УДК 519.687

Ушаков Ю.А., Полежаев П.Н., Шухман А.Е.
Оренбургский государственный университет
E-mail: unpk@mail.ru

СОЗДАНИЕ МУЛЬТИСЕРВИСНОЙ МНОГОТОЧЕЧНОЙ VPN СЕТИ С ДИНАМИЧЕСКОЙ АВТОНАСТРОЙКОЙ

В настоящее время использование сервисов корпоративных VPN-сетей стало неотъемлемой частью любого среднего и крупного бизнеса. SOHO и SMB сегмент также используют VPN для связи с провайдерами, обеспечения VoIP, банковских транзакций. Предлагаемый подход автоматизирует процесс создания и использования сложных топологий VPN. В ходе исследования был создан прототип системы самонастраивающейся VPN-сети. Его экспериментальное исследование показало хорошие результаты по производительности и потреблению памяти, которые позволят реализовать значительную горизонтальную масштабируемость облачного сервера VPN провайдера. Предложенная концепция самонастраивающейся сети по профилю клиента показала себя жизнеспособной, планируется ее развитие в сторону управления трафиком. Также предлагается реализовать функционал full-mesh сети для клиентов без участия сервера при передаче трафика, что еще больше снизит финансовые затраты клиентов.

Ключевые слова: виртуальные частные сети, автоматическая настройка, программно-конфигурируемые сети, сеть как сервис.

Современные технологии передачи данных создали окружение для внедрения огромного числа критических бизнес технологий с сетями на основе протокола IP. Сервисы и услуги, для которых раньше использовались выделенные каналы связи и технологии, сегодня используют IP в качестве основы передачи данных. Лидируют в переходе на платформу IP сервисы телефонии, видеосвязи и выделенных сетей. Каждая организация, имеющая несколько офисов или удаленных работников, вынуждена использовать технологии объединения сотрудников в Intranet.

Для уменьшения сложности задачи объединения разнородных технологий в единый Intranet существует множество решений. Например, Cisco предлагает концепцию Borderless Network, в которой возможна реализация прозрачной сети для всех сотрудников. Множество сетевых сервисов поддерживает client-less VPN технологии для обеспечения безопасности. Банковские транзакции защищаются с помощью туннелей HTTPS. Однако, множество клиентских приложений еще не скоро начнут работать в таких средах. Кроме того, существует критичная проблема обеспечения QoS для приложений VoIP, VCS и подобных в условиях отсутствия сетевого оборудования с поддержкой QoS.

При наличии оборудования с поддержкой VPN возникает проблема сложности настройки full-mesh и double hub-and-spoke топологий, которые требуются для надежной сети и работы критичных приложений. Разные производители сетевого оборудования предлагают различные решения, которые часто несовместимы друг с другом. Как правило, применяются универсальные решения в связке со стеком IPSEC или IPv6 (белые адреса, Teredo, туннели), которые часто не поддерживаются провайдерами или сложны в настройке и требуют высокой квалификации персонала. В последние годы невозможность получения и высокая цена публичного белого адреса от провайдера является серьезной проблемой для бизнеса и делает невозможными традиционные схемы VPN.

Облачные провайдеры VPN

В настоящее время наличие Интернет доступа является одним из основных требований для успешного бизнеса. Такие сервисы, как IP АТС, виртуальные рабочие места, приложения с доступом через браузеры, облачные хранилища и бухгалтерские сервисы вошли в IT портфолио бизнеса. Облачные корпоративные VPN провайдеры, такие как Google Cloud, Amazon VPC, Aerohive Networks предоставляют единую точку входа для всех офисов и панель управления внутренней сетью, фильтрами и выходом в Интернет через VPN. Пример принципа работы облачного провайдера изображен на рисунке 1.

Основная проблема подобных топологий в отсутствии гибкости при наличии гетерогенного трафика с различными требованиями к QoS, потерям или маршрутизации. Кроме того, вся

инфраструктура передачи данных полностью зависит от провайдера, которым контролируется весь трафик.

Сетевая часть подобных систем строится на основе протоколов IPSEC, SSL/TLS или программных методов шифрования и часто требует наличия выделенного белого IP-адреса (вариант site-to-site соединения). Например, Google VPN [1] использует IPSEC с AES и ключом 128 бит, хешированием ESP SHA1 для подписи, общим ключом, заданным клиентом, и выключенной поддержкой NAT-T. Это означает, что множество предприятий не смогут воспользоваться им из-за требований законодательства по защите информации. Кроме того, параметры VPN необходимо настраивать на каждом маршрутизаторе в отдельности, нет единой централизованной системы. Маршрутизаторы, которые поддерживают IPSEC в туннельном режиме, также не всегда имеются в распоряжении малого и среднего бизнеса из-за высокой стоимости.

