Титульный лист

Вместо всех выводов по разделу написать «таким образом»

Лист задания 1

Лист задания 2

содержание

Оглавление

[Введение 3](#_Toc10363053)

[Глава 1 5](#_Toc10363054)

[1.1 Динамическая и статическая маршрутизация 5](#_Toc10363055)

[1.2 Примеры использования протоколов динамической маршрутизации 6](#_Toc10363056)

[1.3 OSPF протокол динамичекской маршрутизации по состоянию канала 9](#_Toc10363057)

[1.4 Язык YANG 11](#_Toc10363058)

[1.5 Язык P4 13](#_Toc10363059)

[1.6 Централизованное управление Ansible 17](#_Toc10363060)

[Глава 2 реализация протокола динамической маршрутизаци 20](#_Toc10363061)

[2.1 Моделирование протокола OSPFv3 20](#_Toc10363062)

[2.2 Состояния маршрутизаторов в процессе установки соседских отношений OSPFv3 21](#_Toc10363063)

[2.3 Обмен маршрутами OSPFv3 23](#_Toc10363064)

[2.4 Реализация маршрутизации OSPFv3 25](#_Toc10363065)

[2.5 Тестирование работы протокола 28](#_Toc10363066)

[Заключение 30](#_Toc10363067)

[Список использованных источников 32](#_Toc10363068)

[Приложение 33](#_Toc10363069)

# Введение

Протокол IPv6 был разработан еще в 1996 году, когда стало понятно что количества адресов IPv4 диапазона будет недостаточно, в связи с быстрым ростом количества устройств в сети. Сегодня проблема стоит очень остро, так как с 2011 года мы перешили от интернета людей к интернету вещей – людей стало в интернете меньше чем вещей: умные часы, автомобили с автопилотом, умные холодильники и микроволновки, продолжать этот список можно еще очень долго.

Переход на IPv6 затянулся по причине того что изначально, оборудование не поддерживало новый протокол IPv6 и могло работать только в IPv4. Сегодня почти все телекоммуникационное оборудование может работать в дуал-стеке – режим работы оборудования когда протоколы IPv4 и IPv6 могут работать одновременно и независимо друг от друга. Теперь переход на IPv6 тормозит так же неоходимость обучать системных администраторов к работе с новым протоколом.

Адреса IPv6 имеют размер 128 бит и представляются в шестнадцатеричной системе исчисления, разбиты на группы по 8 бит разделенные двоеточиями, они сложны для запоминания, ручной записи в таблицы маршрутизации применение протоколов динамической маршрутизации необходимо. Сегодня разработано множество протоколов динамической маршрутизации для сетей IPv6.

Для использования в данной научной работе был выбран протокол OSPFv3 – это протокол динамической маршрутизации по состоянию канала для сетей IPv6 (пришедший на смену протокола OSPFv2 для сетей IPv4) разработанный рабочей группой IETF, и описанный в документе RFC2740 в 1999 году. Данный протокол широко распространен в локальных вычислительных сетях. Для построения таблицы маршрутизации в протоколах OSPFv3 и OSPFv2 применяется алгоритм Дейкстры- алгоритм поиска кратчайшего пути. Каждое соединение устройств, на которых запущен протокол имеет свою собственную стоимость маршрута. Данная стоимость может быть расчитана полностью динамически, а может быть скорректирована сетевым администратором. Каждое устройство строит путь до адреса назначения самостоятельно. Все устройства одной области имеют информацию о стоимости всех маршрутов устройств в этой области, на которых запущен процесс OSPF.

Актуальность данной работы заключается в том что у каждого вендора-производителя физических устройств и программистов операционных систем есть свое виденье описанного в документе RFC2740. За счет этого могут возникать проблемы при работе с оборудованием разных вендоров-производителей. Как правило операционные системы написаны на языке С. Язык С является громоздким, и поиск ошибок в нем сложен. В данной научной работе протокол OSPFv3 предлагается запрограммировать на языке предметной области P4. Преимущество этого языка в том, что он позволяет отделить уровень управления сетью от устройств передачи данных, а значит подходит так же для программно-определяемых сетей.

