-даные подвергаются.

В **туннельном** режиме исходный пакет IP инкапсулируется в новый пакет IP, обеспечивая шифрование IP-данных и IP-заголовка.

Аппаратное шифрование позволяет значительно ускорить процесс шифрования с помощью встроенной внутри процессора устройства шифрования. Для сравнения маршрутизатор RB1100AH с использованием алгоритма AES-128 может обработать 550 Мбит/с зашифрованного трафика. Без аппаратной поддержки он может обработать только 150 Мбит/с зашифрованного трафика.

**IKE**

Гарантии целостности и конфиденциальности данных в спецификации IPsec обеспечиваются за счет использования механизмов аутентификации и шифрования соответственно. Последние, в свою очередь, основаны на предварительном согласовании сторонами информационного обмена применяемых криптографических алгоритмов, алгоритмов управления ключевой информацией и их параметров. После согласования возникает ассоциация безопасности SA (Security Association). Два компьютера на каждой стороне каждого SA хранят режим, протокол, алгоритмы и ключи, которые будут использованы для шифрования или подписи передаваемых данных. Каждый SA используется только в одном направлении. Для двунаправленной связи требуется два SA. Каждый SA реализует один режим и протокол; таким образом, если для одного пакета необходимо использовать два протокола (например AH и ESP), то требуется два SA. SA динамически создаётся с помощью IKE до начала обмена данными между двумя пирами. SA содержится в базе SAD (Security Association Database).

Каждое SA единственным образом определяется тремя параметрами

* Security Parameter Index (SPI): 32-битовое числ, назначамое SA и имеющее только локальное значение. SPI включается в заголовки AH и ESP, чтобы получатель смог выбрать корректный SA для обработки получаемого пакета.

* IP-адрес получателя - противоположной стороны с которой установлено SA.

* Security Protocol Identifier: указывает какой протокол IPSec используется на SA (AH или ESP).

Для создания общих ключей IKE использует протокол Oakely. Эти ключи затем используются для шифрования и проверки. Процесс создания ключей базируется на алгоритме обмена ключей Diffie-Hellman. Алгоритм предлагает процедуру обмена данными между двумя сторонам через небезопасное соединение, которая приводит к генерации общего секретного ключа. Сам ключи не передаётся.

**Политики IPsec**

Политика определяют, что делать с группой пакетов, удовлетворяющих некоему условию. Политики хранятся в SPD (База данных политики безопасности) и базе SAD. Политика может указать для пакета данных одно из трёх действий: отбросить пакет, не обрабатывать пакет с помощью IPSec и обработать пакет с помощью IPSec. В последнем случае политика также указывает, какой SA необходимо использовать (если, конечно, подходящий SA уже был создан) или указывает, с какими параметрами должен быть создан новый SA. IPsec согласовывает с другой стороной по протоколу IKE создание нового SA и его параметры

Устройство проверяет базу данных SPD для определения, что делать с конкретной датаграммой. SPD может ссылаться на конкретный SA в SAD. Если так, то устройство будет искать этот SA и использовать его для обработки датаграммы

SPD является очень гибким механизмом управления, который допускает очень хорошее управление обработкой каждого пакета. Пакеты классифицируются по большому числу полей, и SPD может проверять некоторые или все поля для того, чтобы определить соответствующее действие. Это может привести к тому, что весь трафик между двумя машинами будет передаваться при помощи одного SA, либо отдельные SA будут использоваться для каждого приложения, или даже для каждого TCP соединения.

**Фазы IKE**

Основное время служба IKE ничего не делает. Если имеется некий трафик, соответствующий правилу политики, который нуждается в шифровании или аутентификации, но политика не имеет никакого SA, то политика извещает об этом службу IKE и она инициирует связь к удалённому хосту. IKE выполняет две фазы:

* Фаза 1 - аутентификация сторон IPSec и создание защищенного канала между сторонами, позволяющего начать обмен IKE для фазы 2. Также рассчитываются общие ключи для SA
* Фаза 2 - согласование параметров SA с целью создания туннеля IPSec.

Все SA, установленные службой IKE, будут иметь значение времени жизни. Имеются два значения времени жизни - твёрдое и мягкое. Когда SA достигает своего мягкого порога, служба IKE получает извещение и запускает ещё одну фазу 2 обмена для освежения SA . Если SA достигли твёрдого времени жизни, то они отбрасываются.

Генерация ключей требует значительных вычислительных затрат. Это обычно имеет место один раз в течение фазы 1.

IKE может дополнительно обеспечить идеальную прямую безопасность (Perfect Forward Secrecy (PFS)). Наличие PFS означает невозможность расшифровки всего трафика при компрометации любого ключа в системе

**Конфигурация IPsec в Mikrotik RouterOS**

**Конфигурация пиров осуществляется через п**одменю: **`/ip ipsec peer согласно табл.1.`** Конфигурации используются для установки соединения между службами IKE (1-я фаза). Это соединение далее используется в процессе согласования ключей и алгоритмов для SA.

**Конфигурация пиров** **Таблица 1**

| Свойство | Описание |
|---|---|
| ***address*** (*IP[/Netmask]:port*; Default: **0.0.0.0/32:500**) | Адресный префикс. Если адрес удалённого пира подпадает под этот префикс, то эта конфигурация используется в аутентификации и установке фазы 1. |
| ***auth-method*** (*pre-shared-key* \| *rsa-signature*; Default: **pre-shared-key**) | Метод аутентификации:<br>*pre-shared-key* - аутентификации по общей для пиров парольной строке<br>*rsa-signature* – аутентификация с помощью пары RSA -сертификатов |
| ***certificate*** (*string*; Default: ) | Имя локального сертификата. Он должен иметь |

| | |
|---|---|
| | закрытый ключ. Применимо при аутентификации RSA-signature |
| ***dh-group*** (*ec2n155* \| *ec2n185* \| *modp1024* \| *modp1536* \| *modp768*; Default: **modp1024**) | Группа Diffie-Hellman (сила шифра) |
| ***dpd-interval*** (*disable-dpd* \| *time*; Default: **disable-dpd**) | Интервал обнаружения отсутствия пира. При *disable-dpd* обнаружения не производится |
| ***dpd-maximum-failures*** (*integer: 1..100*; Default: **5**) | Максимальное количество сбоев до определения пира как отсутствующего |
| ***enc-algorithm*** (*3des* \| *aes-128* \| *aes-192* \| *aes-256* \| *des* \| *blowfish* \| *camellia-128* \| *camellia-192* \| *camellia-256*; Default: **3des**) | Алгоритм шифрования |
| ***exchange-mode*** (*aggressive* \| *base* \| *main*; Default: **main**) | Режим обмена в фазе 1. Не меняйте, если не понимаете |
| ***generate-policy*** (*yes* \| *no*; Default: **no**) | Позволяет пиру установить SA для несуществующей политики. Политика создаётся динамически во время жизни SA. Этот способ полезен, когда адрес удалённого пира неизвестен |
| ***hash-algorithm*** (*md5* \| *sha1*; Default: **md5**) | Алгоритм хеширования. SHA сильнее, но медленнее |
| ***lifebytes*** (*Integer: 0..4294967295*; Default: **0**) | Время жизни первой фазы. Определяет сколько байт может быть передано до того как SA будет отброшено. Если указан 0, то SA не отбрасывается. |
| ***lifetime*** (*time*; Default: **1d**) | Время жизни фазы 1: определяет время действительности SA |
| ***nat-traversal*** (*yes* \| *no*; Default: **no**) | Использование механизма Linux NAT- *traversal* для разрешения несовместимости IPsec и NAT |
| ***proposal-check*** (*claim* \| *exact* \| *obey* \| *strict*; Default: **obey**) | Логика проверки времени жизни фазы 2:<br>*claim* – взять самое короткое из предложенных времён жизни и известить инициатора об этом<br>*exact* – требовать, чтобы время жизни было такое же<br>*obey* – принять всё, что пошлёт инициатор<br>*strict* – если предложенное время больше, чем время по умолчанию, то отбросить. Иначе принять предложенное |
| ***remote-certificate*** (*string*; Default: ) | Имя сертификата для аутентификации удалённой стороны. Закрытый ключ не требуется. Применимо при аутентификации RSA-signature |
| ***secret*** (*string*; Default: **""**) | Парольная строка. Применимо при аутентификации pre-shared key. Если начинается с 0x, то рассматривается как 16-е число |
| ***send-initial-contact*** (*yes* \| *no*; Default: **yes**) | Определяет отсылать ли начальную IKE-информацию или ждать удалённую сторону |

При смене этой конфигурации на лету информация о фазах IPSec уничтожается, однако пакеты продолжают шифроваться и дешифроваться согласно установленным SA.

**Политика настраивается в п**одменю **`/ip ipsec policy согласно табл.2.`** Таблица политик предназначена для определения того, какие установки безопасности будут применены к пакету

**Конфигурация политик** **Таблица 2**

| **Свойство** | **Описание** |
|---|---|
| ***action*** (*discard* \| *encrypt* \| *none*; Default: **encrypt**) | Оределяет, что делать с пакетом, который удовлетворяет политике.<br>*none* – пропустить пакет без изменения<br>*discard* - отбросить пакет<br>*encrypt* – применить преобразование, определённое в этой политике и SA |
| ***dst-address*** (*IP/Mask:Port*; Default: **0.0.0.0/32:any**) | Адресный префикс и порт назначения. |
| ***ipsec-protocols*** (*ah*\|*esp*; Default: **esp**) | Определяет какая комбинация протоколов AH и ESP будет применена к трафику, который удовлетворяет политике. |
| ***level*** (*require* \| *unique* \| *use*; Default: **require**) | Определяет, что делать, если некоторые SA для этой политике не могут быть найдены:<br>*use* – пропустить это преобразование, не отбрасывать пакет и не получать SA от IKE-службы<br>*require* - отбросить пакет и получить SA<br>*unique* - отбросить пакет и получить уникальный SA , который только и используется для этой конкретной политики |
| ***priority*** (*Integer: -2147483646..2147483647*; Default: **0**) | Классификатор порядка политик (целое со знаком). Большее число значит больший приоритет |
| ***proposal*** (*string*; Default: **default**) | Имя предлагаемой информации, которое будет передано IKE-службой для установки SA для этой политики |
| ***protocol*** (*all* \| *egp* \| *ggp* \| *icmp* \| *igmp* \| ...; Default: **all**) | Для каких протоколов использовать |
| ***sa-dst-address*** (*IP*; Default: **0.0.0.0**) | Удалённый IP-адрес SA (удалённый пир). |
| ***sa-src-address*** (*IP*; Default: **0.0.0.0**) | Локальный IP-адрес SA (локальный пир). |
| ***src-address*** (*IP/Mask:Port*; Default: **0.0.0.0/32:any**) | Адресный префикс и порт источника. |
| ***tunnel*** (*yes* \| *no*; Default: **no**) | Определяет, использовать ли туннельный режим |

В транспортном режиме следует указать ***dst-address= sa-dst-address и src-address= sa-dst-address.*** Для шифрования трафика между сетями следует использовать туннель. Пакет из сети ***src-address*** в сеть ***dst-address*** будет помещён в новый пакет с адресом источника ***sa-src-address*** и адресом приёмника ***sa-dst-address.***

**Установка** предлагаемой информации производится в подменю /ip ipsec proposal `согласно табл.3.` Предлагаемая информация отсылается службой IKE для установки SA для политики. Имя предлагаемой информации устанавливается в политике.

Установка предлагаемой информации Таблица 3

| **Свойство** | **Описание** |
|---|---|

***auth-algorithms*** (*md5*|*sha1*|*null*; Default: **sha1**) — Разрешённые алгоритмы проверки подлинности

***enc-algorithms*** (*null*|*des*|*3des*|*aes-128*|*aes-192*|*aes-256*|*blowfish*|*camellia-128* | *camellia-192* | *camellia-256* | *twofish*; Default: **3des**) — Разрешённые алгоритмы шифрования

***lifetime*** (*time*; Default: **30m**) — Время жизни SA

***pfs-group*** (*ec2n155* | *ec2n185* | *modp1024* | *modp1536* | *modp768* | ...; Default: **modp1024**) — Группа Diffie-Helman для Perfect Forward Secrecy.

Подменю **`/ip ipsec installed-sa`** `предоставляет` информацию об установленных SAи их ключах. С помощью команды /ip ipsec installed-sa flush можно вручную сбросить SA, которые некорректно установлены IKE. Подменю /ip ipsec remote-peers содержим информацию только для чтения об активных удалённых пирах

## Шифрация соединения точка-точка.

Соберём топологию ipsec1, изображённую на рис.1. Пусть первый пакет исходит от R1. В winbox для настройки пиров R0 и R1 выбираем IP->IPsec->Peers->+. Вводим секретную фразу и адрес противоположной стороны. В R0 настраиваем генерацию политики по запросу на шифрование от R1 (Рис. 2). В R1 настраиваем инициацию шифрования (Рис. 3).

Для определения политики в R1 выбираем в winbox IP->IPsec->Policies->+ и вводим адреса (рис. 4). Не нажимая Apply, переходим к вкладке Action и вводим те же адреса (рис. 5). Нажимаем Ok и инициируем шифрование, запуская на R1 в сторону R0 пинги (Tools->Ping). Видим некоторую задержку (**Рис.6)**.

Проверяем. На R0 появилась политика (IP>IPsec->Policies). И R0 и R1 увидели удалённого собеседника (IP->IPsec->Remote Peers). И на R0 и на R1 инсталлировались SA (IP->IPsec->Installed SAs). Трафик шифруется.

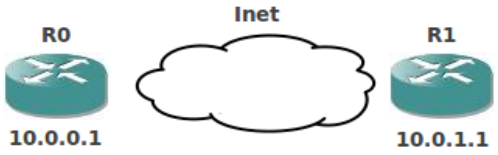



**Рис.1 Топология ipsec1**

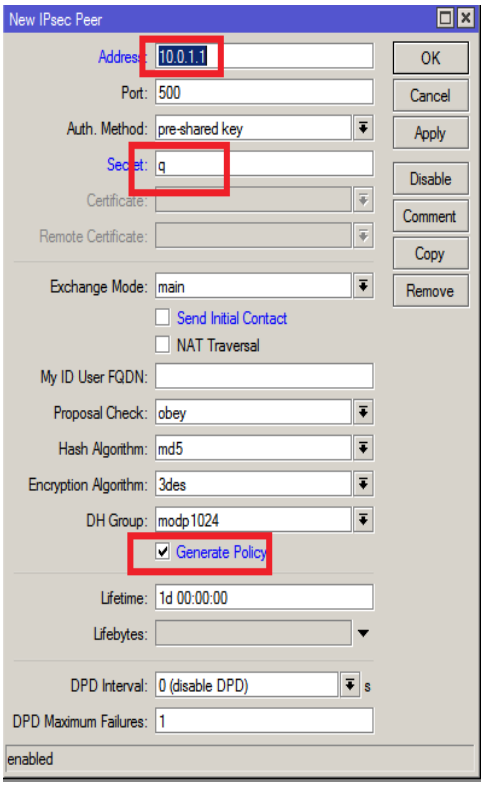



**Рис.2 Настройка пира R0**

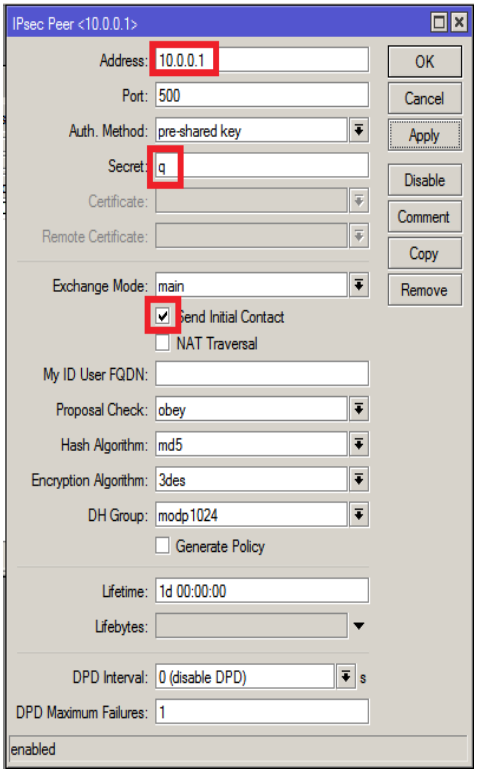



**Рис.3 Настройка пира R1**

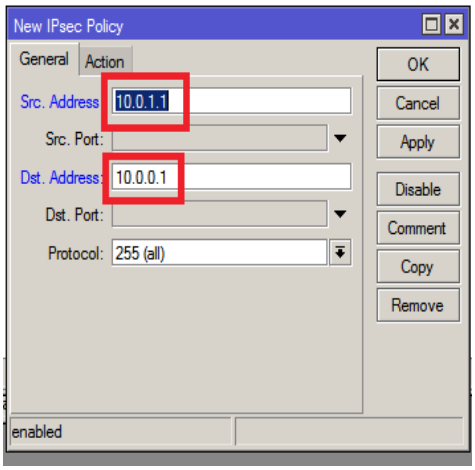



**Рис.4. Настройка политики в R1**

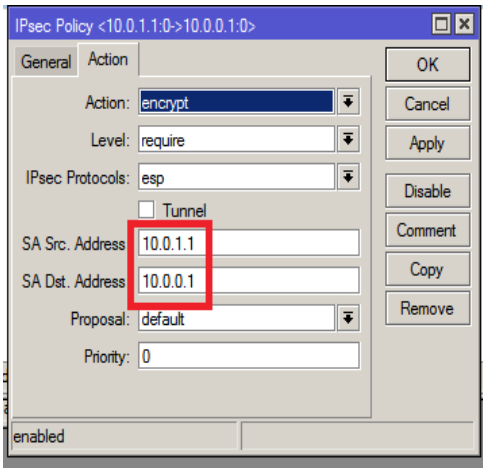



**Рис.5. Дальнейшая настройка политики в R1**

**Задание.** В Windows протокол L2TP представлен не в чистом виде, а в сочетании с IPsec. Сделайте копию IPsecL2TP оследней топологии, организуйте L2TP соединение от R1 к R0 и организуйте поверх него IPsec.