Многие облачные провайдеры VPN (например, Aerohive, VeloCloud), используют закрытые протоколы и программные клиенты, что сужает область их применения только конечными устройствами или серверными системами. В [2] рассматривается использование множественных связей для доступа в облачный VPN сервер с последующей маршрутизацией доступа к ресурсам. Многие разработчики решений VPN (например, OpenVPN, OpenSwan, GoVPN) [3] предоставляют инструменты для самостоятельной реализации подобных серверов на облачных платформах, имеющих тип IaaS. В этом случае вся конфигурация осуществляется силами предприятия, что не подходит для малого и среднего бизнеса. В настоящее время для них отсутствуют готовые гибкие и простые решения, ориентированные на использование массового недорого оборудования. Кроме этого, предприятиям желательно иметь в VPN канале QoS, раздельную маршрутизацию, резервирование, использование нескольких провайдеров и т.д.

Также существуют решения экономичного уровня – Cisco Meraki AutoVPN [4], Juniper AutoVPN [5] и пр. Они позволяют организовывать VPN-хабы на оборудовании с минимальными затратами на конфигурацию, но требуют наличие белого IP-адреса и выделенного оборудования, также они плохо совместимы с NAT провайдера.

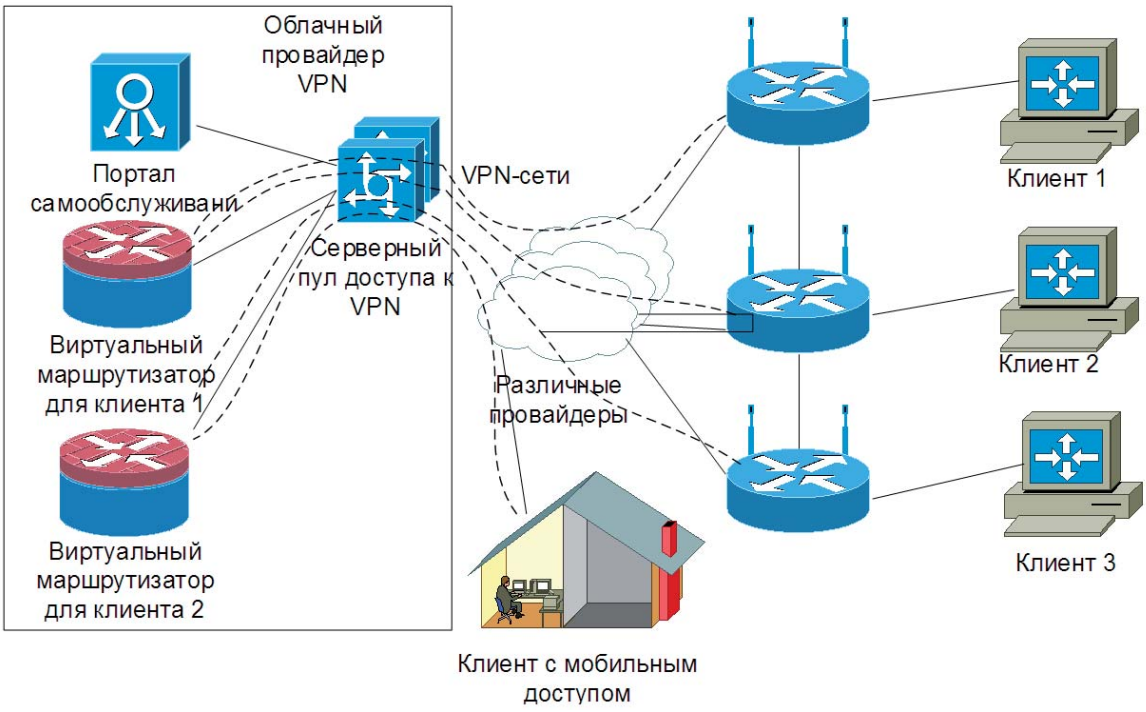

Рисунок 1. Облачный провайдер VPN

Концепция частного провайдера VPN

Основной проблемой, возникающей при реализации маршрутизации между VPN клиентами на едином сервере, является изоляция их трафика при произвольных топологиях маршрутизации и коммутации. В случае применения клиентской топологии hub-and-spoke возникает высокая нагрузка на сервер при использовании шифрования. Если взять произвольную VPN-сеть компании, то с большой вероятностью несколько клиентов будут иметь возможность прямого IP соединения в случае использования одного провайдера.