Программно определяемые сети- это сети, которых требует сегодня развитие облачных технологий, большие данные, стремительный рост сетевого трафика. Таких сетей требует сегодняшний бизнес для построения своих крупномасштабных сетей. Программно-определяемые сети (SDN) упрощают задачи, и позволяют перевести сетевые элементы под контроль настраиваемого программного обеспечения, облегчают управление.

Целью данного исследования является исправление ошибок в протоколе FRRouting, которые зафиксированы в результате проведения тестов RFC Compliance Test Report. Так как исправление ошибок сетевого протокола является сложной и трудоемкой задачей, в данной работе будет использован подход декларативного программирования. Использоваться для этого будут языки P4 и YANG, которые позволяют взаимодействовать уровнем управления не зависимо от протоколов канального уровня.

Для реализации цели будут решены следующие задачи:

-Задача установки соседских отношений OSPFv3

-Задача построения графа для вычисления алгоритма поиска кратчайшего пути Э. Дейкстры

-Задача обмена маршрутами между маршрутизаторами одной области.

Объектом исследования является протокол динамической маршрутизации для сетей IPv6 OSPFv3

Предметом исследования являются проблемы маршрутизации в сетях IPv6.

Научная новизна в исследовании применимости технологий YANG и P4 в качестве инструментария для разработки протокола маршрутизации OSPFv3.

Основные методы и средства исследования данной задачи:теория программирования и методы проектирования структур данных.

Автором произведено подробное исследование предметной области – маршрутизации в сетях IPv6, написание программы, реализующей сетевой протокол и не зависящей от протоколов канального уровня, а значит с возможностью исопльзования в сетях SDN.

# Глава 1

Маршрутизация играет важную роль в локальных и глобальных вычислительных сетях. Маршрутизация работает на третьем уровне сетевой модели OSI (Базовая Эталонная Модель Взаимодействия Открытых Систем (ЭМВОС)) — сетевая модель стека (магазина) сетевых протоколов OSI/ISO (ГОСТ Р ИСО/МЭК 7498-1-99). Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем). Работа второго уровня модели OSI предусматривает большое количесство сетевого широковещательного трафика. В больших сетях лишний широковещательный трафик представляет проблему недостаточной пропускной способности каналов связи, недостатком памяти коммутационного оборудования, выходом из строя всей сети при возникновении неполадок в широковещательном домене (примером может служить широковещательный шторм- это лавиннообразная рассылка кадров, при неспособности оборудования определять широковещательный шторм и отключать порты, на которых возникает проблема происходит заполнение всей возможной пропускной способности каналов, сеть приходит в негодное состояние. Такие шторма могут происходить по причине неисправности сетевой карты или же появления петли). Для того чтобы при возникновении проблем страдала не вся сеть, а только минимальное количество пользователей – крупные сети необходимо сегментировать. Трафик между разными сегментами второго уровня модели OSI не будет пересекаться. Для обеспечения возможности обмена трафиком между сегментами требуется маршрутизация.

## Динамическая и статическая маршрутизация

Статическая маршрутизация является простой в настройке для системного администратора. Статическая маршрутизация выгодна в маленьких сетях, так как добавление (N+1)-ой сети потребует сделать 2\*(N+1) записей о маршрутах, причём на большинстве маршрутизаторов таблица маршрутов будет различной, при N>3-4 процесс конфигурирования становится весьма трудоёмким. Использование динамической маршрутизации требует большей вычислительной мощности коммуникационного оборудования, а так же поддержку протоколов. При настройке статической маршрутизации все маршруты указываются системным администратором во время конфигурации оборудования. Для указания маршрута администратору необходимо знать адрес сети в которую необходим маршрут, адрес шлюза, через который нужно пропустить трафик для достижения сети назначения. Можно добавить метрику (цену) маршрута. Эта метрика будет учитываться при одновременном использовании статической и динамической маршрутизации. Маршрутизаторы по умолчанию используют маршрут с наименьшей метрикой. Отличительной особенностью статической маршрутизации является ее предсказуемость в каждый момент времени. В то время как реальное состояние маршрута при использовании динамической маршрутизации без дополнительных исследований будет неизвестно.