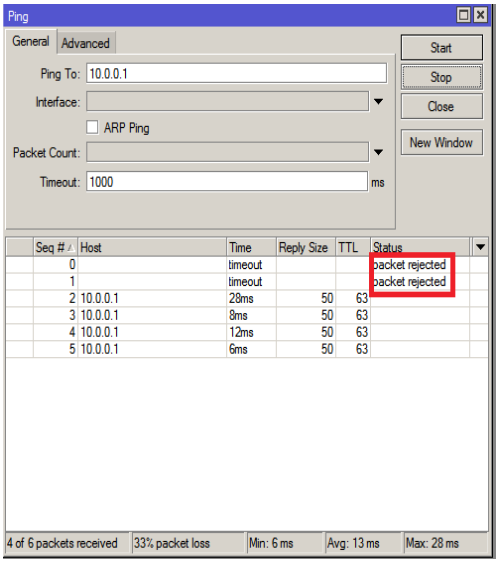



**Рис.6. Шифрация началась**

## Организация VPN типа сайт-сайт с помощью IPsec

Соберём топологию ipsec2, изображённую на рис.7. Внешние адреса 10.0.0.1 и 10.0.1.1 измените согласно своему номеру. У кого номер 1 измените внутренние адреса сетей 10.1.101.0/24 и 10.1.101.0/24.

Дайте сетевым устройствам имена внутри RouterOS. Превратите sw6 и sw7 в свичи, поместив используемые эсернет-интерфейсы в мост. Проверьте соседей. Назначьте адреса согласно рисунку. На R2 и R3 пропишите маршрут по умолчанию на 10.1.202.1. На R4 и R5 пропишете маршрут по умолчанию на 10.1.101.1

На R0 пропишем маршрут на R1 через модель Интернета

**[admin@R0] > Route add dst=10.0. 1.0/24 gate=10.0.0.2**

На R1 пропишем маршрут на R0 через модель Интернета

**[admin@R1] > Route add dst=10.0. 0.0/24 gate=10.0.1.2**

На R0 пропишем маршрут на сеть 10.1.101.0/24 через 10.0.1.1

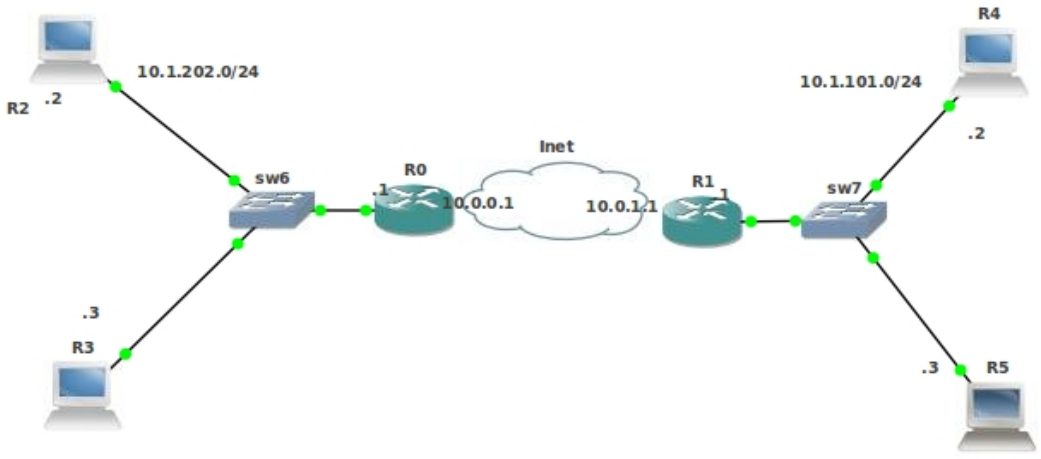



**Рис.7. Топология ipsec2**

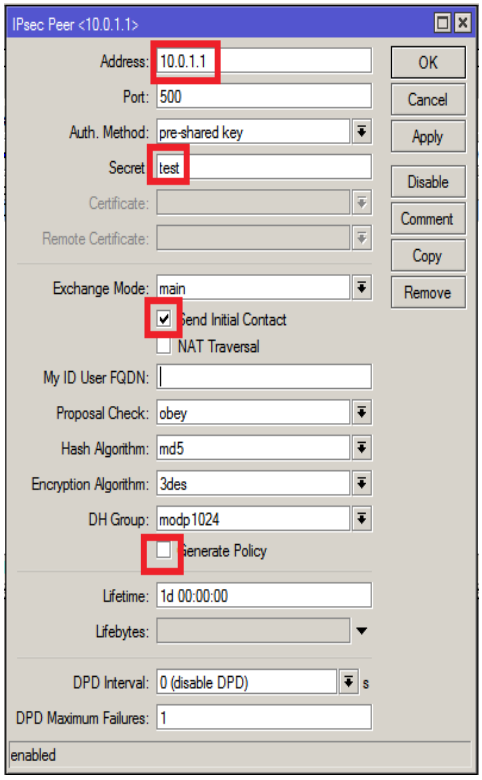



**Рис.8. Настройка пира R0**

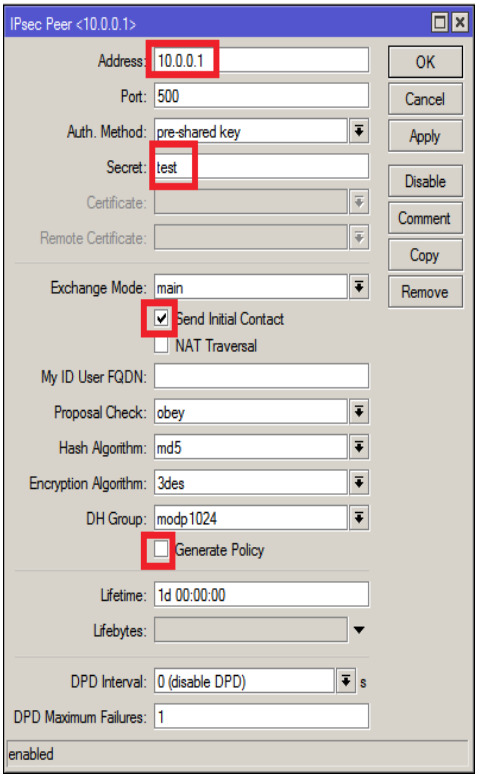



**Рис.9. Настройка пира R1**

**[admin@R0] > Route add dst=10.1.101.0/24 gate=10.0.1.1**

На R1 пропишем маршрут на сеть 10.1.202.0/24 через 10.0.0.1

**[admin@R1] > Route add dst=10.1.202.0/24 gate=10.0.0.1**

Компьютеры из сети 10.1.101.0/24 не видят компьютеры из сети 10.1.202.0/24 и наоборот. Это естественно. Так как ubuntu не знает, что делать с пакетами из этих сетей. Выполните в Ubuntu команду **Netstat –r** и вы не увидите маршрутов на эти сети.

Мы не знаем, кто инициирует шифрование. Поэтому произведём симметричную конфигурацию пиров, указав адрес собеседника и секрет (рис. 8 и 9).

Определим на R0 политику, задающую шифрование от сети 10.1.202.0/24 к сети 10.1.101.0/24 (рис.10)**.** Для SA в качестве адреса источника возьмём внешний адрес 10.0.0.1 R0, а в качестве адреса приёмника возьмём внешний адрес 10.0.1.1 R1. Обмен осуществляется в туннельном режиме (рис.11)**.**

Определим на R1 политику, задающую шифрование от сети 10.1.101.0/24 к сети 10.1.202.0/24 (рис.12). Для SA в качестве адреса источника возьмём внешний адрес 10.0.1.1 R1, а в качестве адреса приёмника возьмём внешний адрес 10.0.0.1 R0. Обмен осуществляется в туннельном режиме (рис.13)**.**

Проверяем. И R0 и R1 увидели удалённого собеседника (IP->IPsec->Remote Peers). И на R0 и на R1 инсталировались SA (IP->IPsec->Installed SAs).

Убедимся, что мы с помощью IPsec создали VPN типа сайт-сайт между сетями 10.1.101.0/24 и 10.1.101.0/24. Для этого протрассируем пакеты из сети 10.1.101.0/24 в сеть 10.1.101.0/24. Наберём в R4 команду

[admin@ R4**] > tools tracert 10.1.202.2**

где 10.1.202.2 – это адрес R2.

Пакеты протокола ICMP между сайтами пошли (рис.14). VPN создано.

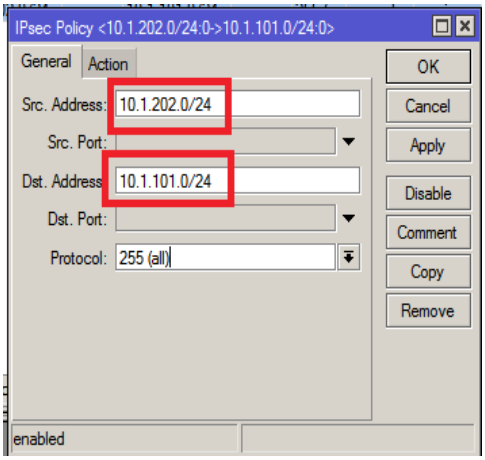



**Рис.10. Политика на R0**

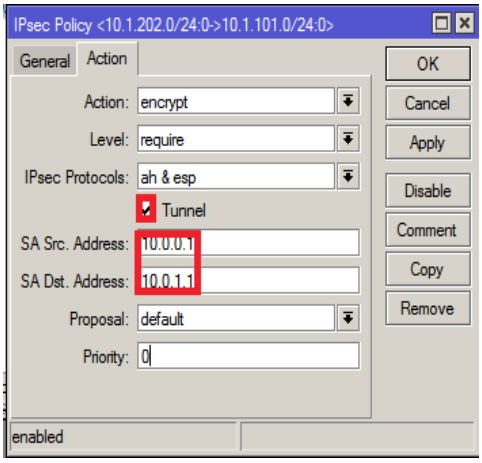



**Рис.11. Политика на R0. SA**

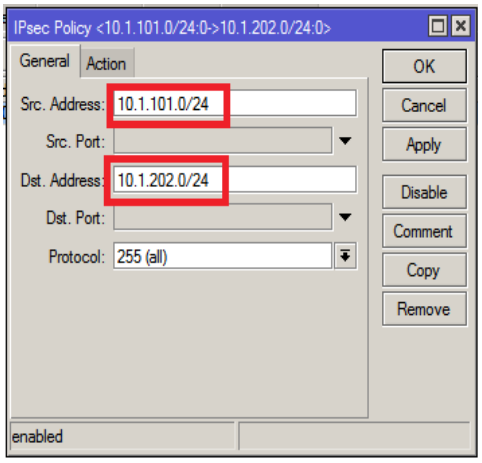



**Рис.12. Политика на R1**

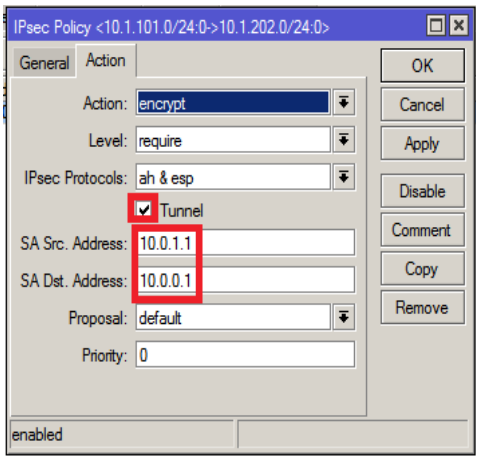



**Рис.13. Политика на R1. SA**

```
[admin@MikroTik] > tool trace 10.1.202.2
 # ADDRESS                                  RT1   RT2   RT3   STATUS
 1 10.1.101.1                               2ms   2ms   1ms
 2 0.0.0.0                                  0ms   0ms   0ms
 3 10.1.202.2                               6ms   4ms   4ms
```

**Рис.14. Прохождение пакетов ICMP между сайтами в VPN**

Компьютеры из сети 10.1.101.0/24 не видят компьютеры из сети 10.1.202.0/24 и наоборот. Это естественно. Так как Ubuntu не знает, что делать с пакетами из сетей 10.1.101.0/24 и 10.1.202.0/24. Что случилось? Как образовалась VPN? Просто теперь пакеты из этих сетей зашифрованы и посещены в пакеты с адресами 10.0.0.1/24 и 10.0.1.1/24, с которыми Ubuntu знает, что делать. Если вы обладаете правами суперпользователя, то это можно увидеть с помощью программы анализатора сетевых пакетов Wireshark (рис.15).

```
⊞ Frame 11: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
⊞ Ethernet II, Src: 00:ff:82:a8:65:34 (00:ff:82:a8:65:34), Dst: LogicMod_2f:4a:00 (00:00:ab:2f:4a:00)
⊞ Internet Protocol, Src: 10.0.1.1 (10.0.1.1), Dst: 10.0.0.1 (10.0.0.1)
⊟ Encapsulating Security Payload
    ESP SPI: 0x0aa17809
    ESP Sequence: 81
```

**Рис.15. Wireshark показывает шифрование пакетов**

На рис.14 видим отсутствие внешних адресов 10.0.0.1 и 10.0.0.1 маршрутизаторов R0 и R1, хотя трафик не может через них не походить. Это объясняется тем, что ICMP в R4 и R6 не может знать, что пакеты будут зашифрованы и помещены в пакеты с адресами 10.0.0.1/24 и 10.0.1.1/24.

**Задание**. Организуйте VPN типа сайт-сайт с помощью IPsec для топологии ipsec3, представленной на рис.16. Внешние адреса R0, R1 и R10 измените согласно своему номеру. У кого номер 1 измените адреса сетей.

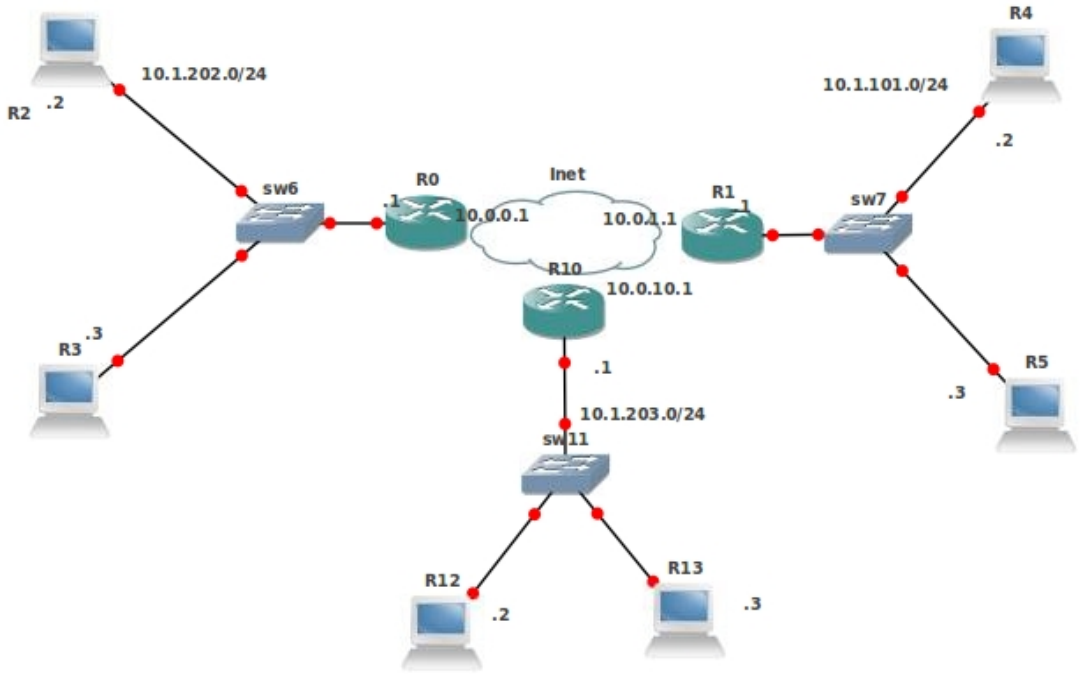



**Рис.16. Топология ipsec3**

Работа считается выполненной если настроены работающие топологии IPsecL2TP и ipsec3

# 8. VPN уровня 2 и 3 на основе MPLS

**MPLS**
**VPLS**
**Требования для MPLS.**
**IP адрес Loopback.**
**IP-связь**
**Настройка LDP**
**Настройка LDP VPLS**
**Мосты**
Настройка VPLS-интерфейсов
**Мост ethernet с VPLS**
**Мост с расщепленным горизонтом**
**Фильтрация меток**
**VPLS, основанные на BGP**
**Конфигурирование сессий BGP для VPLS сигнализации**
**Настройка отражателя маршрутов**
**Настройка VPLS на основе BGP-сигнализации**
**Настройка ethernet-мостов**
**Настройка экземпляров VPLS на основе BGP-сигнализации**
**Cisco BGP VPLS**
**MPLS VPN 3-го уровня**

## MPLS

MPLS это многопротокольная коммутация по меткам (MultiProtocol Label Switching). Она отчасти заменяет IP-маршрутизацию - решение о пересылке пакета (исходящий интерфейс и следующий хоп маршрута) принимается не на основе полей IP-заголовка (обычно адрес назначения) и таблицу маршрутизации, а на метках, которые прикреплены к пакету. Такой подход ускоряет процесс пересылки, потому что поиск следующего хопа становится значительно проще по сравнению с поиском маршрута (поиском самого длинного соответствующего префикса).

Эффективность пересылки это не основное преимущество MPLS. MPLS вносит в сетевые технологии качественно новые возможности, например маршрутизация по адресу источника.

Должно быть принято во внимание, что MPLS-пересылки отключают обработки заголовков сетевого уровня (например, IP), поэтому действия, основанные на сетевом уровне, такие NAT и фильтрации не могут быть применены к пересылаемым по MPLS пакетам. Любые действия, основанные на сетевом уровне, должны быть предприняты на входе или выходе из облака MPLS.

В простейшей форме MPLS можно рассматривать как улучшение маршрутизации - метки распределяются с помощью протокола LDP для маршрутов, которые являются активными и помеченный пакет проходит тот же путь, который он бы прошёл, если он не был бы маркирован.

Путь коммутации по метке обеспечивает передачу данных на выход облака MPLS. Применение MPLS основываются на этой базовой концепции пути коммутации по метке LSP (label switching path).

Когда метка прикрепляется к пакету, он увеличивает его длину на 32 бита (4 байта). Эти 32 бита разделяются так:

- сама метка (20 бит)
- EXP экспериментальное поле(3 бита)

- время жизни (8 бит)
- вершина стека (1 bit)

Экспериментальное поле не определено стандартом, но используется. Оно позволяет различать 8 типов трафика.

MPLS работает с классами эквивалентности при пересылке (forwarding equivalence class -FEC). Примеры классов FEC:

Множеств пакетов с одинаковой сетью назначения.

Множеств пакетов с одинаковой сетью назначения и одинаковой сетью источником.

Множество пакетов с одинаковыми портами назначения

и т.д.