Произвольные правила коммутации и маршрутизации внутри сети клиента, пересечение адресных пространств, использование произвольных протоколов для реализации VPN побуждают провайдеров облачных VPN использовать наиболее низкий уровень OSI для управления связностью – протоколы MPLS (или его аналоги для уровня 3), VPLS (и его аналоги для уровня 2), ebtables/iptables и другие.

Очевидно, что при множестве подключений от разных клиентов могут пересекаться области IP адресации, что влечет за собой необходимость создавать виртуальный интерфейс при присоединении каждого нового пользователя. Существуют и другие решения по обеспечению изоляции сетевого контекста. Как правило, они требуют отдельного изучения и не предоставляются «из коробки» в большинстве продуктов на рынке.

Современные провайдеры предоставляют услугу Network-as-a-Service (NaaS), как статическую преднастроенную виртуальную среду. В ней с помощью некоторого интерфейса пользователя и программных решений на базе виртуальных маршрутизаторов и коммутаторов может быть реализована любая топология. Как правило, эти решения не являются простыми, дешевыми и быстро развёртываемыми. Также не исчезает необходимость настройки клиентского оборудования для организации канала до провайдера. Например, сервисы VeloCloud, Pertino просто предоставляют свое преднастроенное оборудование на базе малых серверов для организации такой сети в концепции M2M соединений. Решения Cisco DMVPN и его развитие FlexVPN, используемые очень широко в корпоративной среде, требуют только оборудование Cisco и некоторую предварительную настройку специалистом. Решения на базе маршрутизаторов не требуют дополнительных провайдеров услуг и обеспечивают работу в режиме multihomed hub-and-spoke. Для них необходимо только дорогостоящее оборудование и хотя бы один белый IP-адрес для маршрутизатора в роли хаба.

Для реализации данных технологий в среде малого и среднего бизнеса не хватает простоты, независимости от производителя и экономичности. Стандартный набор сетевого оборудования малого бизнеса на несколько офисов обычно включает в себя маршрутизаторы начального уровня, коммутаторы, управляемые через Web-интерфейс, и несколько серверов.

В рамках настоящего исследования предлагается использовать средства удаленного администрирования для автоматической настройки маршрутизаторов при создании full-mesh или hub-and-spoke сети из VPN каналов по технологиям, которые поддерживаются имеющимися у компаний маршрутизаторами. Будут использоваться решения на базе прошивок RouterOS, Vyatta, IPCop, OpenWrt, их выбор обусловлен наличием готовых программно-аппаратных маршрутизаторов на базе существующих прошивок в форм-факторе small-router и security-appliance.

Проблема горизонтального масштабирования

При использовании топологии hub-and-spoke основной проблемой является неуправляемая связность клиентов. По умолчанию в большинстве VPN серверов клиенты получают адреса из пулов адресов в одной или нескольких подсетях. Иногда существуют множественные пулы и\или адреса, выдаваемые сервером аутентификации, например, сервером RADIUS. Изоляция сетей клиентов может быть реализована посредством MPLS или VPLS на стороне сервера, однако это требует соответствующей поддержки на стороне клиента.

Существующие решения на базе технологии Cisco FlexVPN [6] решают проблему горизонтального масштабирования через балансировку сетевого трафика нескольких хабов по принципу Least Loaded Gateway (наименее загруженный шлюз). При этом сетевая изоляция клиентов осуществляется через технологию VRF (virtual routing and forwarding). Это совсем не бюджетное решение. Облачные провайдеры

NaaS технологий с поддержкой VPN используют по одной виртуальной машине на каждого клиента, что приводит к существенным накладным расходам и сказывается на стоимости и производительности не в лучшую сторону.

Доля памяти самой виртуальной машиной на всех узлах в горизонтально масштабированном кластере может быть выше доли памяти непосредственно стека NaaS в несколько раз. Существуют другие концепции разделения контекста и изоляции приложений при горизонтальном масштабировании сетевого стека. В ОС Linux существует несколько вариантов изоляции:

– VRF Linux (Network Namespaces, NetNS);

– VRF-Lite (iproute2);

– песочница (SeLinux, OpenVZ, lxc и т.д.).

Полноценный VRF NetNS позволяет использовать полностью изолированный виртуальный сетевой стек. Но для этого необходимо совместно с облачной инфраструктурой настроить выход из виртуального пространства в общий стек, что может привести к перемешиванию трафика разных пользователей.

Виртуализация окружения (песочница) позволяет изолировать приложение, использующее сеть, при этом можно создать клон физического интерфейса и работать с ним. Этот способ повышает накладные расходы и требует отдельный адрес и/или порт для работы виртуального приложения. При использовании единой точки входа потребуется распределитель потоков по пользователям в виртуальные контейнеры, для этого необходимо знание владельцев VPN потоков, а это непросто при использовании динамических адресов клиентов.