Динамическая маршрутизация эффективна в сетях с большим количеством компьютеров и маршрутизаторов, так же в сетях, в которых выполняется резервирование. Без протоколов динамической маршрутизации не может существовать одновременно несколько маршрутов к одному и тому же хосту. А переключение на другой маршрут будет занимать значительное время- для переключения будет необходимо переконфигурирование работы маршрутизаторов системным администратором в ручном режиме. К протоколам динамической маршрутизации относятся протоколы, которые предназначены для автоматизации процесса построения маршрутных таблиц маршрутизаторов. Принцип работы: маршрутизаторы с помощью устанавливаемого протоколом порядка рассылают определенную информацию из своей таблицы маршрутизаторы другим маршрутизаторам и корректируют свою таблицу при получении сообщений от других маршрутизаторов. Особенностью динамической маршрутизации является что испльзоваться она может начать только после того как отработает настроенный протокол динамической маршрутизации (то есть маршрутизатор получит маршруты от других маршрутизаторов), в отличии от статической маршрутизации, которая готова к использованию сразу после конфигурации отдельно взятого маршрутизатора.

Таким образом, в сетях с малым количеством широковещательных сегментов и не имеющих требований к быстрому переключению на резервное оборудование целесообразно использовать статическую маршрутзиацию, так как она значительно проще и предсказуемее в каждый момент времени. В крупных сетях, или сетях, в которых простой предоставления сервисов недопустим динамическая маршрутизация является обазательной для использования.

## Примеры использования протоколов динамической маршрутизации

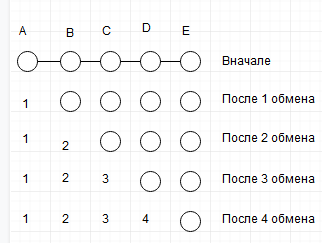
Протоколы динамической маршрутизации удобно использовать в больших распределенных сетях, в которых недопустим простой в предоставлении сервисов. Это могут быть локальные сети крупных организаций, или глобальные сети, например сеть Интернет.

Протоколы динамической маршрутизации максимально быстро перестраивают таблицу маршрутизации при возникновении проблем в сети. Так же протоколы динамической маршрутизации позволяют делать балансировку трафика динамически. Все протоколы маршрутизации делятся на две большие группы: дистанционно-векторные протоколы и протоколы по состоянию канала. Дистанционно-векторные протоколы строят таблицы маршрутизации основываясь только на расстоянии (количество hop) до маршрутизатора назначения. Ярким представителем дистанционно-векторной маршрутизации является протокол RIP (имеется спецификация на 2 версии протокола RIPv1 предназначен для сетей с классовой адресацией. Такие сети не используются сегодня в виду неэкономного использования адресов; RIPv2 предназначен для сетей с бесклассовой адресацией. Такая адресация широко распространена в глобальных и локальных сетях)

При использовании дистанционно-векторных протоколов следует обратить внимание на то, что при добавлении маршрутизаторов в локальную сеть все маршрутизаторы очень быстро «узнают» о появившемся новом маршруте, в то время как распространение информации о том что маршрутизатор «покинул» сеть, и через него больше нет маршрута в нужную сеть занимает значительное время. Данная проблема называется проблемой счета до бесконечности.

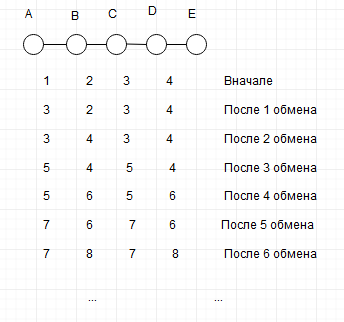
Алгоритм маршрутизации по вектору расстояний работает в теории, но обладает серьезным недостатком на практике: хотя правильный ответ в конце концов и находится, процесс его поиска может занять довольно много времени.