Для пакетов из каждого класса FEC MPLS назначает свою систему меток, что позволяет их одинаково обрабатывать. Где бы пакет не находился в MPLS-сети, метка определяет к какому FEC пакет относится и следовательно может быть обработан в зависимости от FEC.

Именно этим свойством MPLS вносит принципиально новое качество в современные сетевые технологии.

Рассмотрим сеть на рис.1. В качестве FEC возьмём пакеты идущие в сеть 172.16.0.0./24. Рассмотрим систему меток для пакетов, идущих от LSR1 и LSR4 в сторону сети 172.16.0.0./24. Пусть, благодаря обычной маршрутизации на маршрутизаторах есть маршруты на сеть 172.16.0.0/24, приведенные а табл.1

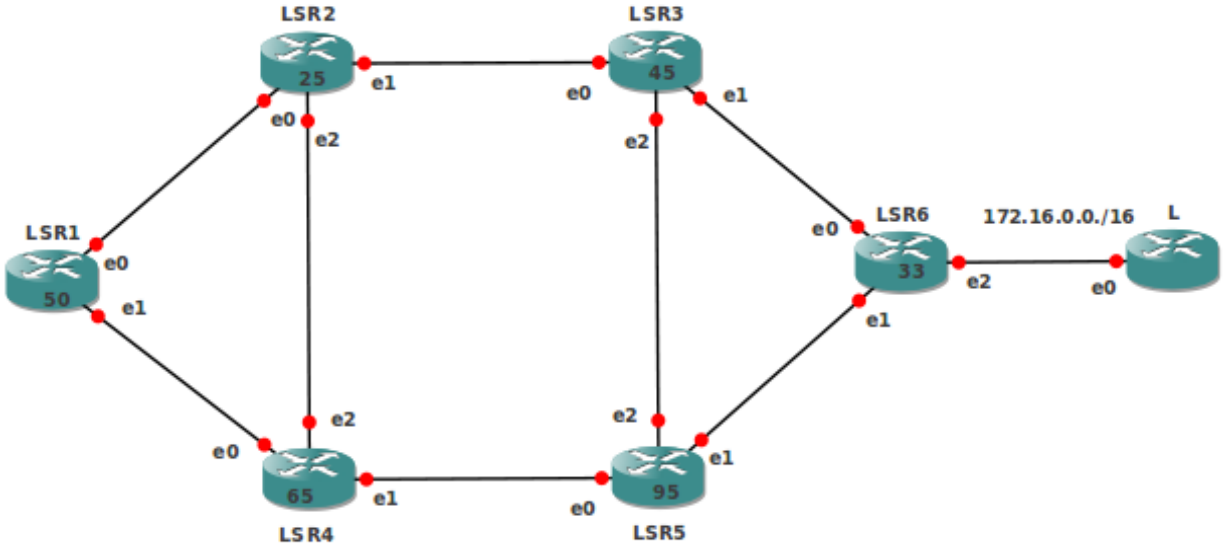



**Рис.1. MPLS-сеть**

Маршруты на сеть 172.16.0.0/24 Таблица 1

| Устройство | IP-дресс следующего перехода | Выходной интерфейс |
|---|---|---|
| LSR1 | LSR2 | e0 |
| LSR2 | LSR3 | e1 |
| LSR3 | LSR6 | e1 |

| | | |
|---|---|---|
| LSR3 | LSR5 | e1 |
| LSR5 | LSR5 | e1 |
| LSR6 | Локально | e2 |

Пакеты из LSR1 идут по пути LSR2 LSR3 LSR6. Пакеты из LSR4 идут по пути LSR5 LSR6. Каждый LSR (label switching router- маршрутизатор, коммутирующий по меткам) строит базу пересылки по меткам LFIB (label forward information base) во взаимодействии с другими LSR. Информация о метках распространяется с помощью протокола распространения меток LDP (label distribution protocol).

В табл.2 приведём информацию из итоговых LFIB. Далее мы покажем, как это работает и затем как это строится. Для удобства входные метки изображены на рис. 1.

Информация из баз LFIB для пересылки пакетов на сеть 172.16.0.0./24 Таблица 2

| Устройство | Входная метка | Выходная метка | MAC-адрес следующего перехода | Выходной интерфейс |
|---|---|---|---|---|
| LSR1 | 50 | 25 | LSR2 | e0 |
| LSR2 | 25 | 45 | LSR3 | e1 |
| LSR3 | 45 | 33 | LSR6 | e1 |
| LSR3 | 65 | 95 | LSR5 | e1 |
| LSR5 | 95 | 33 | LSR6 | e1 |
| LSR6 | 33 | | Локально | e2 |

Пакету, вошедшему в MPLS-сеть, назначается метка. Она помещается в резервное поле заголовка протокола 2 уровня. LSR смотрит у вошедшего пакета метку. Ищет строку в своей LFIB, где поле входная метка равна этой метке. Меняет у IP-пакета заголовок 2-го уровня, взяв для него в качестве адреса приёмника значение поля “MAC-адрес следующего перехода”, а в качестве метки - значение поля “выходная метка” и направляет пакет на интерфейс, определяемый полем “Выходной интерфейс”.

Смотрим на последнюю таблицу. Пакет в сторону сети 172.16.0.0/24 входит в граничный маршрутизатор LSR1. Там он получает метку 50. Согласно LFIB он получает новый заголовок 2 уровня (MAC-адрес LSR2 и метка 25) и направляется на интерфейс e0. Пакет приходит на LSR2. LSR2 читает у пришедшего пакета метку 25 и согласно записи LFIB для этой метки меняет пакету заголовок 2 уровня (MAC-адрес LSR3 и метка 45) и направляет пакет на интерфейс e1. Пакет приходит на LSR3. На LSR3 ищется строка в LFIB со значением поля “Входная метка” равным 45. У пакета меняется заголовок 2 уровня (MAC-адрес LSR6 и метка 33) и он направляется через интерфейс e1. Пакет приходит на граничный маршрутизатор LSR6. LSR6 читает у пришедшего пакета метку 33 и отправляет пакет в локальную сеть интерфейса e0.

Теперь обсудим, как строится LFIB. Последние две колонки LSR строит на основании таблицы маршрутов и ARP-таблиц. Входную метку для сети 172.16.0.0./24 LSR (пусть это LSR6 и метка 33) назначает произвольно из списка не занятых в данной MPLS-сети меток.

Назначенная метка вместе с FEC-кодом сети 172.16.0.0./24 рассылается всем соседям. Те соседи (LSR3 и LSR5), у которых в строке LFIB для сети 172.16.0.0./24 значение поля “MAC-адрес следующего перехода” равно MAC-адресу источника рассылки (LSR6) заносят в поле “Выходная метка” полученную метку (33).

## VPLS

VPLS-это служба виртуальных частных локальных сетей (Virtual Private LAN Service). Служит для обеспечения широковещательной функциональности эсернет-сетей поверх сетей MPLS. Позволяет объединить территориально удалённые локальные сети в один домен широковещания через виртуальный псевдокабель эсернет. Изученный нами протокол EoIP служит той же цели. VPLS по сравнению с EoIP обладает большей гибкостью и функциональностью и поддерживается большим числом производителей сетевого оборудования.

Рассмотрим (рис.2**)** провайдера сетевых услуг, который соединяет 3 удалённых сайта заказчика A (A1 A2 A3) и 2 удалённых сайта заказчика B (B1 B2), используя свою сеть MPLS, состоящую из 5 маршрутизаторов (R1-R5). Заказчики требуют прозрачное ethernet-соединение между сайтами и хотят сами назначать IP-адреса. Из рисунка видим, что заказчики выбрали одинаковые адреса. Это может быть сделано и с помощью моста между физическими ethernet-интерфейсами заказчика и EoIP туннелями, что не есть эффективное решение.

Настройка MPLS ускорит процесс проброски пакетов в такой сети. Использование одного из сервисов MPLS - VPLS может ещё более увеличить эффективность проброски фреймов ethernet, без их инкапсуляции в IP-пакеты и накладных расходов на IP -заголовок.

Приведём пошаговую инструкцию настройки VPLS для обеспечения запрашиваемой услуги. Сделайте копию mpls шаблона template. Соберите топологию и проверьте соседей.

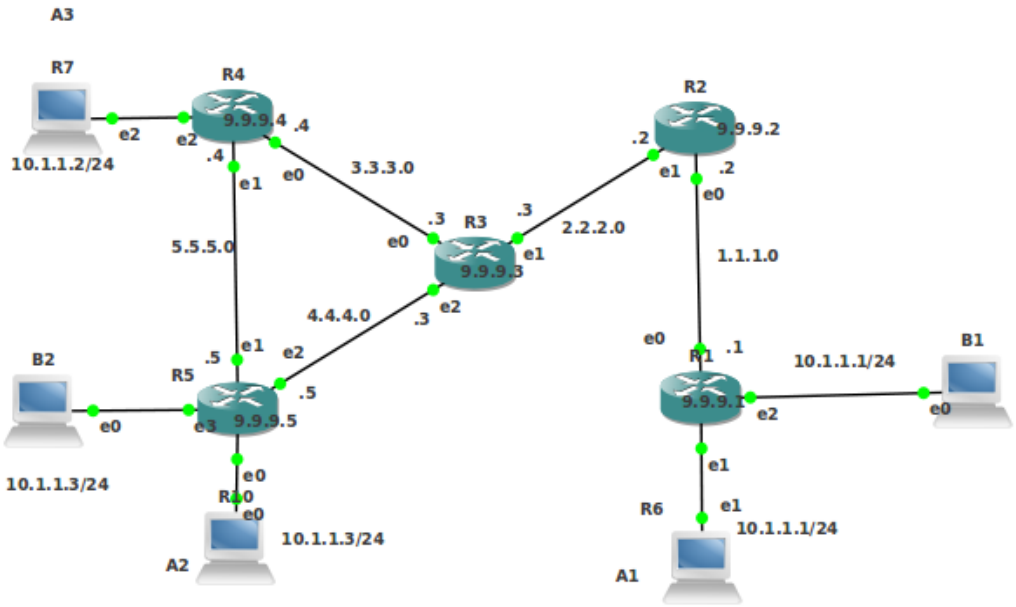



**Рис.2. Топологии** mpls mplsvplsvpn

**Требования для MPLS.**

**IP адрес Loopback.**

Интерфейс loopback не связан с каким-то реальным устройством. Хотя это и не строгое требование, желательно в маршрутизаторах, участвующих в MPLS-сети, назначать IP-адрес на интерфейс Loopback, и использовать этот адрес для установки LDP сессий.

Это служит двум целям:

- так как может быть только одна сессия LDP между двумя маршрутизаторами вне зависимости от того, сколько линков соединяет их, использование IP-адреса на интерфейс Loopback гарантирует, что LDP не будет зависеть от смены состояния и адресов других интерфейсов;
- использование IP-адреса на интерфейс Loopback как транспортного адреса LDP обеспечивает надлежащую работу MPLS, когда несколько меток назначаются на пакет.

В RouterOS IP-адреса на интерфейс Loopback может быть настроен путём создания пустого моста. Например, для R1

[admin@R1]>**interface bridge add name=lobridge**

[admin@R1]>**ip address add address=9.9.9.1/32 interface=lobridge**

Создайте на остальных маршрутизаторах R2-R5 интерфейс Loopback и назначьте на них адреса согласно рисунку. На всех маршрутизаторах R1-R5 назначьте адреса на ethernet-интерфейсы согласно рисунку. Проверьте связь с помощью пингов.

**IP-связь**

Так как LDP распределяет метки по активным маршрутам, важным требованием является правильная настройка IP-маршрутизации. Например, на R5 протокол маршрутизации OSPF может быть настроен с помощью следующих команд:

[admin@R5] >**routing ospf set redistribute-connected=as-type-1**

[admin@R5]>**routing ospf network add area=backbone network=4.4.4.0/24**

[admin@R5]>**routing ospf network add area=backbone network=5.5.5.0/24**

На остальных маршрутизаторах R2-R5 проделайте настройку OSPF подобным образом. На всех маршрутизаторах проверьте таблицы маршрутов и проверьте связь между всеми маршрутизаторами, например

[admin@R5]>**tool traceroute 9.9.9.1**

```
  ADDRESS                         STATUS
1       4.4.4.3 11ms 1ms 4ms
2       2.2.2.2 23ms 3ms 2ms
3       9.9.9.1 26ms 3ms 3ms
```

**Настройка LDP**

Для распределения метож LDP должен быть активирован. На R1 это делается командой

[admin@R1]>**mpls ldp set enabled=yes transport-address=9.9.9.1 lsr-id=9.9.9.1**

[admin@R1]>**mpls ldp interface add interface=ether1**

Транспортный адрес установлен как IP-адрес 9.9.9.1 интерфейса Loopback. Это заставляет маршрутизатор начинать связь сессии LDP с этим адресом и рекламировать этот адрес как транспортный адрес соседям по LDP. Объявляются интерфейсы, смотрящие внутрь MPLS –сети. ether1 смотрит в сеть 1.1.1.0/24 в MPLS-сети.

Остальные маршрутизаторы настраиваются подобным образом - LDP активируется на интерфейсах идущих в сторону других маршрутизаторов и не активируется на интерфейсах идущих в сторону сетей заказчика. Например на R5 это ether2  и ether3.

После установления LDP сессии на R5 появятся 2 LDP соседа

[admin@R5] > **mpls ldp neighbor print**

Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello,
V - vpls

| # | TRANSPORT | LOCAL-TRANSPORT | PEER | SEN |
|---|---|---|---|---|
| 0 | DOTV 9.9.9.4 | 9.9.9.5 | 9.9.9.4:0 | yes |
| 1 | DOTV 9.9.9.1 | 9.9.9.5 | 9.9.9.1:0 | yes |
| 2 | DO 9.9.9.3 | 9.9.9.5 | 9.9.9.3:0 | no |

Рассмотрим прохождение пакетов MPLS из R1 в сеть 9.9.9.4 на R4. Пакеты пойдут по пути R1-R2-R3-R4. На маршрутизаторах для маршрута на 9.9.9.4 будут назначены входные метки, которые можно посмотреть на каждом маршрутизаторе командой

**mpls local-bindings print  where dst-address=9.9.9.4/32**

Видим такие назначенные метки (у вас могут быть другие числа)

R1 24

R2 25

R3 18

R4 – метка на граничном маршрутизаторе не назначается.

Маршрутизаторы получают информацию о метках от соседей. Для маршрута на 9.9.9.4 её можно посмотреть на каждом маршрутизаторе командой

**mpls remote-bindings print  where dst-address=9.9.9.4/32**

Для нашего маршрута (1.1.1.2-2.2.2.3-3.3.3.4) видим

| | NEXTHOP | LABEL | PEER | |
|---|---|---|---|---|
| R1 | 1.1.1.2 | 25 | 9.9.9.2:0 | получил метку 25 от R2 |
| R2 | 2.2.2.3 | 18 | 9.9.9.3:0 | получил метку 18 от R3 |
| R3 | 3.3.3.4 | - | 9.9.9.4:0 | |

Исходя из маршрута, назначенных и полученных меток маршрутизаторы формируют правила переключения по меткам, которые на каждом маршрутизаторе выводит команда

**mpls forwarding-table prшte where destination=9.9.9.4/32**

Для сети 9.9.9.4 и нашего маршрута (1.1.1.2-2.2.2.3-3.3.3.4) видим

| | IN-LABEL | OUT-LABELS | IN NEXTHOP |
|---|---|---|---|
| R1 | 24 | 25 | 1.1.1.2 |
| R2 | 25 | 18 | 2.2.2.3 |
| R3 | 18 | | 3.3.3.4 |

То есть имеем такую смену меток 24-25-18.

ICMP протокол поддерживает MPLS, например можно увидеть эту смену меток

[admin@R1] > /tool traceroute 9.9.9.4 src-address=9.9.9.1

| # | ADDRESS | RT1 | RT2 | RT3 | STATUS |
|---|---|---|---|---|---|
| 1 | 1.1.1.2 | 2ms | 1ms | 1ms | <MPLS:L=25,E=0> |
| 2 | 2.2.2.3 | 1ms | 2ms | 1ms | <MPLS:L=18,E=0> |
| 3 | 9.9.9.4 | 2ms | 1ms | 3ms | |

Сохранитесь. Сделайте копию mplsvrfvpn папки текущей топологии  mpls, которая понадобится при рассмотрении VPN уровня 3 ниже.

## Настройка LDP VPLS

**Мосты**

Добавим мосты, включив в них эсернет-интефейсы, идущие в сторону заказчика

[admin@R1]>**interface bridge add name=A**
[admin@R1]>**interface bridge port add bridge=A interface=ether2**
[admin@R1]>**interface bridge add name=B**
[admin@R1]>**interface bridge port add bridge=B interface=ether3**
[admin@R4]>**interface bridge add name=A**
[admin@R4]>**interface bridge port add bridge=A interface=ether3**
[admin@R5]>**interface bridge add name=A**
[admin@R5]>**interface bridge port add bridge=A interface=ether1**
[admin@R5]>**interface bridge add name=B**
[admin@R5]>**interface bridge port add bridge=B interface=ether4**

Назначаем на компьютеры A1, A2 и A3 заказчика A адреса согласно рисунку. Аналогично и для второго заказчика. Сохранитесь. Сделайте 3 копии mplsvpls, mplsvplsbgp mplsvplcisco папки текущей топологии mpls. Используйте пункт меню GNS3 SaveProjectAS

Откройте топологию mplsvpls.

**Настройка VPLS-интерфейсов**

VPLS-интерфейсы могут быть рассмотрены как туннели похожие на интерфейсы EoIP. Для достижения прозрачного ethernet-соединения между сайтами заказчиков следует организовать следующие туннели:

2 R1-R5 (заказчик A)
3 R1-R4 (заказчик A)
4 R4-R5 (заказчик A)
5 R1-R5 (заказчик B)

Каждый туннель требует создания VPLS-интерфейсов на обоих концах. Переговоры о VPLS-туннели ведёт LDP-протокол: оба конца туннеля обмениваются метками, которые они намереваются использовать для туннеля. Проброс данных в туннели происходит путем наложения двух меток: туннельной и транспортной. VPLS-туннели настраиваются в меню **/interface vpls.** Параметр vpls-id идентифицирует туннель и должен быть уникальным для каждого туннеля между двумя пирами. Необходимые настройки

[admin@R1]>**interface vpls add name=A1toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a1 vpls-id=0:10 disabled=no**
[admin@R1]>**interface vpls add name=A1toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a1 vpls-id=0:10 disabled=no**
[admin@R1]>**interface vpls add name=B1toB2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:b1 vpls-id=0:11 disabled=no**
[admin@R4]>**interface vpls add name=A3toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a3 vpls-id=0:10 disabled=no**
[admin@R4]>**interface vpls add name=A3toA2 remote-peer=9.9.9.5 mac-address=00:00:00:00:00:a3 vpls-id=0:10 disabled=no**
[admin@R5]>**interface vpls add name=A2toA1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:a2 vpls-id=0:10 disabled=no**
[admin@R5]>**interface vpls add name=A2toA3 remote-peer=9.9.9.4 mac-address=00:00:00:00:00:a2 vpls-id=10:0 disabled=no**
[admin@R5]>**interface vpls add name=B2toB1 remote-peer=9.9.9.1 mac-address=00:00:00:00:00:b2 vpls-id=0:11 disabled=no**

Настройка VPLS-туннеля приводит к созданию динамического LDP-соседа и

установки целевой LDP-сессии. Целевая LDP-сессия это сессия, устанавливаемая между маршрутизаторами, не являющимися прямыми соседями.