Подход VRF-Lite позволяет, с одной стороны, не виртуализировать сетевой стек, с другой – изолировать трафик клиентов на втором/третьем уровне, используя изолированные таблицы маршрутизации и ARP.

На рисунке 2 показана концепция VPN сервера с изоляцией клиентов через VRF-Lite.

Различные VPN сервисы используют виртуальные интерфейсы для маршрутизации трафика VPN клиентов. Перенаправление трафика в различные VRF позволяет его изолировать и, вместе с этим, делать это в пределах одного сервера и, при необходимости, с использованием различных протоколов VPN. При этом маршрутизация сети клиента на сервере независима от других клиентов и адресное пространство может пересекаться. Главное, что при изоляции клиентов друг от друга остается связность пользователей и офисов компании.

Горизонтальное масштабирование в этом случае осуществляется созданием кластера серверов, увеличивающегося по запросу при недостатке производительности. Все виртуальные коммутаторы OpenVSwitch объединяются в один кластер и управляются одним контроллером OpenFlow [7-8] для канальной коммутации VPN клиентов.

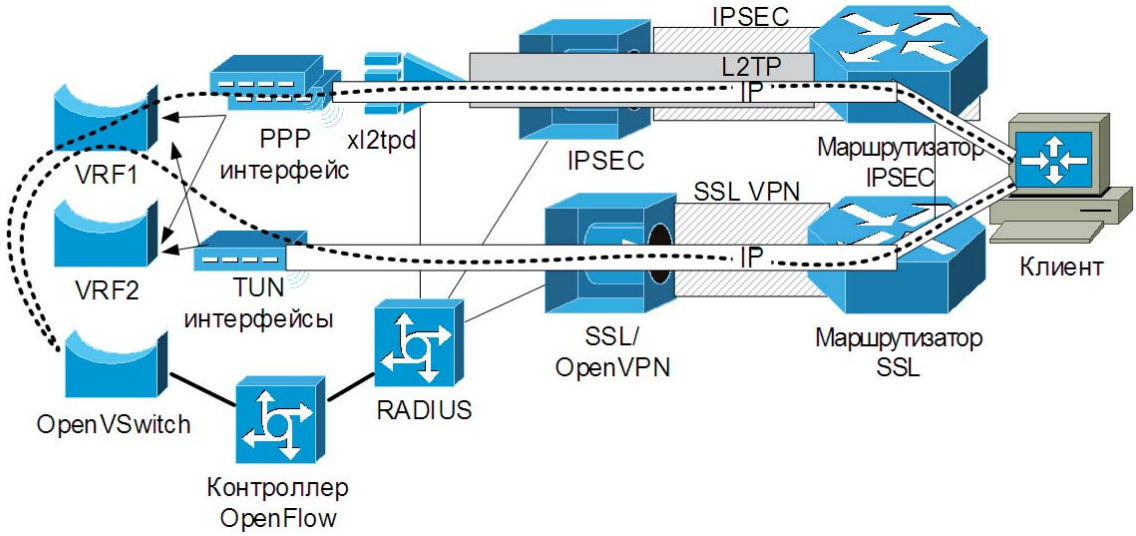

Рисунок 2. Одинарный VPN сервер с изоляцией клиентов

В таблице 1 показаны VPN технологии, которые могут быть использованы при создании сервера VPN для автонастраиваемых устройств.

Как видно, практически все ОС умеют использовать распространённые протоколы, совместимые со стандартными программами VPN распространенных дистрибутивов Linux.

Предлагаемый подход к реализациия сервера VPN провайдера

Для реализации сервера VPN провайдера необходимо сформулировать основные правила:

– доступ к VPN осуществляется по принципу NaaS: клиент регистрируется и получает возможность реализовывать любые топологии и логику маршрутизации и фильтрации;

– трафик внутри пользователей одного клиента маршрутизируется по заданным правилам;

– трафик разных клиентов не пересекается;

– при технической возможности и отсутствии правил фильтрации трафика маршрутизаторы клиента должны соединятся по схеме full-mesh;

– клиент должен иметь ПО для быстрой настройки маршрутизатора без необходимости знания технических подробностей;

– для организации VPN сети необходим только доступ к Интернет;

– маршрутизаторы должны иметь скрипт подключения к провайдеру и авто-настройки подключения;

– для смены топологии и правил фильтрации клиенту достаточно изменить их только на сервере.

Сервер доступа VPN может быть реализован на основе открытого ПО и ОС Linux Debian. Поскольку проект не ограничен конкретной ОС или производителем маршрутизаторов, то и технология VPN соединения может выбираться в зависимости от ситуации.