Если при очередном обмене векторами его сосед сообщит ему, что от него до маршрутизатора X совсем недалеко, наш маршрутизатор просто переключится Для передач маршрутизатору X на линию, проходящую через этого соседа. Таким образом, хорошая новость распространилась всего за один обмен информацией. Пример показан на рисунке (НОМЕР)



Когда в сети появляется А, остальные маршрутизаторы узнают об этом с помощью обмена векторами. После первого обмена В узнает, что у его соседа слева нулевая задержка при связи с А, а В помечает в своей таблице маршрутов, что А находится слева на расстоянии одного транзитного участка. Все остальные маршрутизаторы в этот момент еще полагают, что А выключен. При следующем обмене информацией С узнает, что у В есть путь к А длиной 1, поэтому он обновляет свою таблицу, указывая длину пути до А, равную 2, но D и Е об этом еще не знают. Таким образом, хорошие вести распространяются со скоростью один транзитный участок за один обмен векторами. Если самый длинный путь в подсети состоит из N транзитных участков, то через N обменов все маршрутизаторы подсети будут знать о включенных маршрутизаторах и заработавших линиях.

Пример распространения информации при отключении маршрутизации приведен на рисунке (НОМЕР)



Маршрутизаторы В, С, и Е находятся от А на расстоянии 1, 2, 3 и 4 соответственно. Внезапно А отключается или, возможно, происходит обрыв линии между А V. В, что с точки зрения В одно и то же.

При первом обмене пакетами В не слышит ответа от А. К счастью, С сообщает о наличии маршрута к А с метрикой 2. В не знает, что путь от С к А проходит через В. В может только предполагать, что у С около 10 выходных линий с независимыми путями к А, кратчайшая из которых имеет длину 2. Поэтому теперь В думает, что может связаться с А через С по пути длиной 3. При этом первом обмене маршрутизаторы не обновляют свою информацию об А.

При втором обмене векторами С замечает, что у всех его соседей есть путь к А длиной 3. Он выбирает один из них случайным образом и устанавливает свое расстояние до А равным 4.

Теперь должно быть понятно, почему плохие новости медленно распространяются — ни один маршрутизатор не может установить значение расстояния, более чем на единицу превышающее минимальное значение этого расстояния, хранящееся у его соседей. Таким образом, все маршрутизаторы будут до бесконечности увеличивать значение расстояния до выключенного маршрутизатора. Количество необходимых для завершения этого процесса обменов векторами можно ограничить, если установить значение этой «бесконечности» равным длине самого длинного пути плюс 1. Если расстояния в подсети измеряются во временных задержках, такую верхнюю границу выбрать сложнее. Следует выбирать достаточно большое ограничивающее значение, чтобы линия с большой задержкой не была сочтена совсем не работающей. Неудивительно, что эта проблема называется счетом до бесконечности. Было сделано несколько попыток решить ее (например, технология расщепления горизонта (Split horizon), весьма бесполезный во многих случаях, но внесенный в RFC 1058. Согласно технологии расщепления горизонта маршрутизатор не будет распространять информацию об определенном маршруте через интерфейс, который является источником данной информации. Другими словами, маршрутизатор не будет информировать о достижимости получателя своего соседа, от которого была получена информация о маршруте к получателю. Полезен для профилактики петель маршрутизации. Использование метода расщепления горизонта основано на том, что , как правило, нет необходимости в отправке информации о маршруте в обратном направлении, точнее в том направлении, по которому этот маршрут поступил. Но в некоторых конфигурациях сетей может оказаться целесообразным не использовать механизм расщепления горизонта и необходимо отключать его. Отключение производится для каждого интерфейса отдельно. Примером когда требуется отключать технологию split-horizon может являться динамическая маршрутизация в DMVPN сетях, Frame-Relay Point-to-Multipoint сетях.