[admin@R1] > **mpls ldp neighbor pr**

Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, **V - vpls**

| # | TRANSPORT | LOCAL-TRANSPORT | PEER | SEND-TARGETED | ADDRESSES |
|---|---|---|---|---|---|
| 0 | DO 9.9.9.2 | 9.9.9.1 | 9.9.9.2:0 | no | 1.1.1.2 |
| | | | | | 2.2.2.2 |
| | | | | | 9.9.9.2 |
| **1** | **DOTV 9.9.9.5** | 9.9.9.1 | 9.9.9.5:0 | yes | 4.4.4.5 |
| | | | | | 5.5.5.5 |
| | | | | | 9.9.9.5 |
| **2** | **DOTV 9.9.9.4** | 9.9.9.1 | 9.9.9.4:0 | yes | 3.3.3.4 |
| | | | | | 5.5.5.4 |
| | | | | | 9.9.9.4 |

Для туннелей назначаются метки. Заметим, что метки для IP-маршрутов также обмениваются между VPLS-пирами, хотя и не используются. Например, R4 не станет следующим хопом для любого маршрута в R1, т.е. метки туннелей, полученные от R4, не используются. Маршрутизаторы обслуживают весь обмен меток, и они готовы для использования немедленно.

Можно увидеть информацию о состоянии VPLS-интерфейса

[admin@R1]>interface vpls>monitor A1toA3 once

remote-label: 30
local-label: 31
remote-status:
transport: 9.9.9.4/32
transport-nexthop: 1.1.1.2
imposed-labels: 25,30

Видим, что R1 назначил метку 31для тунеля между R1 и R4. Удалённой стороне назначена метка 30. Таблица пробросов MPLS показывает, что метка распознана на обеих сторонах (команда **/mpls forwarding-table print**).

**Мост ethernet с VPLS**

VPLS-туннель предоставляет виртуальную ethernet-связь между маршрутизаторами. Для прозрачного соединения двух физических ethernet-сегментов они должны быть объединены в мост с VPLS-туннелем. В общем, это делается так же, как и для протокола EoIP. Для организации виртуальной сети заказчика B организовать мосты

[admin@R1]>**interface bridge port add bridge=B interface=B1toB2**

[admin@R5]>**interface bridge port add bridge=B interface=B2toB1**

Назначаем на компьютеры B1 и B2 заказчика B адреса 10.1.1.1 и 10.1.1.3 согласно рисунку. Пустим пинги из B1 в B2. Видим, что B1 в B2 связаны виртуальным ethernet-кабелем.

**Мост с расщепленным горизонтом**

В нашем примере сегменты A1, A2 и A3 имеют полносвязную топологию. Если использовать мосты без протокола (R)STP, то могут возникнуть петли трафика. Например, если от A1 исходит широковещательный пакет , то он достигнет через VPLS-туннель и R5 и R4. R5, получив такой пакет, отошлёт его R4. R4 получит две копии одного пакета. Петля. Избежать петель можно так

- разрешить (R)STP , что не эффективно;

- использовать фаервол, что также не эффективно;
- использовать свойство horizon.

Базовая идея расщепленного горизонта (split horizon) – не пересылать трафик, возникающий на порту, на некоторое множество портов. Для VPLS это значит не пересылать пакеты, появившиеся на одном туннеле в другой туннель, так как для этого есть прямая связь.

Пакет, принятый на порту со значением horizon=X не пробрасывается на порты с тем же значением horizon=X. Так в случае полносвязной топологии VPLS-туннелей, каждый маршрутизатор должен иметь одинаковое значение свойства horizon для VPLS-туннелей, помещённых в один мост.

Для организации виртуальной сети заказчика А мосты на R1 организовать можно так

[admin@R1]>**interface bridge add name=A**
[admin@R1]>**interface bridge port add bridge=A interface=A1toA2 horizon=1**
[admin@R1]>**interface bridge port add bridge=A interface=A1toA3 horizon=1**

Аналогичным образом произведите конфигурацию на R4 и R5. Свойство horizon не конфигурируется на физическом ethernet-интерфейсе. Свойство horizon имеет локальное значение и не передаётся по сети. Поэтому везде можно установить для него одинаковые значения.

Пустим пинги между адресами 10.1.1.1, 10.1.2.1 и 10.1.1.3. A1, A2 иA3 связаны виртуальным ethernet-кабелем.

**Фильтрация меток**

Не все привязки меток нам нужны. Нет нужды в обмене метками между R1 и R5 или R1 и R4, так как нет шансов, что они будут использоваться. Нужны метки только для маршрутов на сети с маской /32 (loopback интерфейсы). Можно использовать фильтрацию привязок меток для распределения определённого множества меток. Можно осуществлять фильтрацию или по выходу или по выходу.

В данном примере разрешим маршрутизаторам выдавать информацию о метках, только для пакетов с префиксом 9.9.9.0/24. Выполним на всех маршрутизаторах команды

**/mpls ldp advertise-filter add prefix=9.9.9.0/24 advertise=yes**
**/mpls ldp advertise-filter add prefix=0.0.0.0/0 advertise=no**

Пепегрузим MPLS мягко на всех маршрутизаторах, убив соседей

/mpls ldp neighbor>**remove [find]**

Можно продолжить более тонкую настойку фильтров. Так на R1

[admin@R1] **mpls ldp advertise-filter> print**

```
Flags: X - disabled
 #   PREFIX          NEIGHBOR       ADVERTISE
 0   0.0.0.0/0       9.9.9.4        no
 1   0.0.0.0/0       9.9.9.5        no
 2   9.9.9.0/24      all            yes
   3   0.0.0.0/0       all            no
```

Получим минимальные таблицы привязок

[admin@R1] > **mpls local-bindings print**

```
Flags: X - disabled, A - advertised, D - dynamic, L - local-route, G - gateway-route, e - egress
 #     DST-ADDRESS       LABEL     PEERS
 0 ADLe 9.9.9.1/32        impl-null  9.9.9.2:0
 1 ADG  9.9.9.3/32        40         9.9.9.2:0
 2 ADG  9.9.9.4/32        41         9.9.9.2:0
```

```
 3 ADG  9.9.9.2/32       42        9.9.9.2:0
 4 ADG  9.9.9.5/32       43        9.9.9.2:0
[admin@R1] > /mpls remote-bindings print
Flags: X - disabled, A - active, D - dynamic
 #    DST-ADDRESS       NEXTHOP        LABEL     PEER
 0 AD 9.9.9.2/32        1.1.1.2        impl-null  9.9.9.2:0
 1  D 9.9.9.1/32                       24        9.9.9.2:0
 2 AD 9.9.9.3/32        1.1.1.2        25        9.9.9.2:0
 3 AD 9.9.9.4/32        1.1.1.2        26        9.9.9.2:0
 4 AD 9.9.9.5/32        1.1.1.2        27        9.9.9.2:0
```

Заметим, что не следует фильтровать метки между R4 и R5, хотя это и концы туннелей. В случае, когда связь между R3 и R5 пропадёт, весь трафик на R5 от R1 пойдёт через R4. И если мы запретили обмен меток между R4 и R5, то R4 не сможет пробросить MPLS трафик на R5.

## VPLS, основанные на BGP

Благодаря своей статической природе, VPLS-туннели, основанные на LDP имеют проблемы масштабируемости, которые возрастают при увеличении числа сайтов, подключённых по VPLS. Одной из проблем является требование сохранения полносвязной торологии LDP-туннелей между сайтами, формирующими VPLS. В случае если число узлов в VPLS высока, добавление нового сайта в существующим VPLS может стать обременительным для администратора сети. Например, добавление нового сайта к существующей VPLS будет означать настройку маршрутизатора (к которому присоединён новый сайт) для создания необходимого количества двусторонних VPLS-туннелей, что потребует настройки многих маршрутизаторов на других концах туннелей.

В общем,VPLS, основанные на BGP, служат двум целям:

- автоматического обнаружения: нет необходимости настраивать каждый VPLS маршрутизатор со всеми удаленных конечными точками туннелей VPLS;
- сигнализации: метки, используемые для туннелей VPLS на удаленных конечных точках, распространяются в одном и том же обновлении BGP, это значит, что нет необходимости для целевых сессий LDP между конечныеми точками туннеля, как в случае VPLS на основе LDP.

Правильная настройка VPLS, основанных на BGP, позволяет избежать конфигурации маршрутизаторов, которые не подключены к новому сайту.

При BGP маршрутизаторы обмениваются сообщениями NLRI (Network Layer Reachability Information), которые могут содержать некую многопротокольную информацию, например о VPLS. Для организации BGP VPLS необходимо обеспечить доставку многопротокольной NLRI между VPLS-маршрутизаторами. Это можно сделать либо путём установки BGP-сессий между всеми парами VPLS-маршрутизаторов, либо использованием отражателя маршрутов. В первом случае преимущество BGP VPLS над LDP VPLS сомнительно: при добавлении нового сайта в VPLS необходимо настроить BGP-пиры на всех маршрутизаторах, формирующих VPLS. При использовании отражателя маршрутов добавление нового сайта к VPLS становится более простым - маршрутизатор, к которому присоединён новый сайт, должен только установить BGP сессию с отражателем маршрутов. На остальных маршрутизаторах (кроме отражателя маршрутов) настроек не требуется. Сам маршрутизатор отражатель маршрутов может также участвовать в формировании VPLS.

BGP VPLS это лишь метод обмена метками для туннелей VPLS, а не метод обмена трафиком между конечными точками туннелей VPLS. Поэтому следует обеспечить распространение фреймов MPLS между конечными точками туннелей VPLS.

**Конфигурирование сессий BGP для VPLS сигнализации**

Возьмём топологию, mplsvplsbgp. Можно использовать экземпляр BGP по умолчанию

[admin@R1] > **routing bgp instance print**

Flags: X - disabled

0 name="default" as=65530 router-id=0.0.0.0 redistribute-connected=no redistribute-static=no

redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no out-filter=""

client-to-client-reflection=yes ignore-as-path-len=no

Для обеспечения распространения VPLS NLRI по BGP должна быть использована многопротокольная возможность BGP. Это осуществляется установкой в BGP-пире для свойства **address-families** значения **l2vpn**. Например, для настройки соединения между R1 и R5 следует выполнить команды

[admin@R1]>**routing bgp peer add remote-address=9.9.9.5 remote-as=65530 address-families=l2vpn update-source=lobridge**

[admin@R5]>**routing bgp peer add remote-address=9.9.9.1 remote-as=65530 address-families=l2vpn update-source=lobridge**

Между R1 и R5 должно установиться BGP-соединение. Об этом можно убедится, просматривая в winbox поле времени существования соединения (routing-bgp-peers-uptime).

По аналогии установите соединение между R4 и R5.

Есть несколько замечаний о конфигурации пиров BGP:

5. Для обмена VPLS NLRI нет нужды распростанять IP или Ipv6 маршруты, достаточно определить **address-families=l2vpn** (смотри в winbox в свойствах пира BGP закладку advanced);
6. для адресации пиров используются адреса мостов, то есть интерфейса "loopback" (локальный адрес определяется установкой **update-source**). BGP пир, начиная VPLS NLRI, назначает свой локальный адрес как BGP NextHop (в нашем примере R1, начиная BGP NLRI, использует 9.9.9.1 как адрес BGP NextHop). Принимающий VPLS-маршрутизатор использует полученный адрес BGP NextHop как адрес конечной точки туннеля.

**Настройка отражателя маршрутов**

В простейшем смысле отражатель маршрутов передаёт полученный BGP-маршрут без изменения BGP NextHop для маршрута. Это свойство позволяет избежать BGP-соединений типа все со всеми. Для того чтобы маршрутизатор был отражателем маршрутов для VPLS NLRI он не обязан участвовать в VPLS и даже может не поддерживать MPLS. Экземпляр BGP для отражателя маршрутов должен иметь установку **client-to-client-reflection=yes,** которая является установкой по умолчанию. Убидитесь в этом

[admin@R5]>**routing bgp instance print**

Для обеспечения должного распространения VPLS NLRI обязательно подсоединённые пиры (R1 и R4) должны быть настроены с установкой **route-reflect=yes**

Назначим маршрутизатор R5 отражателем маршрутов.

[admin@R5]>**routing bgp peer>print**

[admin@R5]>**routing bgp peer> set 0 route-reflect=yes**

[admin@R5]>**routing bgp peer> set 1 route-reflect=yes**

[admin@R5]>**routing bgp peer>print**

В настройках пиров в R1 R4 оставляем route-reflect=no.

Если надо добавить новый сайт в VPLS путём присоединения его к маршрутизатору, например R3, необходимо лишь настроить BGP пир на R3 и BGP пир на R5. Причем на R5 с установкой **route-reflect=yes**

## Настройка VPLS на основе BGP-сигнализации

### Настройка ethernet-мостов

С помощью BGP-сигнализации VPLS-туннели создаются динамически при получении правильного BGP NLRIs. Следовательно, не надо конфигурировать VPLS-интерфейсы. Для прозрачной доставки ethernet-сегментов сквозь VPLS должны быть настроены мосты.

Откройте топологию mplsvplsbgp Мосты мы уже создали (для LDP VPLS). Оставьте в них только физические ethernet-интерфейсы.

### Настройка экземпляров VPLS на основе BGP-сигнализации

Настройка экземпляра VPLS на основе BGP-сигнализации заставляет маршрутизатор рассылать информацию VPLS BGP NLRI, которая указывает, что он принадлежит некоторой VPLS. Получив такую инфомацию, другие члены той же VPLS знают, как установить VPLS-туннель с этим маршрутизатором.

Для настройки VPLS для заказчиков A и B на R1 надо выполнить команды:

[admin@R1]>**interface vpls bgp-vpls add bridge=A bridge-horizon=1 route-distinguisher=1:1 site-id=1 import-route-targets=1:1 export-route-targets=1:1**

[admin@R1]>**interface vpls bgp-vpls add bridge=B bridge-horizon=1 route-distinguisher=2:2 site-id=1 import-route-targets=2:2 export-route-targets=2:2**

**route-distinguisher** определяет значение, прикрепляемое к VPLS NLRI; получающие маршрутизаторы смотрят на это значение для образования туннеля. Это значение должно быть различно для разных VPLS.

**export-route-targets** устанавливант целевую BGP NLRI.

**import-route-targets** определения принадлежность BGP NLRI к конкретной VPLS. В простейшем случае эти значения можно брать одинаковыми. Именно они определяют принадлежность моста к конкретной VPLS.

**site-id** – должно быть уникально в пределах конкретной VPLS. Для повышения эффективности следует брать эти значения из узкого диапазона целых чисел.

**Bridge** – определяет мост к которому добавятся динамически создаваемые VPLS-туннели.

**Bridge-horizon** – служит для предотвращения петлей (см. выше).

После настройки VPLS для заказчика A на R4 с помощью команды

[admin@R4]>**interface vpls bgp-vpls add bridge=A bridge-horizon=1 route-distinguisher=1:1 site-id=4 import-route-targets=1:1 export-route-targets=1:1**

на R1 и R4 создастся динамический VPLS-туннель. Например

[admin@R4]>**interface vpls print**

Flags: X - disabled, R - running, D - dynamic, B - bgp-signaled, C - cisco-bgp-signaled

0 RDB name="vpls2" mtu=1500 l2mtu=1500 mac-address=02:98:52:97:DA:55 arp=enabled

disable-running-check=no remote-peer=**9.9.9.4** cisco-style=no cisco-style-id=0

advertised-l2mtu=1500 pw-type=raw-ethernet vpls=bgp-vpls1

и он автоматически добаввится в мост, что можно посмотреть R1 и R4 командой **interface bridge port print**

Здесь мы также получили подтверждение, что работает отражение маршрута на R5, так как нет BGP-пира между R1 и R4.

Настроим VPLS для заказчика A на R5:

[admin@R5] >**interface vpls bgp-vpls> add bridge=A bridge-horizon=1 route-distinguisher=1:1 site-id=5 import-route-targets=1:1 export-route-targets=1:1**

Это заставит R1 и R4 установить дополнительный VPLS-туннель с R5. Например, на R1:

[admin@R1] >**interface vpls print**

```
Flags: X - disabled, D - dynamic, R - running, B - bgp-signaled
 0 RDB name="vpls1" mtu=1500 mac-address=02:FA:33:C4:7A:A9 arp=enabled
     disable-running-check=no remote-peer=9.9.9.4 cisco-style=no
     cisco-style-id=0 vpls=bgp-vpls1
 1 RDB name="vpls2" mtu=1500 mac-address=02:FF:B7:0E:4B:97 arp=enabled
     disable-running-check=no remote-peer=9.9.9.5 cisco-style=no
     cisco-style-id=0 vpls=bgp-vpls1
```

[admin@R1] >**interface bridge port print**

```
 #   INTERFACE          BRIDGE          PRIORITY PATH-COST  HORIZON
 0   ether2             A                 0x80     10         none
 2 D vpls1              A                 0x80     50         1
 3 D vpls2              A                 0x80     50         1
```

Для завершения настройки добавим заказчика B к VPLS на R5:

[admin@R5] >**interface vpls bgp-vpls> add site-id=5 route-distinguisher=2:2 bridge=B bridge-horizon=1 import-route-targets=2:2 export-route-targets=2:2**

Как результат установлена полносвязная топология VPLS-туннелей, например на R5 видим

[admin@R5]>**interface vpls print**

```
Flags: X - disabled, R - running, D - dynamic, B - bgp-signaled, C - cisco-bgp-signaled 0
RDB name="vpls1" mtu=1500 l2mtu=1500 mac-address=02:08:D3:CE:77:D6 arp=enabled
disable-running-check=no remote-peer=9.9.9.4 cisco-style=no cisco-style-id=0 advertised-
l2mtu=1500 pw-type=raw-ethernet vpls=bgp-vpls1
1 RDB name="vpls2" mtu=1500 l2mtu=1500 mac-address=02:7F:E6:EE:87:B9 arp=enabled
disable-running-check=no remote-peer=9.9.9.1 cisco-style=no cisco-style-id=0 advertised-
l2mtu=1500 pw-type=raw-ethernet vpls=bgp-vpls1
2 RDB name="vpls3" mtu=1500 l2mtu=1500 mac-address=02:BD:FF:90:90:56 arp=enabled
disable-running-check=no remote-peer=9.9.9.1 cisco-style=no cisco-style-id=0 advertised-
l2mtu=1500 pw-type=raw-ethernet vpls=bgp-vpls2
```

Пустим пинги или посмотрим соседей на сайтах заказчика A. Видим, что A1, A2 и A3 связаны виртуальным ethernet-кабелем. Аналогично и для второго заказчика.