Для реализации VPN соединений были выбраны технологии IPSEC/L2TP (наиболее универсальный метод, поддерживаемый не только маршрутизаторами, но и многими клиентскими ОС) и OpenVPN в качестве альтернативы (в случае плохих каналов, ограничений провайдеров и т.п.)

Программная реализация IPSEC может быть выполнена на основе StrongSwan реализации стека протокола. Эта программа умеет взаимодействовать с балансировщиками нагрузки на основе IP адреса источника.

В качестве балансировщика предлагается использовать приложение OpenFlow, работающее в связке с OpenVSwitch (OVS). Из OVS собирается виртуальный коммутатор (базовая функция), который управляется OpenFlow контроллером. IP адрес, являющийся основным для пользователей, выполнен в качестве виртуального IP адреса на интерфейсе loopback, к которому есть маршрутизация и который присоединен к виртуальному коммутатору. Для резервирования используется механизм CARP (рисунок 3).

При запросе входящего соединения трафик коммутируется на первичный сервис VPN для анализа принадлежности канала. Каждое удаленное устройство имеет свои логин и пароль и принадлежность к конкретному клиенту. После первичной идентификации абонента генерируется правило OpenFlow для перенаправления на соответствующий сервер и сервис. Правило существует до пяти минут, даже после прекращения соединения, что обеспечивает кеширование правил.

При вторичной аутентификации в любом случае происходит проверка принадлежности устройства к конкретному клиенту. В случае ошибки сервер аутентификации FreeRadius через скрипт посылает команду OpenFlow контроллеру на перекоммутацию абонента.

Таблица 1. Возможные VPN технологии

ОС	Типы VPN, полностью совместимые с Linux	Поддерживаемые скрипты
RouterOS	PPTP, L2TP, L2TP+IPSEC, IPSEC, OpenVPN	Собственный язык программирования, поддерживает сетевые операции, REST, доступ ко всем настройкам
Vyatta	PPTP, L2TP, L2TP+IPSEC, IPSEC, OpenVPN	Bash скрипты, доступ ко всем компонентам системы, curl возможности
IPCop	PPTP, L2TP, L2TP+IPSEC, IPSEC, OpenVPN	Любые Linux-native скрипты
OpenWRT	PPTP, L2TP, L2TP+IPSEC, IPSEC, OpenVPN	Любые Linux-native скрипты

В случае с L2TP/IPSEC соединением, каждый абонент после подключения получает виртуальный интерфейс pppN, где N – номер абонента. Этот интерфейс должен динамически добавляться в правила iproute2 для реализации VRF. Это происходит в момент соединения в скрипте демона и содержит следующие настройки:

echo «$ClientID VRF_$ClientID»>> /etc/iproute2/rt_tables

ip route add $GW dev $DEV proto static scope link table VRF_$ClientID

ip rule add iif $DEV pref $ClientID lookup VRF_$ClientID

где $ClientID – номер клиента, $DEV – устройство ppp от абонента, $GW – шлюз по умолчанию.

Если разрешены все соединения в пределах одного клиента, то все запросы на маршруты выполняем в пределах его таблицы маршрутизатора:

ip route add unreachable default proto static table VRF_$ClientID

В другом случае (логическая топология задана жестко) устанавливаем правило отсутствия маршрута:

ip rule add unreachable iif $DEV pref 21

Затем необходимо установить маршруты для работы протоколов маршрутизации, ARP и т.д.

ip route add $TOPO[$i] dev $DEV proto static scope link table VRF_$ClientID

где $TOPO – динамически созданная таблица маршрутов для конкретного устройства клиента.

При необходимости динамической маршрутизации нужно добавить возможность передавать multicast трафик на адреса 224.0.0.X. Совместно с VRF может работать демон BIRD, реализующий стандартный OSPF протокол с поддержкой VRF. Он необходим для автоматизации маршрутизации клиентов, изолированных от остальных в разных процессах OSPF.

Реализация управления маршрутами с помощью OpenFlow построена через механизм RouteFlow, который позволяет работать с локальными ARP– и Route-таблицами Linux через протокол OpenFlow.

Проведение экспериментальных исследований

Для реализации был выбран облачный сервер, совместимый с EC2 API. Базовый сервер реализован на описанной выше связке ПО и ОС Ubuntu Server. Остальные копии сервера запускаются с помощью EC2 API при необходимости и выполняют роль ведомых для CARP и OVS (рисунок 4). Главный сервер содержит кроме OVS, VRF и OSPF еще и RADIUS функционал. Каждый сервер также имеет копию REST-based сервера для обмена данными с клиентскими устройствами.