Так же маршрутизаторам, использующим дистанционно-векторные протоколы маршрутизации необходимо иметь представление обо всей сети. Что требует больше памяти на устройствах, использующихся в больших сетях. Преимуществом протокола RIP является поддержка его старыми маршрутизаторами. Сегодня RIP может использоваться в качестве межшлюзового протокола внутри автономной системы (IGP)

Протоколы по состоянию канала учитывают пропускную способность всех каналов до сети назначения и основываясь на наилучшей пропускной способности строят оптимальный маршрут доставки. К представителям протоколов по состоянию канала относятся протоколы OSPF и IS-IS. (IS-IS проприетарный протокол по состоянию канала для оборудования CISCO на операционной системе IOS; Протокол OSPF открытый протокол, часто используемый в локальных вычислительных сетях организаций, в виду поддержки данного протокола большинством вендоров, выпускающими маршрутизаторами корпоративного назначения (enterprise)). Протоколы по состоянию канала позволяют роутерам иметь представление только о своих соседях, в отличие от дистанционно-векторных, которым необходимо иметь представление обо всей сети, в которой они работают, а значит требуют меньше памяти маршрутизатора.

Динамическая маршрутизация значительно упрощает задачу администрирования сетей, в которых могут происходить изменения (например, маршрутизаторы выходят из строя, либо сеть расширяется).

Таким образом, динамическая дистанционно-векторная маршрутизация гораздо старше, чем маршрутизация по состоянию канала. Использовать в современных сетях дистанционно-векторную маршрутизацию неудобно, из-за возникающей проблемы счета до бесконечности. Эта проблема не решается при помощи расщипления горизонта полностью, так как существуют сети, в которых данную функцию необходимо отключать. Динамическая маршрутзация по состоянию канала удобна для работы в локальных вычислительных сетях: она позволяет управлять балансировкой трафика, хорошей сходимостью в крупных сетях.

## OSPF протокол динамичекской маршрутизации по состоянию канала

OSPFv3 – протокол динамической маршрутизации класса “по состоянию канала”, основанный на идеях протокола OSPFv2 (IPv4), но предназначенный для сетей IPv6. Протоколы данного класса основаны на хранении маршрутизатором информации о каналах связи (link) с другими маршрутизаторами: названия маршрутизатора и соответствующей подсети, метрику расстояния канала, приоритет и т.д. Протоколы “по состоянию канала” так же учитывают пропускную способность всех каналов до сети назначения и основываясь на наилучшей пропускной способности строят оптимальный маршрут. На основе анализа структуры сети производится изменение таблиц маршрутизации узлов сети.

Для построения таблицы маршрутизации в протоколах OSPFv2 и OSPFv3 применяется модификация алгоритма Э.Дейкстры, позволяющая вычислять кратчайший путь между маршрутизаторами (узлами графа) в условиях неполной информации о структуре нагруженного графа. Стоимость дуг графа (метрика расстояния между маршрутизаторами по общему каналу) оценивается как динамически, так и задается в конфигурации. Каждое устройство строит путь до адреса назначения самостоятельно. При разделении сети на области граф OSPF (информация о его структуре)ограничивается одной областью. Разделение сети на области позволяет улучшить масштабируемость OSPF, иерархически объединяя области; оптимизировать использование вычислительных ресурсов маршрутизаторов, фиксируя размер известной части графа.

Пример включения и настройки OSPFv3 в разных операционных системах:

CISCO IOS

Для включения IPv6 протокола на маршрутизаторе CISCO необходимо ввести команду в режиме глобальной конфигурации:

*router(config)# ipv6 unicast-routing #Запуск процесса OSPFv3 производится в режиме глобальной конфигурации на маршрутизаторе:*

*router(config)#ipv6 router ospf 1#В режиме конфигурирования OSPFv3 нужно присвоить маршрутизатору идентификатор:*

*router(config-rtr)#router-id 1.1.1.1 #Для настройки в режиме конфигурирования интерфейса необходимы команды:*

*router(config-interface)#ipv6 address 2000:1:2:3::1/126 #(присвоение адреса ipv6 интерфейсу)*

*router(config-interface)#ipv6 ospf 1 area 0 #(запуск процесса OSPFv3 на интерфейсе)*

MikroTik (RouterOS)