## Cisco BGP VPLS

RouterOS поддерживает автообнаружение VPLS в стиле Cisco, основанное на BGP. Этот метод предполагает использование BGP только для автоматического обнаружения других пиров, принимающих участие в VPLS. VPLS-сигнализация, в отличие от предыдущего раздела, осуществляется с помощью LDP.

Возьмём топологию, mplsvplsbgp. Мосты мы уже создали (для LDP VPLS). Оставьте в них только физические ethernet-интерфейсы. Настройте BGP-сессии и отражатель

маршрутов R5, как в предыдущем разделе, только вместо семейства адресов **l2vpn** используйте семейство **l2vpn-cisco.**

Настроим экземпляры BGP VPLS совместимые с Cisco. Создание таких экземпляров заставляет маршрутизатор вставлять в пакеты обновлений BGP информацию о VPLS NLRI:
[admin@R1]>**interface vpls cisco-bgp-vpls add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=9.9.9.1 route-distinguisher=1:1 vpls-id=1:1**

[admin@R4]>**interface vpls cisco-bgp-vpls add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=9.9.9.4 route-distinguisher=1:1 vpls-id=1:1**

[admin@R5]>**interface vpls cisco-bgp-vpls add bridge=A bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=9.9.9.5 route-distinguisher=1:1 vpls-id=1:1**

В результате каждое из устройств R1, R4 и R5 стало LDP-соседом друг другу, например

```
[admin@R4] > mpls ldp neighbor print
Flags: X - disabled, D - dynamic, O - operational, T - sending-targeted-hello, V - vpls
 #      TRANSPORT       LOCAL-TRANSPORT PEER                 SEND-TARGETED ADDRESSES
 0 DO   9.9.9.3         9.9.9.4         9.9.9.3:0            no            2.2.2.3
                                                                           3.3.3.3
                                                                           4.4.4.3
                                                                           9.9.9.3
                                                                           10.0.3.1
 1 DOTV 9.9.9.5         9.9.9.4         9.9.9.5:0            no            4.4.4.5
                                                                           5.5.5.5
                                                                           9.9.9.5
                                                                           10.0.5.1
 2 DOTV 9.9.9.1         9.9.9.4         9.9.9.1:0            yes           1.1.1.1
                                                                           9.9.9.1
                                                                           10.0.1.1
```

Автоматически на каждом устройстве появилось по два VPLS-интерфейса. Например

[admin@R4] >**interface vpls pr**

0 RDC name="vpls1" mtu=1500 l2mtu=1500 mac-address=02:70:A5:16:F9:DC arp=enabled disable-running-check=no remote-peer=9.9.9.1 vpls-id=1:1 cisco-style=no cisco-style-id=0 advertised-l2mtu=1500 pw-type=raw-ethernet vpls=cisco-bgp-vpls1 1 RDC name="vpls2" mtu=1500 l2mtu=1500 mac-address=02:F6:E6:73:B1:B7 arp=enabled disable-running-check=no remote-peer=9.9.9.5 vpls-id=1:1 cisco-style=no cisco-style-id=0 advertised-l2mtu=1500 pw-type=raw-ethernet vpls=cisco-bgp-vpls1

Эти интерфейсы доббавились сами в мост A.

Пустим пинги или посмотрим соседей на сайтах заказчика A. Видим, что A1, A2 и A3 связаны виртуальным ethernet-кабелем. Конфигурация для заказчика B производится аналогично.

## MPLS VPN 3-го уровня

Этот тип VPN основывается на VRF (Virtual Routing and Forwarding-Виртуальная маршрутизация и пересылка). RouterOS позволяет создавать много экземпляров виртуальной маршрутизации и пересылки. Это полезно для MPLS VPN, основанных на BGP. В отличие от BGP VPLS, которое работает на 2-м уровне OSI, VRF VPN работают на 3-м уровне и обмениваются IP-префиксами между маршрутизаторами. VRF решает проблем пересечения одинаковых IP-префиксов и обеспечивает конфиденциальность с помощью разделённой маршрутизации для разных VPN. VRF создаётся в меню **/ip route vrf**. Мы можете теперь

добавлять маршрут в этот VRF, определяя атрибут **routing-mark.** Если интерфейс лежит в VRF, то соответствующий присоединённый маршрут автоматически помещается в этот VRF.

Вы можете использовать многопротокольный BGP с адресным семейством VPNv4 для распределения маршрутов из таблицы маршрутов VRF — не только другим маршрутизаторам, а также в различные таблицы маршрутов в самом маршрутизаторе.

Вначале для VRF следует настроить route distinguisher. Это делается через меню **/ip route vrf**. Обычно используют соотношение один-к-одному между route distinguisher и VRF, но это не обязательно.

Установка маршрута в таблицу VRF управляется атрибутами BGP. Следует настроить список экспорта и импорта командами **/ip route vrf**, **import-route-targets** and **export-route-targets**. Список экспорта должен содержать хотя бы один route distinguisher для этого VRF. Далее следует настроить список VRF для каждого экземпляра BGP, участвующего в VRF-маршрутизации. Как только VRF для BGP, route distinguisher и список экспорта настроены, могут быть созданы активные маршруты семейства адресов VPNv4, что зависит от установок BGP. Эти маршруты устанавливаются в отдельную таблицу маршрутов и видны из меню **/routing bgp vpnv4-route**. Эти, так называемые VPNv4-маршруты, имеют префикс, который состоит из route distinguisher и префикса IPv4. Таким образом через BGP вы можете распространять одинаковые префиксы Ipv4 для разных сетей. Маршруты VPNv4 с одинаковыми префиксами будут распространятся только после должной настройки MPLS.

Откроем топологию mplsvrfvpn. Назначьте адреса для сетей 10.1.2.0/24, 10.1.2.0/24 и 10.1.3.0/24 заказчиков A и B согласно рис.3. На в компьютерах заказчика пропишите шлюзы в сторону MPLS-сети.

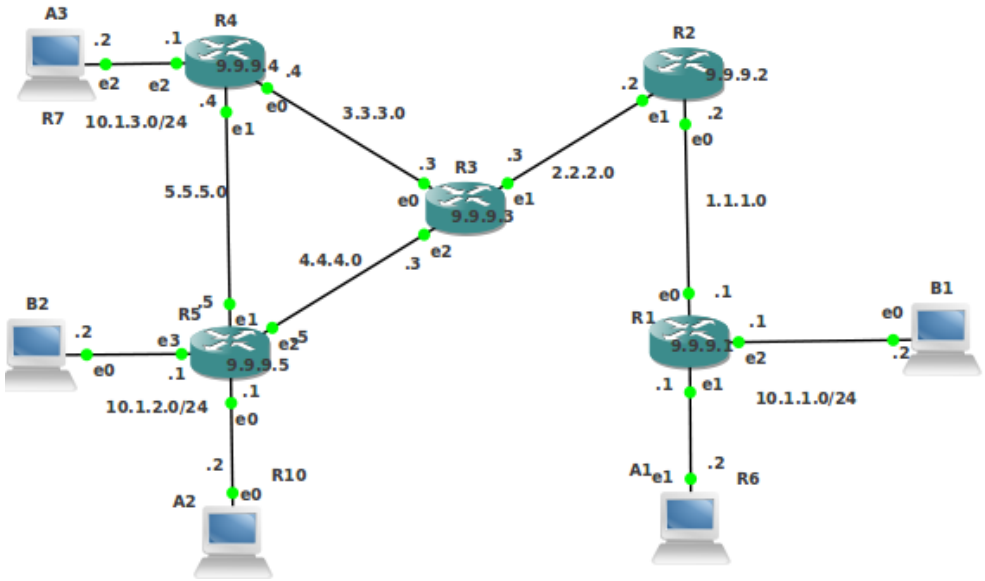



**Рис.3. Топология mplsvrfvpn**

Поместим интерфейсы в две VRF. Для заказчика A VRF определяется идентификатором 1:1, а для заказчика B - 2:2 Для VRF заказчика A (B) назначим маркер маршрутов A (B).

[admin@R1] > **ip route vrf add routing-mark=A interfaces=ether2 route-distinguisher=1:1**

**import-route-targets=1:1 export-route-targets=1:1**
[admin@R1] > **ip route vrf add routing-mark=B interfaces=ether3 route-distinguisher=2:2 import-route-targets=2:2 export-route-targets=2:2**
[admin@R4] > **ip route vrf add routing-mark=A interfaces=ether3 route-distinguisher=1:1 import-route-targets=1:1 export-route-targets=1:1**
[admin@R5 > **ip route vrf add routing-mark=A interfaces=ether1 route-distinguisher=1:1 import-route-targets=1:1 export-route-targets=1:1**
[admin@R5] > **ip route vrf add routing-mark=B interfaces=ether4 route-distinguisher=2:2 import-route-targets=2:2 export-route-targets=2:2**

Определим R5 как отражатель маршрутов

[admin@R1] > **routing bgp instance set 0 client-to-client-reflection=no**
[admin@R4] > **routing bgp instance set 0 client-to-client-reflection=no**
[admin@R5] > **routing bgp instance set 0 client-to-client-reflection=yes**

Укажем BGP, какие VRF будут участвовать в маршрутизации для семейства адресов vpnv4.

[admin@R1] > **routing bgp instance vrf add routing-mark=A redistribute-connected=yes**
[admin@R1] > **routing bgp instance vrf add routing-mark=B redistribute-connected=yes**
[admin@R4] > **routing bgp instance vrf add routing-mark=A redistribute-connected=yes**
[admin@R5] > **routing bgp instance vrf add routing-mark=A redistribute-connected=yes**
[admin@R5] > **routing bgp instance vrf add routing-mark=B redistribute-connected=yes**

Учредим BGP-сессии в адресном пространстве vpnv4. Конфигурируем R5 как отражатель маршрутов

[admin@R1] > **routing bgp peer add remote-address=9.9.9.5 remote-as=65530 route-reflect=no address-families=vpnv4 update-source=lobridge**
[admin@R4] > **routing bgp peer add remote-address=9.9.9.5 remote-as=65530 route-reflect=no address-families=vpnv4 update-source=lobridge**
[admin@R5] >**routing bgp peer add remote-address=9.9.9.1 remote-as=65530 route-reflect=yes address-families=vpnv4 update-source=lobridge**
[admin@R5] > **routing bgp peer add remote-address=9.9.9.4 remote-as=65530 route-reflect=yes address-families=vpnv4 update-source=lobridge**

Об установке BGP-соединения можно убедится, просматривая в winbox поле времени существования соединения (routing-bgp-peers-uptime)

Посмотрим на маршруты

[admin@R1] > **ip route print detail where routing-mark=A**

Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit
0 ADC dst-address=10.1.1.0/24 pref-src=10.1.1.1 gateway=ether2 gateway-status=ether2 reachable distance=0 scope=10 routing-mark=A
1 ADb dst-address=**10.1.2.0/24** gateway=9.9.9.5 gateway-status=9.9.9.5 **recursive** via 1.1.1.2 ether1 distance=200 scope=40 target-scope=30 routing-mark=A bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:1:1"
2 ADb dst-address=**10.1.3.0/24** gateway=9.9.9.4 gateway-status=9.9.9.4 **recursive** via 1.1.1.2 ether1 distance=200 scope=40 target-scope=30 routing-mark=A bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:1:1"

[admin@R1] > **ip route print detail where routing-mark=B**

Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

3 ADC dst-address=10.1.1.0/24 pref-src=10.1.1.1 gateway=ether3 gateway-status=ether3 reachable distance=0 scope=10 routing-mark=B

4 ADb dst-address=**10.1.2.0/24** gateway=9.9.9.5 gateway-status=9.9.9.5 recursive via 1.1.1.2 ether1 distance=200 scope=40 target-scope=30 routing-mark=B bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:2:2"

[admin@R1] > **routing bgp vpnv4-route pr**

```
0 L 1:1   10.1.2.0/24   9.9.9.5   ether1   53   53
1 L 1:1   10.1.3.0/24   9.9.9.4   ether1   37   37
2 L 2:2   10.1.2.0/24   9.9.9.5   ether1   54   54
3 L 1:1   10.1.1.0/24             ether2   20
4 L 2:2   10.1.1.0/24             ether3   21
```

[admin@R4] > **ip route print detail where routing-mark=A**

Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

0 ADb dst-address=**10.1.1.0/24** gateway=9.9.9.1 gateway-status=9.9.9.1 recursive via 3.3.3.3 ether1 distance=200 scope=40 target-scope=30 routing-mark=A bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:1:1"

1 ADb dst-address=**10.1.2.0/24** gateway=9.9.9.5 gateway-status=9.9.9.5 recursive via 5.5.5.5 ether2 distance=200 scope=40 target-scope=30 routing-mark=A bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:1:1"

2 ADC dst-address=10.1.3.0/24 pref-src=10.1.3.1 gateway=ether3 gateway-status=ether3 reachable distance=0 scope=10 routing-mark=A

[admin@R4] > **routing bgp vpnv4-route pr**

```
0 L 1:1   10.1.1.0/24   9.9.9.1   ether1   20   20
1 L 1:1   10.1.2.0/24   9.9.9.5   ether2   53   53
2 L 2:2   10.1.1.0/24   9.9.9.1   ether1   21   21
3 L 2:2   10.1.2.0/24   9.9.9.5   ether2   54   54
4 L 1:1   10.1.3.0/24             ether3   37
```

[admin@R5] > **ip route print detail where routing-mark=A**

Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

0 ADb dst-address=**10.1.1.0/24** gateway=9.9.9.1 gateway-status=9.9.9.1 recursive via 4.4.4.3 ether3 distance=200 scope=40 target-scope=30 routing-mark=A bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:1:1"

1 ADC dst-address=10.1.2.0/24 pref-src=10.1.2.1 gateway=ether1 gateway-status=ether1 reachable distance=0 scope=10 routing-mark=A

2 ADb dst-address=**10.1.3.0/24** gateway=9.9.9.4 gateway-status=9.9.9.4 recursive via 5.5.5.4 ether2 distance=200 scope=40 target-scope=30 routing-mark=A bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:1:1"

[admin@R5] > **ip route print detail where routing-mark=B**

Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme, B - blackhole, U - unreachable, P - prohibit

3 ADb dst-address=**10.1.1.0/24** gateway=9.9.9.1 gateway-status=9.9.9.1 recursive via 4.4.4.3 ether3 distance=200 scope=40 target-scope=30 routing-mark=B bgp-local-pref=100 bgp-origin=incomplete bgp-ext-communities="RT:2:2"

4 ADC dst-address=10.1.2.0/24 pref-src=10.1.2.1 gateway=ether4 gateway-status=ether4 reachable distance=0 scope=10 routing-mark=B

[admin@R5] > **routing bgp vpnv4-route pr**

| | | | | | |
|---|---|---|---|---|---|
| 0 L 1:1 | 10.1.3.0/24 | 9.9.9.4 | ether2 | 37 | 37 |
| 1 L 1:1 | 10.1.1.0/24 | 9.9.9.1 | ether3 | 20 | 20 |
| 2 L 2:2 | 10.1.1.0/24 | 9.9.9.1 | ether3 | 21 | 21 |
| 3 L 1:1 | 10.1.2.0/24 | | ether1 | 53 | |
| 4 L 2:2 | 10.1.2.0/24 | | ether4 | 54 | |

Получили VPN третьего уровня для заказчиков A и B. Убедитесь в этом с помощью команды телнет. Например зайдите из B1 в B2, из A1 в A2 и из A2 в A3.

Для сдачи работы следует предъявить работающие топологии **mplsvpls, mplsvplsbgp mplsvplcisco mplsvrfvpn**

# 9. Курсовой проект.

**Постановка задачи.**
**Пример выполнения**
**Быстрый старт**
**Требование к отчёту и порядок сдачи проекта.**

## Постановка задачи

У корпорации 4 филиала и в каждом по два маршрутизатора: в 0-м - R0 и R4, в 1-м - R1 и R5, во 2-м - R2 и R6, 3-м - R3 и R7. Внешние маршрутизаторы R0, R1, R2 и R3 имеют выход в интернет, а внутренние R4, R5, R6 и R7 – нет (рис. 1).

а. Адреса внутренних сетей филиалов представлены на рис. 1: 0-192.168.4*V.0/24; 1-192.168.4*V+1.0/24; 2-192.168.4*V+2.0/24; 3-192.168.4*V+3.0/24. Например, для варианта V=7 0-192.168.28.0/24 1-192.168.29.0/24 2-192.168.30.0/24 3-192.168.31.0/24. К внутренним маршрутизаторам R4, R5, R6 и R7 подсоединены локальные сети филиалов, представленные в каждом филиале одним компьютером R13, R12, R10 и R11, соответственно (рис. 2 или 3).