На клиентских устройствах был установлен скрипт для инициализации первичного соеди-

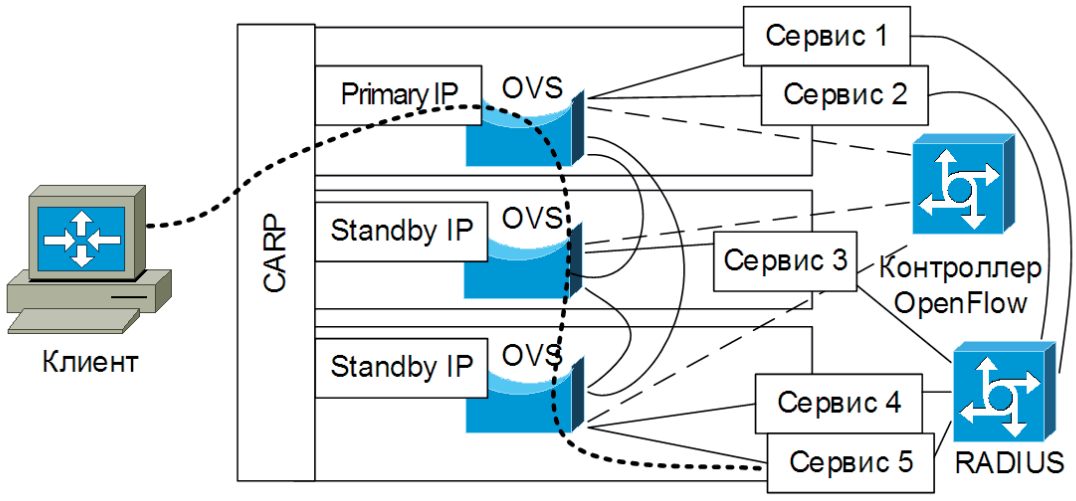

Рисунок 3. Балансировка нагрузки средствами OVS

нения, по которому они получают информацию об адресе сервера и сертификат для соединения (последний генерируется при первом запросе от клиента). Скрипт запускается при соединении к сети Интернет и сначала проверяет доступность главного сервера, затем загружает и анализирует файл конфигурации для пользователя и сохраняет на диск файл сертификата. После анализа настроек скрипт настраивает новое соединение уже для конкретного пользователя. При работе, раз в пять минут скрипт проверяет наличие обновления для конфигурационного файла.

Для моделирования большого числа соединений были использованы непроизводительные экономичные виртуальные машины в облаке с установленными RouterOS x86 v6.27 и OpenWRT. x86 v12.09. Кроме скрипта установки соединения в маршрутизаторы был записан скрипт проверки пропускной способности через каждые две минуты и сохранения результатов на сервер через REST.

Экспериментальное исследование производительности (пропускной способности) осуществлялось при различных настройках для разных клиентов (рисунок 5). Средняя скорость измерялась на пакетах размером 80 байт для UDP (эмуляция SIP), 1500 байт для TCP и UDP. Для эталона было взято соединение PPTP, обеспечивающее наименьший уровень безопасности и наибольшую скорость из протестированных соединений.

Анализ результатов показывает, что наилучшую производительность имеет протокол OpenVPN с шифрованием AES128. Самый распространенный протокол IPSEC с шифрованием 3DES в связке с L2TP показал наименьшую производительность, что вызвано существенной зависимостью скорости шифрования от производительности процессора.

Также было измерено потребление памяти при работе IPSEC и OpenVPN с включенной оптимизацией размера стека с помощью команды «ulimit –s 1000» (для ограничения стека каждого потока). В результате при загрузке главный сервер потребляет 200 Мб ОЗУ, подчиненные сервера – 112 Мб, каждая копия OpenVPN – 2.5Мб, каждый клиент OpenVPN – 0.7 Мб, демон IPSEC – суммарно 25 Мб, каждый клиент IPSEC – около 0.5 Мб.

С целью стресс-тестирования было использовано API Amazon для клонирования виртуальных машин типа m1.large, результаты приведены на рисунке 6.

Как видно по критерию использования памяти выгоднее применять IPSEC, причем потребление памяти растет практически нелинейно. Однако при использовании IPSEC существенно выше потребление процессорного времени, которое в современных тарифных планах облачных провайдеров дороже, чем использование большего объема памяти. Кроме этого, в OpenVPN есть возможность сжатия трафика. Пользователь должен самостоятельно выбрать конкретный механизм реализации VPN-сети.