Для включения пакета прошивки IPv6 необходимо ввести команду

*/system package enable [find name=ipv6 ] #Присвоение адреса IPv6 интерфейсу:*

*/ipv6 address add address=2003::1:0:0:0:1/64 advertise=no interface=ether2 #Присвоение идентификатора маршрутизатора:*

*/routing ospf-v3 set router-id=0.0.0.1 distribute-default=always-as-type-1 # Запуск процесса OSPFv3 на интерфейсе:*

*/routing ospf-v3 interfaceadd interface=ether2 area=backbone*

LINUX

*interface fxp0 # выбор интерфейса для запуска процесса ospf:*

*ipv6 ospf6 cost 1 # установка стоимости маршрута ОСПФ*

*ipv6 ospf6 hello-interval 10 #установка интервала между приветственными сообщениями*

*ipv6 ospf6 dead-interval 40 # интервал через который устройство вычеркивается из таблицы маршрутизации при не получении приветственного пакета*

*ipv6 ospf6 retransmit-interval 5 # интервал обмена таблицей маршрутизации*

*ipv6 ospf6 priority 0 # приоритет интерфейса*

*ipv6 ospf6 transmit-delay 1 #*

*ipv6 ospf6 instance-id 0 # номер процесса оspfv3*

*router ospf6*

*router-id 255.1.1.1 # id роутера, параметр обязателен в ospfv3*

*interface fxp0 area 0.0.0.0 # определение области*

FRRouting - программное обеспечение для реализации протоколов маршрутизации. На сегодняшний день данное программное ПО не полностью реализует протокол OSPFv3 для операционной системы Linux.

Таким образом, протокол OSPFv3 может работать на большинстве оборудования разных вендоров-производителей. Используемый алгоритм Дейкстры приводит к тому что каждый маршрутизатор области будет иметь в своей памяти граф, на котором будет работать алгоритм.

## Язык YANG

YANG- язык моделирования данных. В 2003 году были задокументированы требования операторов «удобство использования» как любой новой системы управления сетями. Среди требований были такие пункты как управление не одним устройством, а целыми сетями. Существование четкого разграничения между конфигурационной, рабочей и статической информацией устройства. Возможность разворачивания конфигурации, проверка правильности перед практическим задействованием и выполнение отката к предыдущей конфигурации в случае неудачи. Данные требования привели к созданию рабочей группы NETCONF. К 2010 году появились документы RFC 6020 и 6021 связанные с протоколом NETCONF и языком моделирования YANG. Модели YANG разрабатываются в области систем эксплуатации и управления, маршрутизации интернет, транспорт, безопасность.

Рабочие группы IETF, работающие над аспектами развития модели YANG, включают:

1. LIME (модели YANG OAM) модель для протоколов по состоянию канала. Функции эксплуатации, администрирования и поддержания связи важны для:

* Контроля состояния соединений
* Поиска неиправностей
* Контроля эффективности

1. L3SM (модель YANG для сервиса L3VPN) связь между операторами связи с доставкой на уровне L3 предоставляемые провайдером сервисы.
2. SUPA (модели YANG последовательной политики)
3. I2NSF (модели YANG, связанные с обеспечением безопасности)

Язык моделирования YANG является стандартом, определенным в IETF в рабочей группе NETMOD. YANG можно сказать, что древовидные, а не объектно-ориентированный. Данные конфигурации структурирована в дерево, и данные могут быть сложных типов, таких как списки и объединения. Определения содержатся в модулях и один модуль может увеличить дерево В другом модуле. Сильные правила пересмотра определены для модулей. YANG отображается в представлении NETCONF XML.

YANG также отличается от предыдущих моделей данных языков управления сетью тем что имеет сильную поддержку ограничений и правил проверки данных.