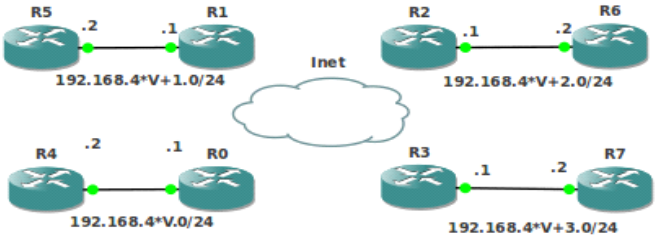



**Рис. 1. Топология для курсового проекта. 1 этап.**

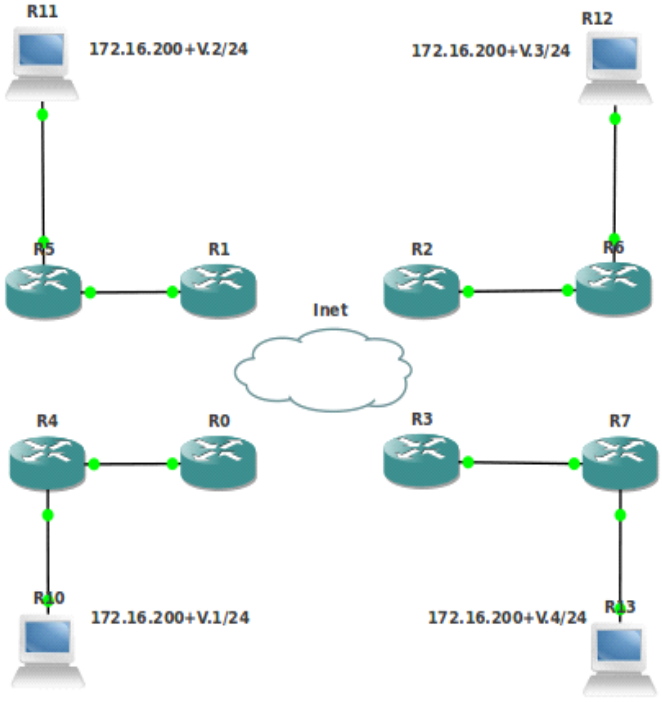



**Рис. 2. Топология для курсового проекта. 2 этап. Адреса для VPN уровня 2.**

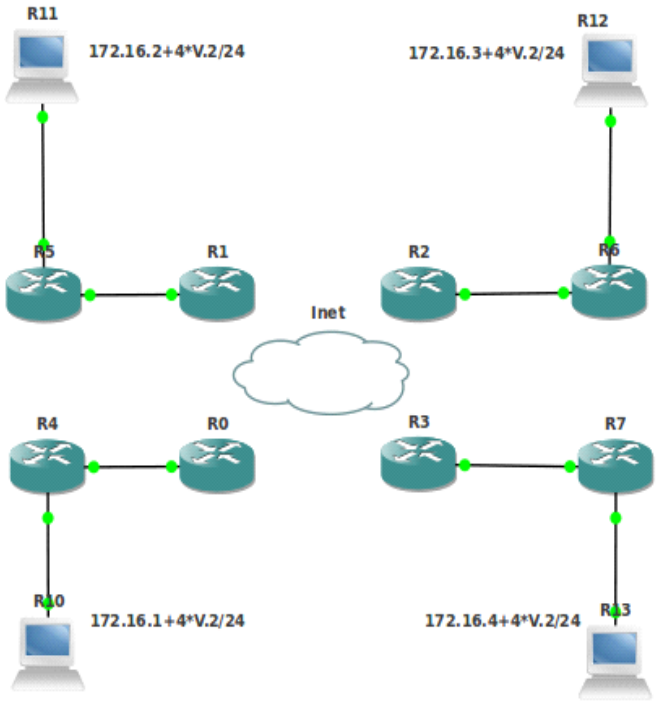



**Рис. 3. Топология для курсового проекта. 2 этап. Адреса для VPN уровня 3.**

Ставится задача поочерёдно объединить компьютеры R10, R11, R12 и R13 локальных сетей филиалов в две единые для всей корпорации виртуальные частные сети на основании технологии MPLS. Одна VPN должна быть уровня 2, а вторая – 3.

Технология для MPLS VPN уровня 3 едина – BGP VRF. Для MPLS VPN уровня 2 изо всех технологий выберем Cisco BGP VPLS.

Участники MPLS-сети должны видеть друг друга по протоколу уровня 2 и иметь возможность обмениваться метками. Поэтому для решения задачи следует объединить внутренние маршрутизаторы R4, R5, R6 и R7 в одну вспомогательную виртуальную частную сеть уровня 2 на основе технологии SSTP. Метки будут помещаться в пакеты PPP при обмене данными по SSTP.

б. Выбор схемы соединения филиалов по протоколу SSTP осуществляется из рис. 4 по формуле V%15+1.

в. В качестве протокола маршрутизации чётные варианты используют OSPF, а нечётные – RIP.

Для связи SSTP-серверов и клиентов на внутренних маршрутизаторах R4, R5, R6 и R7 между собой следует обеспечить их видимость друг другом по протоколу IP. Для этого на внешних маршрутизаторах R0, R1, R2 и R3 следует организовать преобразование исходящих и приходящих адресов.

г. Для организации MPLS VPN уровня 2 и 3 используется BGP с отражателем маршрутов. Номер маршрутизатора отражателем маршрута равен V%4+4. Например, для варианта V=7 это R7 (7%4+4=3+4=7).

д. Адреса компьютеров R10, R11, R12 и R13 для VPN уровня 2 представлены на рис. 2: R10-172.16.200+V.1/24, R11-172.16.200+V.2/24, R12-172.16.200+V.3/24 и R13-

172.16.200+V.4/24, где V-номер варианта. Например, для варианта V=7: R10-172.16.207.1/24, R11-172.16.207.2/24, R12-172.16.207.3/24 и R13-172.16.207.4/24.

е. Адреса компьютеров R10, R11, R12 и R13 для VPN уровня 3 представлены на рис. 3: 172.16.1+4*V.2/24, R11-172.16.2+4*V.2/24, R12-172.16.3+4*V.2/24 и 172.16.4+4*V.2/24, где V-номер варианта. Например, для варианта V=7: R10-172.16.29.2/24 (1+4*7=29), R11-172.16.30.2/24, R12-172.16.31.2/24 и R13-172.16.32.2/24.

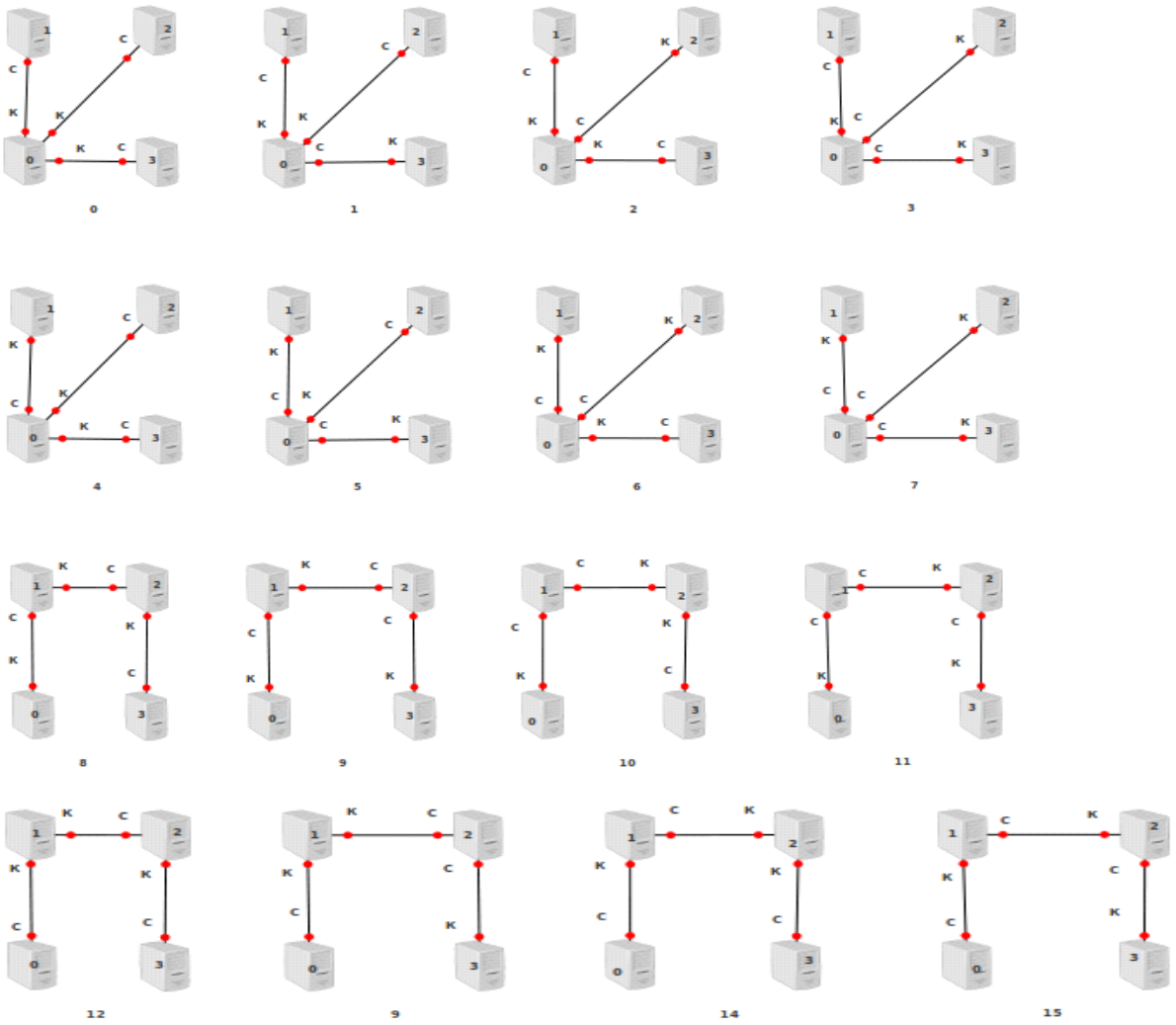


**Рис. 3. Схемы соединения филиалов 0, 1, 2 и 3 по протоколу SSTP. К-клиент, С-сервер. Фактически 0 это R4, 1- R5, 2-R6, 3-R7**.

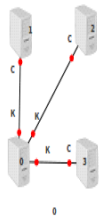
## Пример выполнения

Выполним курсовой проект для варианта V=0 и студента D=0.

Согласно варианту:

а. Адреса внутренних сетей филиалов 0-192.168.0.0/24 1-192.168.1.0/24 2-192.168.2.0/24 3-192.168.3.0/24(4*0+3).

б. Схема соединения филиалов по протоколу SSTP



в. В качестве протокола маршрутизации используем RIP.

г. В качестве отражателя маршрутов используется маршрутизатор R5.

д. Адреса компьютеров для VPN уровня 2: R10-172.16.200.1/24, R11-172.16.200.2/24, R12-172.16.200.3/24 и R13-172.16.200.4/24.

е. Адреса компьютеров для VPN уровня 3: R10-172.16.1.2/24(1+4*0=1), R11-172.16.2.2/24, R12-172.16.3.2/24 и R13-172.16.4.2/24.

Создаём в GNS3 проект с именем MPLS. Соберите в нём топологию, изображённую на рис. 1 и добавьте в каждый маршрутизатор тап-интерфейс. Например для R1

options = -net nic,vlan6 -net tap,vlan6,script=no,downscript=no,ifname=tap001

**1.** Стартуем проект. Назначим имена, например для R0

[admin@R0] >**system identity set name=R0**

Проверим у всех маршрутизаторов соседей с помощью команды **ip neighbour print.** Назначим адреса на тап-интерфейсы, например для R2

[admin@R2] >**ip address add address=10.0.2.1/24 interface=ether7**

Пропингуем Ubuntu, например для R2

[admin@R0] >**ping 10.0.2.1**

Для удобства работы откроем в терминале Ubuntu 8 табов и соединимся из Ubuntu с маршрутизаторами по протоколу telnet , например для R2

**telnet 10.D.2.1**

Дайте табам имена

Для R0 R1 R2 R2 (и только) обеспечьте взаимную связь через свою модель Интернета, назначив шлюз на тап-интерфейс Ubuntu, например для R1

[admin@R0] >**ip route add dst-address=10.0.0.0/16 gateway=10.0.1.2**

Пропингуйте R0, R1, R2 и R2 между собой по адресам тап-интерфейсов.

На каждом филиале на внутренних маршрутизаторах R4, R5, R6 И R7 сделайте шлюз на внешний маршрутизатор R0 R1 R2 R2, соответственно. Например, для филиала 1 после назначения адресов

[admin@R1] >**ip address add address=192.168.1.1/24 interface=ether1**

[admin@R5] >**ip address add address=192.168.1.2/24 interface=ether1**

и обязательной проверки

[admin@R5] >**ping 192.168.1.1**

назначаем шлюз

[admin@R5] >**ip route add gateway=192.168.1.1**

**2.** Настроим NAT для исходящих адресов на маршрутизаторах R0 R1 R2 R2, имеющих доступ в Интернет

**/ip firewall nat add chain=srcnat action=masquerade out-interface=ether7**

Из внешних маршрутизаторов R4, R5, R6 И R7 должны пинговаться адреса 10.0.4.1 10.0.5.1 10.0.6.1 10.0.7.1 тап-интерфейсов внутренних маршрутизаторов R4, R5, R6 И R7.

Настроим NAT для входящих адресов на маршрутизаторах R0 R1 R2 R3. Назначим на тап-интерфейсы R0 R1 R2 R2 дополнительные адреса 10.0.0.22/24 10.0.1.22/24 10.0.2.22/24 10.0.3.22/24. Например для R3

[admin@R3] >**ip address add address=10.0.3.22/24 interface=ether7**

Определим для R0 R1 R2 R2 предпочтительный исходящий адрес для маршрутизации 10.0.0.1/24 10.0.1.1.24 10.0.2.1/24 10.0.3.1/24. Например для R3

[admin@R3] >**ip route set 0 pref-src= 10.0.3.1**

Введём правила преобразования адресов 192.168.0.2, 192.168.1.2, 192.168.2.2, 192.168.3.2 внутренних маршрутизаторов R4, R5, R6 и R7 во внешние адреса тап-интерфейсов маршрутизаторов R0, R1, R2 и R2. Например

[admin@R0] >**/ip firewall nat add chain=dstnat action=dst-nat to-addresses= 192.168.0.2 dst-address=10.0.0.22**

[admin@R1] >**/ip firewall nat add chain=dstnat action=dst-nat to-addresses= 192.168.1.2 dst-address=10.0.1.22**

[admin@R2] >**/ip firewall nat add chain=dstnat action=dst-nat to-addresses= 192.168.2.2 dst-address=10.0.2.22**

[admin@R3] >**/ip firewall nat add chain=dstnat action=dst-nat to-addresses= 192.168.3.2 dst-address=10.0.3.22**

Проверьте тщательно преобразования. Вы должны, поочерёдно находясь на каждом из внутренних маршрутизаторов R4, R5, R6 и R7, соединятся по телнет к адресам 10.0.0.22/24 10.0.1.22/24 10.0.2.22/24 10.0.3.22/24 тап-интерфейсов внешних маршрутизаторов R0, R1, R2 и R3 и попадать в соответствующие внутренние маршрутизаторы R4, R5, R6 И R7.

**3.** Объединим филиалы с помощью VPN с использованием SSTP. Мы рассмотрим вариант 0. Для других вариантов настройка SSTP несколько отличается. Будьте внимательны.

Начнём с сертификатов. Перейдите в папку easy-rsa в Ubuntu. Создайте корневой сертификат CA (Certificate Authority), необходимый для подписи сертификатов клиента и сервера

**./pkitool --initca**

Создадим 3 сертификата s0 s1 s2 сервера, например

**./pkitool --server s1**

и 3 сертификата c0 c1 c2 клиента, например

**./pkitool c0**

Перепишем по FTP корневой сертификат на все внутренние компьютеры R4, R5, R6 и R7. Перепишем по FTP сертификаты и ключи для сервера в SSTP-сервера. Перепишем по FTP сертификаты и ключи для клиента в SSTP-клиенты.

Для топологии 0 нашего варианта 0 (рис. 3) перепишем c0 c1 c2 в R4, а s0 s1 s2 в R5, R6 и R7, соответственно

Импортируем в R4

[admin@R4] >**certificate import file-name=ca.crt**
[admin@R4] >**certificate import file-name=c0.crt**
[admin@R4] >**certificate import file-name=c1.crt**
[admin@R4] >**certificate import file-name=c2.crt**
[admin@R4] >**certificate import file-name=c0.key**
[admin@R4] >**certificate import file-name=c1.key**
[admin@R4] >**certificate import file-name=c2.key**

На запрос passphrase – просто жмём enter. Переименовываем KR сертификаты. Здесь и далее будьте внимательны с номерами после set. Для их правильного назначения используйте команду **certificate print detail.**

[admin@R4] >**certificate set 1 name=c0**
[admin@R4] >**certificate set 2 name=c1**
[admin@R4] >**certificate set 3 name=c2**
Импортируем в R5
[admin@R5 >**certificate import file-name=ca.crt**
[admin@R5 >**certificate import file-name=s0.crt**
[admin@R5 >**certificate import file-name=s0.key**
Переименовываем KR сертификат
[admin@R5 >**certificate set 1 name=s0**
Импортируем в R6
**certificate import file-name=ca.crt**
**certificate import file-name=s1.crt**
**certificate import file-name=s1.key**
Переименовываем KR сертификат
[admin@R6] >certificate set 1 name=s1
Импортируем в R7
[admin@R7] >**certificate import file-name=ca.crt**
[admin@R7] >**certificate import file-name=s2.crt**
[admin@R7] >**certificate import file-name=s2.key**
Переименовываем KR сертификат
[admin@R7] >**certificate set 1 name=s2**
На SSTP-серверах добавляем имена и пароли для SSTP-пользователей
[admin@R5>**ppp secret add name=c0 password=c0 local-address=172.16.0.1 remote-address=172.16.0.2**
[admin@R6>**ppp secret add name=c1 password=c1 local-address=172.16.0.3 remote-address=172.16.0.4**
[admin@R7>**ppp secret add name=c2 password=c2 local-address=172.16.0.5 remote-address=172.16.0.6**
Здесь адреса взяты произвольно. Каких либо рекомендаций по их назначению не даётся.
Добавляем SSTP-сервера и определяем в них сертификаты
[admin@R5 >**int sstp-server add name=s0 user=c0**
[admin@R5] >**int sstp-server server set enabled=yes certificate=s0 verify-client-certificate=ye**
[admin@R6 >**int sstp-server add name=s1 user=c1**
[admin@R6] >**int sstp-server server set enabled=yes certificate=s1 verify-client-certificate=ye**
[admin@R7 >**int sstp-server add name=s2 user=c2**
[admin@R5] >**int sstp-server server set enabled=yes certificate=s2 verify-client-certificate=ye**
Параметр user – не обязателен.
Добавляем 3 SSTP-клиента в R4 и определяем в них сертификаты. Помним, что SSTP-клиенты подсоединяются к SSTP-серверам через NAT. Поэтому вместо адресов SSTP-серверов указываем адреса соответствующих внешних маршрутизаторов.