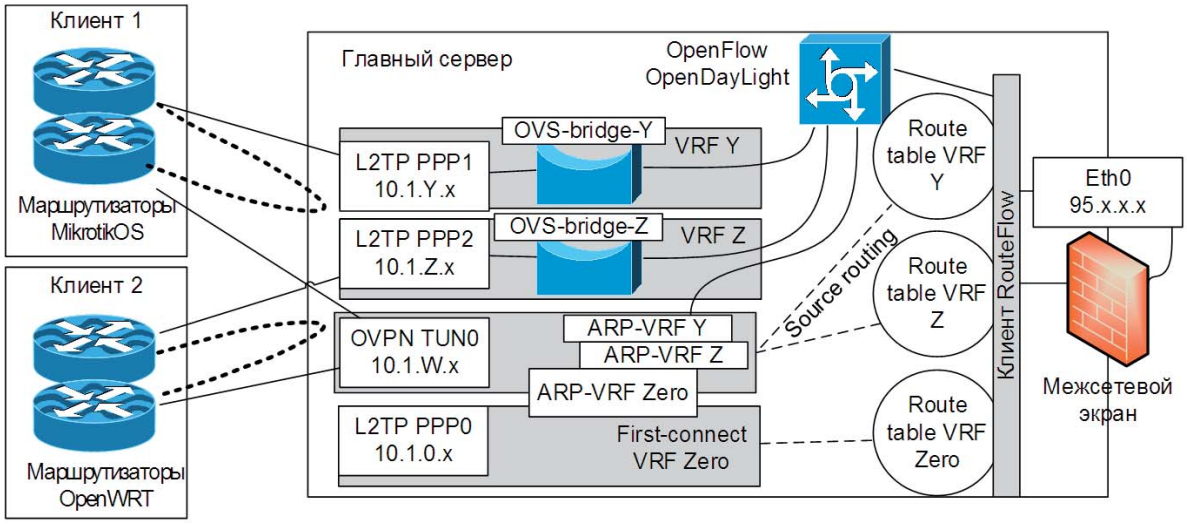

Рисунок 4. Экспериментальный стенд NaaS VPN сервера

Выводы

В результате исследования был создан прототип системы самонастраивающейся VPN-сети. Была протестирована производительность и потребление памяти. Средний период простоя процессора колебался от 0,05 до 0.07, что показывает неплохие результаты для высоконагруженного сетевого сервера. Это позволяет сократить затраты на оплату виртуальных ресурсов до 20%.

Концепция самонастраивающейся сети по профилю клиента показала себя жизнеспособной и будет развиваться в дальнейшем в сторону управления трафиком. Планируется реализовать функционал full-mesh сети для клиентов без участия сервера при передаче трафика (для тех случаев, в которых это возможно), что еще больше снизит финансовые затраты.

Для использования в реальных сетях компаний рекомендуется применение OpenVPN и IPSEC. Оба исследованных протокола имеют свои преимущества и недостатки, выбор конкретного протокола может осуществляться клиентом в личном кабинете на сервере.

Поскольку для маршрутизации и изоляции трафика используется связка OpenFlow

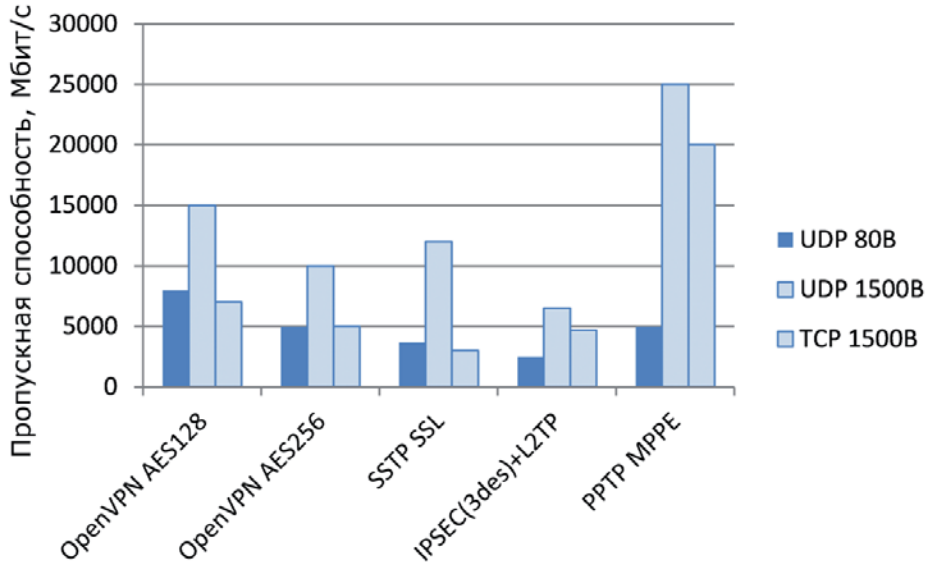

Рисунок 5. Оценка производительность протоколов VPN в тесте

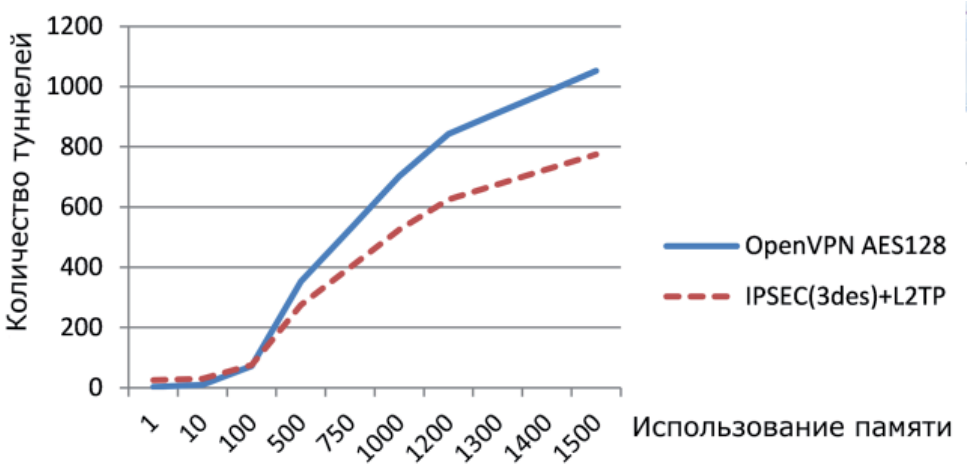

Рисунок 6. Оценка использования памяти протоколов VPN

и OpenVSwitch, VRF и OSPF, требуется реализация алгоритма предварительного расчета топологии, например, на основе муравьиных или генетических алгоритмов обхода дерева. Кроме того, дальнейшая реализация full-mesh архитектуры потребует дополнительных расчетов оптимальных маршрутов для каждой пары маршрутизаторов.

01.09.2015

Исследования выполнены при поддержке РФФИ (проекты № 15-47-02686 и № 14-07-97034), Президента Российской Федерации, стипендии для молодых ученых и аспирантов (СП-2179.2015.5).

Список литературы:

1. Google Cloud Interconnect. — Режим доступа: https://cloud.google.com/interconnect.
2. Wen-Hwa Liao, Shuo-Chun Su. A Dynamic VPN Architecture for Private Cloud Computing. Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference.– 2011.– p. 409 – 414.
3. Arthur D. Little. Reshaping the future with NFV and SDN. Bell Labs. — Режим доступа: http://www.adlittle.com/downloads/tx_adlreports/ADL_BellLabs_2015_Reshapingthefuture.pdf
4. Cisco Meraki's unique auto provisioning site-to-site VPN. — Режим доступа: https://meraki.cisco.com/technologies/auto-vpn
5. Understanding AutoVPN. — Режим доступа: http://www.juniper.net/documentation/en_US/junos12.3x48/topics/concept/security-autovpn-understanding.html
6. Cisco IOS FlexVPN Data Sheet. — Режим доступа:http://www.cisco.com/c/en/us/products/collateral/routers/asr-1000-series-aggregation-services-routers/data_sheet_c78-704277.html
7. Полежаев П.Н., Бахарева Н.Ф., Шухман А.Е. Разработка эффективного генетического алгоритма маршрутизации и обеспечения качества обслуживания для программно-конфигурируемой сети. – Вестник Оренбургского государственного университета. – 2015. – № 1 (176). С. 213-217.
8. Полежаев П.Н. Математическая модель распределенного вычислительного центра обработки данных с программно-конфигурируемыми сетями его сегментов // Вестник «Оренбургского государственного университета», 2013. – №5(154) – С. 198-204.

Сведение об авторах:

Ушаков Юрий Александрович, доцент кафедры геометрии и компьютерных наук Оренбургского государственного университета, кандидат технических наук
460018, г. Оренбург, пр-т Победы, 13, ауд. 1502, тел.: (3532) 372539,
e-mail: unpk@mail.ru

Полежаев Петр Николаевич, преподаватель кафедры компьютерной безопасности и математического обеспечения информационных систем Оренбургского государственного университета
460018, г. Оренбург, пр-т Победы, 13, ауд. 20520, тел.: (3532) 372534,
e-mail: peter.polezhaev@mail.ru

Шухман Александр Евгеньевич, заведующий кафедрой геометрии и компьютерных наук Оренбургского государственного университета, кандидат педагогических наук, доцент
460018, г. Оренбург, пр-т Победы, 13, ауд. 1502, тел.: (3532) 372539,
e-mail: shukhman@gmail.com