[admin@R4] >**interface sstp-client add certificate=c0 connect-to=10.0.1.22 name=c0 user=c0 password=c0 verify-server-certificate=yes disabled=no**

[admin@R4] >**interface sstp-client add certificate=c1 connect-to=10.0.2.22 name=c1 user=c1 password=c1 verify-server-certificate=yes disabled=no**

[admin@R4] >**interface sstp-client add certificate=c2 connect-to=10.0.3.22 name=c2 user=c2 password=c2 verify-server-certificate=yes disabled=no**

На R4, R5, R6 И R7 должны появиться новые адреса из диапазона 172.16.0.1-172.16.0.6. Пропингуйте из SSTP-клиентов соответствующие им SSTP-сервера по динамически назначенным адресам

[admin@R4] >**ping 172.16.0.1**

[admin@R4] >**ping 172.16.0.3**

[admin@R4] >**ping 172.16.0.5**

С помощью команды **interface bridge add** добавим на R4, R5, R6 и R7 интерфейсы-петли в виде моста bridge1. Назначим на них адреса. Значения этих адресов никак не регламентируются

[admin@R4] >**ip address add address=4.4.4.4/32 interface=bridge1**

[admin@R5] >**ip address add address=5.5.5.5/32 interface=bridge1**

[admin@R6] >**ip address add address=6.6.6.6/32 interface=bridge1**

[admin@R7] >**ip address add address=7.7.7.7/32 interface=bridge1**

**4.** Добьемся, чтобы R4, R5, R6 и R7 видели друг друга по этим адресам. В качестве протокола маршрутизации возьмём RIP. Помним, что надо рекламировать сети, а не маршруты

[admin@R4] **>ip ad pr**

```
0   10.0.5.1/24       10.0.5.0       ether7
1   192.168.0.2/24    192.168.0.0    ether1
2 D 172.16.0.2/32     172.16.0.1     c0
3 D 172.16.0.4/32     172.16.0.3     c1
4 D 172.16.0.6/32     172.16.0.5     c2
5   4.4.4.4/32        4.4.4.4        bridge1
```

[admin@R4] >**routing rip network add network=4.4.4.4/32**

[admin@R4] >**routing rip network add network=172.16.0.1/32**

[admin@R4] >**routing rip network add network=172.16.0.3/32**

[admin@R4] >**routing rip network add network=172.16.0.5/32**

[admin@R5] >**ip ad pr**

```
0   10.0.5.1/24       10.0.5.0       ether7
1   192.168.1.2/24    192.168.1.0    ether1
2 D 172.16.0.1/32     172.16.0.2     s0
3   5.5.5.5/32        5.5.5.5        bridge1
```

[admin@R5] >**routing rip network add network=5.5.5.5/32**

[admin@R5] >**routing rip network add network=172.16.0.2/32**

[admin@R6] >**ip ad pr**

```
0   10.0.6.1/24       10.0.6.0       ether7
1   192.168.2.2/24    192.168.2.0    ether1
2 D 172.16.0.3/32     172.16.0.4     s1
3   6.6.6.6/32        6.6.6.6        bridge1
```

**[admin@R6] >routing rip network add network=6.6.6.6/32**

**[admin@R6] >routing rip network add network=172.16.0.4/32**

**[admin@R7] >ip ad pr**
```
 0   10.0.7.1/24        10.0.7.0        ether7
 1   192.168.3.2/24     192.168.3.0     ether1
 2 D 172.16.0.5/32      172.16.0.6      s2
 3   7.7.7.7/32         7.7.7.7         bridge1
```
[admin@R7] >**routing rip network add network= 7.7.7.7/32**
[admin@R7] >**routing rip network add network= 172.16.0.6/32**

Проверяем маршруты

[admin@R4] >**ip route print**
```
0 A S  0.0.0.0/0                    192.168.0.1        1
 1 ADC  4.4.4.4/32        4.4.4.4         bridge1           0
 2 ADr  5.5.5.5/32                    172.16.0.1         120
 3 ADr  6.6.6.6/32                    172.16.0.3         120
 4 ADr  7.7.7.7/32                    172.16.0.5         120
 5 ADC  10.0.4.0/24       10.0.4.1        ether7            0
 6 ADC  172.16.0.1/32     172.16.0.2      c0                0
 7 ADC  172.16.0.3/32     172.16.0.4      c1                0
 8 ADC  172.16.0.5/32     172.16.0.6      c2                0
 9 ADC  192.168.0.0/24    192.168.0.2     ether1            0
```
[admin@R5] >**ip route print**
```
0 A S  0.0.0.0/0                    192.168.1.1        1
1 ADr  4.4.4.4/32                    172.16.0.2         120
2 ADC  5.5.5.5/32        5.5.5.5         bridge1           0
3 ADr  6.6.6.6/32                    172.16.0.2         120
4 ADr  7.7.7.7/32                    172.16.0.2         120
5 ADC  10.0.5.0/24       10.0.5.1        ether7            0
6 ADC  172.16.0.2/32     172.16.0.1      s0                0
7 ADr  172.16.0.3/32                 172.16.0.2         120
8 ADr  172.16.0.5/32                 172.16.0.2         120
9 ADC  192.168.1.0/24    192.168.1.2     ether1            0
```
[admin@R6] >**ip route print**
```
0 A S  0.0.0.0/0                    192.168.2.1        1
1 ADr  4.4.4.4/32                    172.16.0.4         120
2 ADr  5.5.5.5/32                    172.16.0.4         120
3 ADC  6.6.6.6/32        6.6.6.6         bridge1           0
4 ADr  7.7.7.7/32                    172.16.0.4         120
5 ADC  10.0.6.0/24       10.0.6.1        ether7            0
6 ADr  172.16.0.1/32                 172.16.0.4         120
7 ADC  172.16.0.4/32     172.16.0.3      s1                0
8 ADr  172.16.0.5/32                 172.16.0.4         120
9 ADC  192.168.2.0/24    192.168.2.2     ether1            0
```
[admin@R7] >**ip route print**
```
0 A S  0.0.0.0/0                    192.168.3.1        1
1 ADr  4.4.4.4/32                    172.16.0.6         120
2 ADr  5.5.5.5/32                    172.16.0.6         120
3 ADr  6.6.6.6/32                    172.16.0.6         120
4 ADC  7.7.7.7/32        7.7.7.7         bridge1           0
```

5 ADC 10.0.7.0/24 10.0.7.1 ether7 0
6 ADr 172.16.0.1/32 172.16.0.6 120
7 ADr 172.16.0.3/32 172.16.0.6 120
8 ADC 172.16.0.6/32 172.16.0.5 s2 0
9 ADC 192.168.3.0/24 192.168.3.2 ether1 0

Глядя на таблицы маршрутов, мы на всех устройствах видим маршруты на сети 4.4.4.4/32, 5.5.5.5/32, 6.6.6.6/32 и 7.7.7.7/32. Мы с уверенностью в успехе запускаем расширенные пинги из R4, R5, R6 И R7 на адреса 4.4.4.4 5.5.5.5 6.6.6.6 7.7.7.7, например

[admin@R4] >**ping 5.5.5.5 src-address=4.4.4.4**
[admin@R4] >**ping 6.6.6.6 src-address=4.4.4.4**
[admin@R4] >**ping 7.7.7.7 src-address=4.4.4.4**
[admin@R5] >**ping 6.6.6.6 src-address=5.5.5.5**
[admin@R5] >**ping 7.7.7.7 src-address=5.5.5.5**
и т.д.

SSTP VPN уровня 3 настроена. Значит настроена и VPN уровня 2. Настроим MPLS поверх VPN уровня 2.

**5.** Настраиваем LDP, добавляя SSTP-интерфейсы и указывая в качестве транспортного адреса адрес моста

[admin@R4] >**mpls ldp set enabled=yes transport-address=4.4.4.4 lsr-id=4.4.4.4**
[admin@R4] >**mpls ldp interface add interface=c0**
[admin@R4] >**mpls ldp interface add interface=c1**
[admin@R4] >**mpls ldp interface add interface=c2**
[admin@R5] >**mpls ldp set enabled=yes transport-address=5.5.5.5 lsr-id=5.5.5.5**
[admin@R5] >**mpls ldp interface add interface=s0**
[admin@R6] >**mpls ldp set enabled=yes transport-address=6.6.6.6 lsr-id=6.6.6.6**
[admin@R7] >**mpls ldp interface add interface=s1**
[admin@R7] >**mpls ldp set enabled=yes transport-address=7.7.7.7 lsr-id=7.7.7.7**
[admin@R7] >**mpls ldp interface add interface=s2**

Проверим LDP-соседей командой **mpls ldp neighbor print.** Маршрутизатор R4 выдаёт такие транспортные адреса соседей: 5.5.5.5, 6.6.6.6 и 7.7.7.7. И R5 и R6 и R7 выдают адрес 4.4.4.4.

**6.** Настроим BGP. Мы отражателем маршрутов назначим маршрутизатор R5. Настроим BGP сессии к отражателю от остальных внутренних маршрутизаторов R4, R6 и R7. В качестве источника обновлений возьмём интерфейс-петлю bridge1. В качестве адреса удалённого пира используем адрес его интерфейса-петли. Предварительно проверьте доступность **remote-address.**

[admin@R5] >**routing bgp instance set 0 client-to-client-reflection=yes**
[admin@R5]>**routing bgp peer add remote-address=4.4.4.4 remote-as=65530 update-source=bridge1 route-reflect=yes**
[admin@R5]>**routing bgp peer add remote-address=6.6.6.6 remote-as=65530 update-source=bridge1 route-reflect=yes**
[admin@R5]>**routing bgp peer add remote-address=7.7.7.7 remote-as=65530 update-source=bridge1 route-reflect=yes**
[admin@R4] >**routing bgp instance set 0 client-to-client-reflection=no**
[admin@R4]>**routing bgp peer add remote-address=5.5.5.5 remote-as=65530update-source=bridge1 route-reflect=no**

[admin@R6] >**routing bgp instance set 0 client-to-client-reflection=no**

[admin@R6]>**routing bgp peer add remote-address=5.5.5.5 remote-as=65530 update-source=bridge1 route-reflect=no**

[admin@R7] >routing **bgp instance set 0 client-to-client-reflection=no**

[admin@R7]>**routing bgp peer add remote-address=5.5.5.5 remote-as=65530 update-source=bridge1 route-reflect=no**

Для R4 R5 R6 R7 в winbox должно начать изменятся время BGP-сессии в поле routing bgp peer Uptime. Обязательно проверьте. Если это поле пусто-дальнейшая работа бессмыслена. Проверьте маршрутизацию.

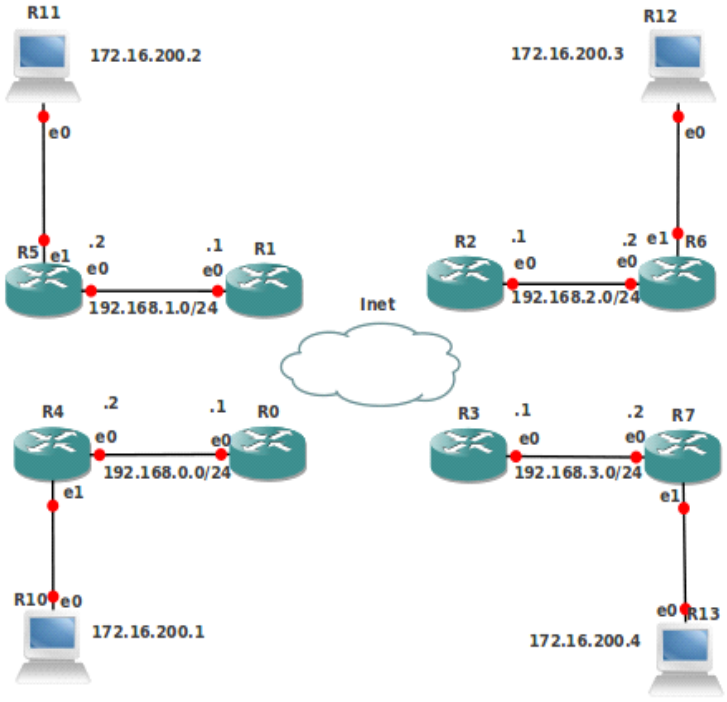



**Рис. 4 Топология CiscoBGPVPLS**

Переходим к топологии на рис.2, добавляя компьютеры R10, R11, R12 и R13 к существующей топологии и дайте им имена. Проверьте соседей у новых компьютеров и назначте адреса на тап-интерфейсы. Сохраните проект MPLS и сделайте две копии: CiscoBGPVPLS и BGPVRF.

Создадим VPN уровня 2 типа Cisco BGP VPLS. Откроем проект CiscoBGPVPLS. На R4, R5, R6 и R7 с помощью коменды **interface bridge add name=vpls** создадим мосты с именем **vpls** и добавим в каждый из них интерфейсы, идущие в сторону компьютеров R10, R11, R12 и R13 локальных сетей филиалов: **interface bridge port add bridge=vpls interface=ether2.** Назначим на компьютеры R10, R11, R12 и R13 адреса согласно варианту. Для варианта V=0 имеем(рис. 4).

[admin@R10] >**ip address add address=172.16.200.1/24 interface= ether1**

[admin@R11] >**ip address add address=172.16.200.2/24 interface= ether1**

[admin@R12] >**ip address add address=172.16.200.3/24 interface= ether1**

[admin@R13] >**ip address add address=172.16.200.4/24 interface= ether1**

Установим для BGP-сессий семейства адресов l2vpn-cisco

[admin@R4]>**routing bgp peer set 0 address-families=l2vpn-cisco**
[admin@R5]>**routing bgp peer set 0,1,2 address-families=l2vpn-cisco**
[admin@R6]>**routing bgp peer set 0 address-families=l2vpn-cisco**
[admin@R7]>**routing bgp peer set 0 address-families=l2vpn-cisco**

Для настройки Cisco VPLS BGP выполните команды

[admin@R4]>**interface vpls cisco-bgp-vpls add bridge= vpls bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=4.4.4.4 route-distinguisher=1:1 vpls-id=1:1**

[admin@R5]>**interface vpls cisco-bgp-vpls add bridge= vpls bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=5.5.5.5 route-distinguisher=1:1 vpls-id=1:1**

[admin@R6]>**interface vpls cisco-bgp-vpls add bridge= vpls bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=6.6.6.6 route-distinguisher=1:1 vpls-id=1:1**

[admin@R7]>**interface vpls cisco-bgp-vpls add bridge= vpls bridge-horizon=1 export-route-targets=1:1 import-route-targets=1:1 l2router-id=7.7.7.7 route-distinguisher=1:1 vpls-id=1:1**

Проверим на маршрутизаторах R4, R5, R6 и R7 LDP-соседей командой **mpls ldp neighbor print.** Видим, что каждый является соседом каждого.

На каждом маршрутизаторе автоматически создадаутся по три VPLS-интерфейса, например

```
[admin@R4] > /interface vpls print
Flags: X - disabled, R - running, D - dynamic,
B - bgp-signaled, C - cisco-bgp-signaled
 0 RDC name="vpls1" mtu=1500 l2mtu=1500 mac-address=02:67:B8:5E:BA:37
     arp=enabled disable-running-check=no remote-peer=5.5.5.5 vpls-id=1:1
     cisco-style=no cisco-style-id=0 advertised-l2mtu=1500
     pw-type=raw-ethernet vpls=cisco-bgp-vpls1
 1 RDC name="vpls2" mtu=1500 l2mtu=1500 mac-address=02:C2:C5:6B:F8:AE
     arp=enabled disable-running-check=no remote-peer=6.6.6.6 vpls-id=1:1
     cisco-style=no cisco-style-id=0 advertised-l2mtu=1500
     pw-type=raw-ethernet vpls=cisco-bgp-vpls1
 2 RDC name="vpls3" mtu=1500 l2mtu=1500 mac-address=02:BD:B0:81:1D:5B
     arp=enabled disable-running-check=no remote-peer=7.7.7.7 vpls-id=1:1
     cisco-style=no cisco-style-id=0 advertised-l2mtu=1500
     pw-type=raw-ethernet vpls=cisco-bgp-vpls1
```

Проверьте командой **interface bridge port print,** что эти интерфейсы добавились в мост vpls.

Устройства R10, R11, R12 и R13 видят друг друга как соседи, например

```
[admin@R10] > ip neighbor pr
 # INTERFACE ADDRESS       MAC-ADDRESS       IDENTITY  VERSION  BOARD
 5 ether1    172.16.200.2   00:AA:00:92:A8:00 R11       5.5      x86
 8 ether1    172.16.200.3   00:AA:00:2C:6B:00 R12       5.5      x86
10 ether1    172.16.200.4   00:AA:00:76:48:00 R13       5.5      x86
```

Проверьте, что устройства R13, R12, R10 и R11 пингуют друг друга по адресам (V=0) 172.16.200.1  172.16.200.2 172.16.200.3  172.16.200.4.

VPN уровня 2 типа Cisco BGP VPLS настроена.

**8.** Создадим VPN уровня 3 типа BGPVRF.  Откроем проект BGPVRF. Назначим, варианту, адреса на интерфейс ether2 внутреннего маршрутизатора, идущий в сторону компьютера локальной сети филиала. Для варианта V=0 имеем (рис. 5**)**

[admin@R4] >**ip address add address=172.16.1.1/24 interface=ether2**
[admin@R6] >**ip address add address=172.16.2.1/24 interface=ether2**
[admin@R7] >**ip address add address=172.16.3.1/24 interface=ether2**
[admin@R5] >**ip address add address=172.16.4.1/24 interface=ether2**

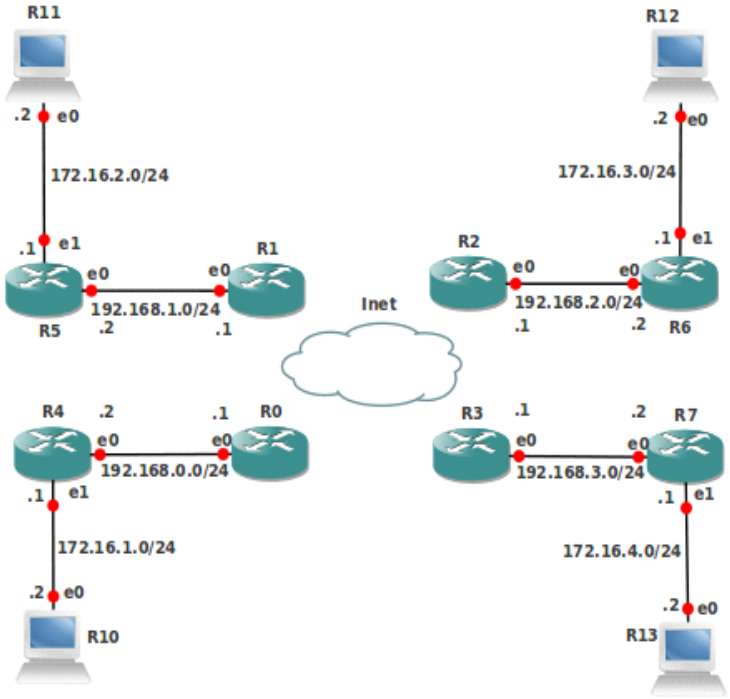



**Рис. 5. Топология BGPVRF**

Установим для BGP-сессий семейства адресов  **vpnv4**
[admin@R4]>**routing bgp peer set 0 address-families= vpnv4**
[admin@R5]>**routing bgp peer set 0,1,2 address-families= vpnv4**
[admin@R6]>**routing bgp peer set 0 address-families= vpnv4**
**[admin@R7]>routing bgp peer set 0 address-families= vpnv4**

Поместим в R4, R5, R6 и R7 интерфейс ether2, идущий в сторону локальной сети филиала  в  VRF с идентификатором  2:2 и назначим маркер маршрутов rm.  Это осуществляется командой

**ip route vrf add routing-mark= rm interfaces= ether2 route-distinguisher=2:2 import-route-targets=2:2 export-route-targets=2:2**

Укажем BGP в R4, R5, R6 и R7, что VRF с идентификатором 2:2 будут участвовать в маршрутизации для семейства адресов vpnv4 с перераспределением присоединённых маршрутов. Это осуществляется командой

**routing bgp instance vrf add routing-mark=rm redistribute-connected=yes**

Посмотрим на маршруты. Например для R4

```
[admin@R4] >ip route print detail where routing-mark=rm
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADC  dst-address=172.16.1.0/24 pref-src=172.16.1.1 gateway=vrf
        gateway-status=vrf reachable distance=0 scope=10 routing-mark=rm
 1 ADb  dst-address=172.16.2.0/24 gateway=6.6.6.6
        gateway-status=6.6.6.6 recursive via 172.16.0.3 c1 distance=200
        scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
        bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 2 ADb  dst-address=172.16.3.0/24 gateway=7.7.7.7
        gateway-status=7.7.7.7 recursive via 172.16.0.5 c2 distance=200
        scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
        bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 3 ADb  dst-address=172.16.4.0/24 gateway=5.5.5.5
        gateway-status=5.5.5.5 recursive via 172.16.0.1 c0 distance=200
        scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
        bgp-origin=incomplete bgp-ext-communities="RT:2:2"
[admin@R4] >routing bgp vpnv4-route pr
Flags: L - label-present
 #   ROUTE-DISTINGUISHER          DST-ADDRESS       GATEWAY            IN..
 0 L 2:2                          172.16.2.0/24     6.6.6.6            c1
 1 L 2:2                          172.16.3.0/24     7.7.7.7            c2
 2 L 2:2                          172.16.4.0/24     5.5.5.5            c0
 3 L 2:2                          172.16.1.0/24                        vrf
[admin@R5] >ip route print detail where routing-mark=rm
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADb  dst-address=172.16.1.0/24 gateway=4.4.4.4
        gateway-status=4.4.4.4 recursive via 172.16.0.2 s0 distance=200
        scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
        bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 1 ADb  dst-address=172.16.2.0/24 gateway=6.6.6.6
        gateway-status=6.6.6.6 recursive via 172.16.0.2 s0 distance=200
        scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
        bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 2 ADb  dst-address=172.16.3.0/24 gateway=7.7.7.7
        gateway-status=7.7.7.7 recursive via 172.16.0.2 s0 distance=200
        scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
        bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 3 ADC  dst-address=172.16.4.0/24 pref-src=172.16.4.1 gateway=vrf
```

```
      gateway-status=vrf reachable distance=0 scope=10 routing-mark=rm
 [admin@R5] >routing bgp vpnv4-route pr
Flags: L - label-present
 #   ROUTE-DISTINGUISHER          DST-ADDRESS       GATEWAY          IN..
 0 L 2:2                          172.16.1.0/24     4.4.4.4          s0
 1 L 2:2                          172.16.2.0/24     6.6.6.6          s0
 2 L 2:2                          172.16.3.0/24     7.7.7.7          s0
 3 L 2:2                          172.16.4.0/24                      vrf
[admin@R6] >ip route print detail where routing-mark=rm
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADb  dst-address=172.16.1.0/24 gateway=4.4.4.4
      gateway-status=4.4.4.4 recursive via 172.16.0.4 s1 distance=200
      scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
      bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 1 ADC  dst-address=172.16.2.0/24 pref-src=172.16.2.1 gateway=vrf
      gateway-status=vrf reachable distance=0 scope=10 routing-mark=rm
 2 ADb  dst-address=172.16.3.0/24 gateway=7.7.7.7
      gateway-status=7.7.7.7 recursive via 172.16.0.4 s1 distance=200
      scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
      bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 3 ADb  dst-address=172.16.4.0/24 gateway=5.5.5.5
      gateway-status=5.5.5.5 recursive via 172.16.0.4 s1 distance=200
      scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
      bgp-origin=incomplete bgp-ext-communities="RT:2:2"
[admin@R6] >routing bgp vpnv4-route pr
Flags: L - label-present
 #   ROUTE-DISTINGUISHER          DST-ADDRESS       GATEWAY          IN..
 0 L 2:2                          172.16.1.0/24     4.4.4.4          s1
 1 L 2:2                          172.16.3.0/24     7.7.7.7          s1
 2 L 2:2                          172.16.4.0/24     5.5.5.5          s1
 3 L 2:2                          172.16.2.0/24                      vrf
[admin@R7] >ip route print detail where routing-mark=rm
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 0 ADb  dst-address=172.16.1.0/24 gateway=4.4.4.4
      gateway-status=4.4.4.4 recursive via 172.16.0.6 s2 distance=200
      scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
      bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 1 ADb  dst-address=172.16.2.0/24 gateway=6.6.6.6
      gateway-status=6.6.6.6 recursive via 172.16.0.6 s2 distance=200
      scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
      bgp-origin=incomplete bgp-ext-communities="RT:2:2"
 2 ADC  dst-address=172.16.3.0/24 pref-src=172.16.3.1 gateway=vrf
      gateway-status=vrf reachable distance=0 scope=10 routing-mark=rm
```

```
 3 ADb  dst-address=172.16.4.0/24 gateway=5.5.5.5
      gateway-status=5.5.5.5 recursive via 172.16.0.6 s2 distance=200
      scope=40 target-scope=30 routing-mark=rm bgp-local-pref=100
      bgp-origin=incomplete bgp-ext-communities="RT:2:2"
admin@R7] >routing bgp vpnv4-route pr
Flags: L - label-present
 #   ROUTE-DISTINGUISHER          DST-ADDRESS       GATEWAY          IN..
 0 L 2:2                      172.16.1.0/24     4.4.4.4          s2
 1 L 2:2                      172.16.2.0/24     6.6.6.6          s2
 2 L 2:2                      172.16.4.0/24     5.5.5.5          s2
 3 L 2:2                      172.16.3.0/24                      vrf
```

Видим, что маршруты на сети 172.16.1.0/24 172.16.2.0/24 172.16.3.0/24 172.16.4.0/24 присутствуют во всех маршрутизаторах R4, R5, R6 и R7.

9. Назначим адреса на компьютеры, согласно варианту R10-172.16.1+4*V.2/24, R11-172.16.2+4*V.2/24, R12-172.16 .3+4*V+2.2/24, R13-172.16 .4+4*V.2/24. Для V=0 имеем

[admin@R10] >**ip address add address=172.16.1.2/24 interface=ether1**
[admin@R11] >**ip address add address=172.16.2.2/24 interface=ether1**
[admin@R12] >**ip address add address=172.16.3.2/24 interface=ether1**
[admin@R13] >**ip address add address=172.16.4.2/24 interface=ether1**

Для каждого компьютера проверьте связь по IP к маршрутизатору и затем пропишите шлюзы

[admin@R10] >**ip route add gateway=172.16.1.1**
[admin@R11] >**ip route add gateway=172.16.2.1**
[admin@R12] >**ip route add gateway=172.16.3.1**
[admin@R13] >**ip route add gateway=172.16.4.1**

Теперь R10, R11, R12 и R13 видят друг друга по адресам 172.16.1.2, 172.16.2.2 172.16.3.2 172.16.4.2. То есть MPLS VRF VPN уровня 3 функционирует.

В отличие от VPN 2 Устройства R10, R11, R12 и R13 не видят друг друга как соседи, а видят только физически присоединённые устройства, например

[admin@R10] > **ip neighbor pr**

```
 # INTERFACE ADDRESS MAC-ADDRESS      IDENTITY   VERSION    BOARD
 0 ether1   172.16.1.1    00:AA:00:C2:5C:01      R4        5.5       x86
 1 ether1   10.0.10.1     52:54:00:12:34:5C      R10       5.5       x86
 2 ether7   172.16.1.1    00:AA:00:C2:5C:01      R4        5.5       x86
 3 ether7   172.16.1.2    00:AA:00:23:42:00      R10       5.5       x86
```

**Быстрый старт**

Для изложенных проектов сделана и помещена на сайт vmg.pp.ua резервная копия. Начните работу над курсовым проектом с восстановления этой копии и изучения работающих конфигураций.

При восстановлении используйте свои тап-интерфейсы и рекомендуемые адреса для них. Это приведёт к неободимости редактировать настройки, особенно во внешних маршрутизаторах R0, R1, R2 и R3.

Конфигурации содержат сертификаты, которые не восстанавливаются. Файлы сертификатов также лежат на vmg.pp.ua. Следует вручную их импортировать в маршрутизаторы до восстановления всей конфигурации. Далее обязательно надо проверить сетевые настройки для SSTP в которых фигурируют сертификаты.

## Требование к отчёту и порядок сдачи проекта.

Отчёт содержит три скриншота топологий типа рис.1,2,3, адаптированных к своему варианту. Программу для выполнения скриншота находится в меню Applications-Accessories-TakeScreeshot.

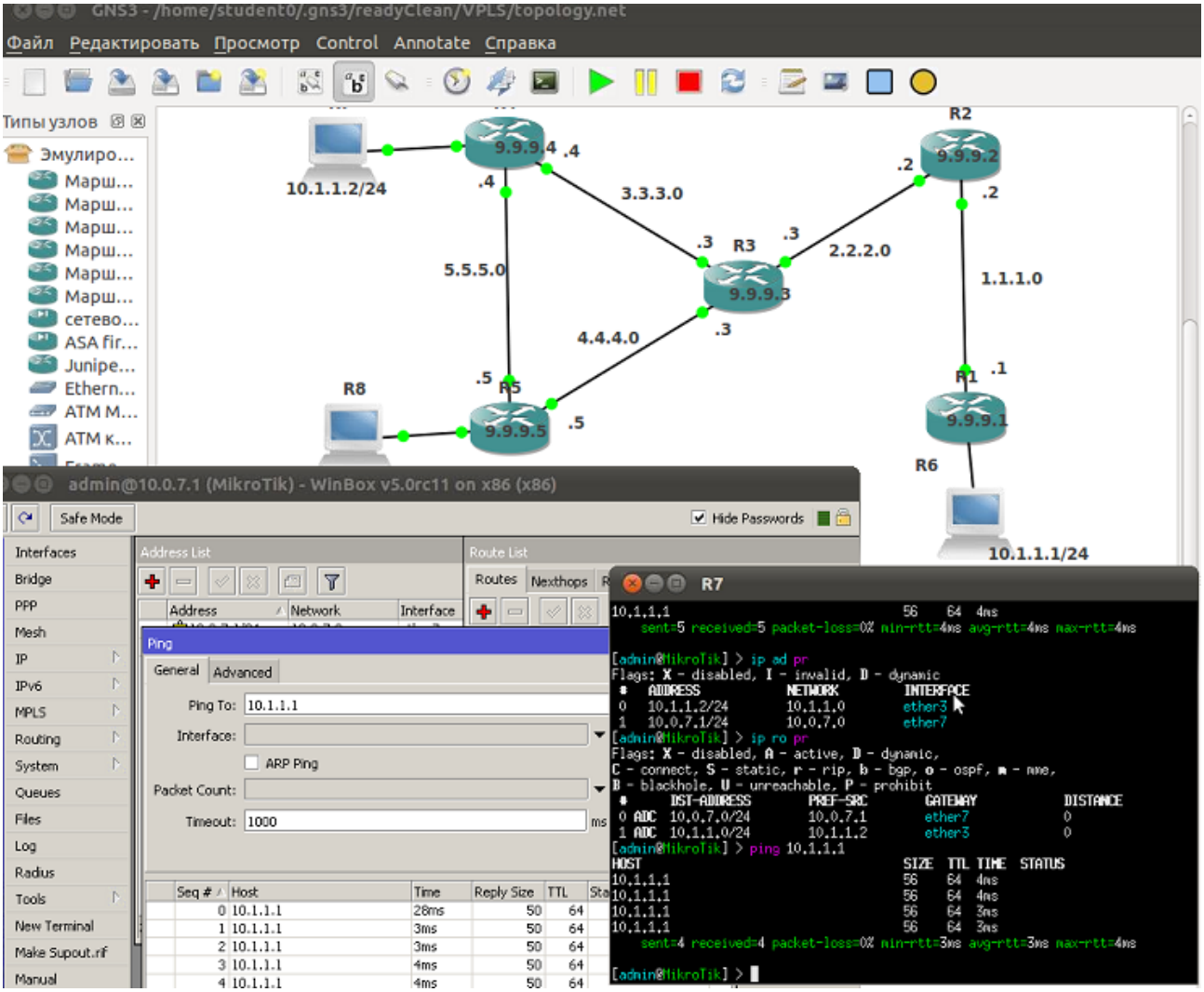
Текст отчёта повторяет вышеизложенный материал с адаптацией к своему варианту. В отчёте все результаты команд вывода типа

```
[admin@R5] >ip ad pr
0   10.0.5.1/24       10.0.5.0       ether7
1   192.168.1.2/24    192.168.1.0    ether1
2 D 172.16.0.1/32     172.16.0.2     s0
3   5.5.5.5/32        5.5.5.5        bridge1
```

должны быть текстовыми скриншотами реального вывода этих команд в консоли RouterOS для работающего проекта согласно варианту. Текстовые скриншоты создаются в консоли RouterOS с помощью мыши по технологии CopyPaste.

Для сдачи курсового проекта следует предъявить работающие проекты CiscoBGPVPLS и BGPVRF. Эти проекты должны находится в вашей папке в Ubuntu на сайте vmg.pp.ua.

Приложение




Внешний вид сетевой лаботатории

**Содержание**

**Введение** **2**

**1. Настройка домашнего компьютера** **5**
**2. Работа с Mikrotik RouterOS в Qemu и GNS3** **15**
**3. Маршрутизация** **32**
**4. DHCP, преобразование адресов и балансировка нагрузки** **47**
**5. Мосты. EoIP - эсернет через IP . VPN уровня 2** **56**
**6. Построение VPN с помощью производных от PPP протоколов и OPENVPN** **65**
**7. IPsec (Internet Protocol Security)** **118**
**8. VPN уровня 2 и 3 на основе MPLS** **132**
**9. Курсовой проект.** **150**

**1. Настройка домашнего компьютера**
Выбор платформы.
Сетевая архитектура виртуальной лаборатории
Способы выполнения лаб и курсовой по сетям
Установка GNS3, Qemu и Winbox под Ubuntu на компьютере дома
Установка tap-интерфейсов под Ubuntu
Работа в Windows
Удалённая работа из Windows и Ubuntu
**2. Работа с Mikrotik RouterOS в Qemu и GNS3**
Начальная подготовка Ubuntu.
Установка и работа с операционной системы RouterOS.
Установка и настройка GNS3
Работа с RouterOS из коммандной строки
Импорт и экспорт топологий
Доступ к удалённому маршрутизатору из домашнено компьютера
**3. Маршрутизация**
Статическая маршрутизация
Маска /32
Динамическая маршрутизация.
- RIP
- OSPF
- Перераспределение маршрутов и BGP

**4. DHCP, преобразование адресов и балансировка нагрузки**
Хост-машина Ubuntu как модель интернета
DHCP и преобразование адресов источника
Преобразование адресов приёмника
Балансировка нагрузки
**5. Мосты. EoIP - эсернет через IP . VPN уровня 2**
Мосты
EoIP. VPN уровня 2
VPN уровня 2 через NAT
**6. Построение VPN с помощью производных от PPP протоколов и OPENVPN**
PPP
Протоколы PPTP, SSTP и L2TP
OPENVPN
PPPoE
Настройка PPP, PPTP, SSTP, L2TP и OPENVPN в Routeros Mikrotik с помощью winbox

- Настройка PPP
- Настройка PPTP
- Настройка L2TP
- RSA-сертификаты
- Настройка SSTP
- Настройка OPENVPN

VPN уровня 2
- PPP
- PPTP
- L2TP
- SSTP
- OPENVPN

Особенности работы из командной строки
- PPP
- PPTP
- L2TP
- SSTP
- OPENVPN

Распределённый мост
Использование профилей пользователя.
VPN уровня 3
- Маршрутизация RIP
- Маршрутизация OSPF
- VPN 3 уровня через NAT

Настройка PPPoE

**7. IPsec (Internet Protocol Security)**

Немного теории
- Заголовок аутентификации (AH)
- Инкапсуляция зашифрованных данных (ESP)
- IKE
- Политики IPsec

Фазы IKE
Конфигурация IPsec в Mikrotik RouterOS
Шифрация соединения точка-точка.
Организация VPN типа сайт-сайт с помощью IPsec

**8. VPN уровня 2 и 3 на основе MPLS**

MPLS
VPLS
Требования для MPLS.
- IP адрес  Loopback.

IP-связь
Настройка LDP
Настройка LDP VPLS
Мосты
Настройка  VPLS-интерфейсов

Мост ethernet с VPLS
Мост  с расщепленным горизонтом
Фильтрация меток
VPLS, основанные на  BGP
Конфигурирование сессий BGP для VPLS сигнализации
Настройка отражателя маршрутов
Настройка VPLS на основе BGP-сигнализации
Настройка ethernet-мостов
Настройка экземпляров  VPLS на основе BGP-сигнализации
Cisco BGP VPLS
MPLS VPN 3-го уровня
**9. Курсовой проект.**
Постановка задачи.
Пример выполнения
Быстрый старт
Требование к отчёту и порядок сдачи проекта.