На рисунке (НОМЕР) представлено дерево, которое будет получено в результате написанного YANG файла

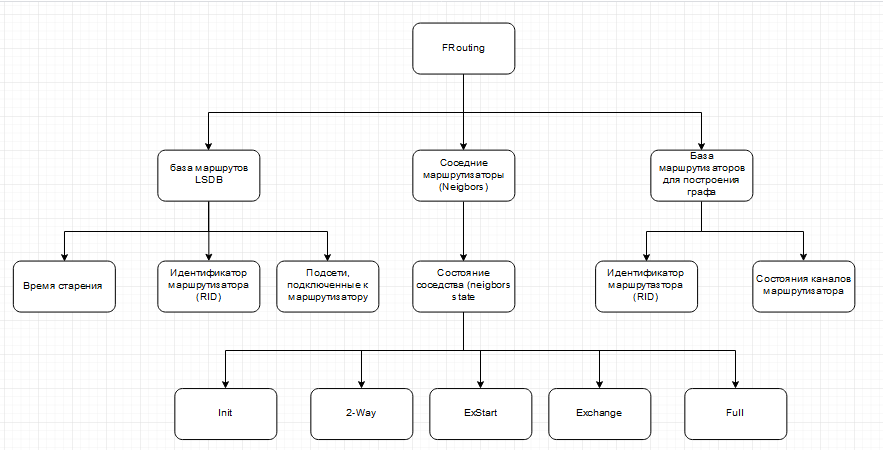


Рисунок-1. Структура модуля FRRouting во время работы протокола OSPFv3

В данной работе используется модуль для маршрутизации FRouting. Этот модуль состоит из следующих элементов:

1. LSDB база сетей для области OSPF. Она использует значения: RID отправившего маршрутную информацию маршрутизатора, подсети, которые к этому маршрутизатору подключены и таймер старения (время актуальности данной записи)
2. Список соседей. Это список смежных маршрутизаторов. Смежность маршрутизаторов определяется состоянием в котором они находятся. Нормальное состояние маршрутизаторов для соединений «точка-точка»- FULL, нормальное состояние для нешироковещательных сетей общего пользования FULL/DR или 2-WAY/Drother. (более подробно состояния маршрутизаторов будут описаны далее на странице НОМЕР)
3. База состояний всех маршрутизаторов области для построения графа и вычисления алгоритма Дейкстры для поиска кратчайшего пути. Основывается на сообщениях, полученных от маршрутизаторов области, содержащих информацию о состоянии каналов каждого маршрутизатора.

Таким образом, язык моделирования YANG является универсальным языком для представления структур данных разных типов, в том числе и сложных (таких как списки или объединения). На языке YANG удобно описывать структуру модуля FRRouting.

## Язык P4

P4 – язык выражения обработки кадров канального уровня программируемого элемента пересылки, например аппаратного или программного коммутатора, маршрутизатора или сетевой карты. Изначально язык задумывался как язык только для программирования коммутаторов, но область его применения была расширена для возможности охвата широкого спектра устройств.

Многие объекты реализуют не только управление канальным уровнем, и сетевым уровнем, но и плоскостью управления уровнями модели OSI. Язык P4 предназначен для указания только уровня управления для общения канального и сетевого уровня. P4 нельзя использовать для объектного программирования целевых объектов.

Коммутатор, программируемый на языке P4 отличается от традиционного коммутатора двумя существенными отличиями:

Функциональность канального уровня не фиксируется заранее и определяется программой P4. Канальный уровень конфигурируется во время инициализации для реализации функциональности, описанной программой P4 и не имеет встроенной информации о существующих сетевых протоколах. Это значит, что программа P4 будет создавать при прохождении трафика канальный уровень заново. Это позволяет изолировать ошибки, возникающие во время работы в ненадежной среде передачи данных. Если ошибка находится в программе P4 ошибка будет воспроизводиться каждый раз при запуске этой программы не зависимо от надежности среды.

Сетевой уровень связывается с канальным, используя те же методы, что и устройство со встроенными фунциями, но набор таблиц и других объектов канального уровня не фиксирован, поскольку канальный уровень определяется программой P4. Компилятор P4 генерирует API, который использует сетевой уровень для связи с канальным.

Следовательно, P4 можно назвать независимым от протокола и позволяющим программистам выражать богатый набор протоколов и поведений с канальным уровнем.

Основные абстракции, представляемые языком P4:

* Типы заголовков. Описывают формат (набор полей и их размеры) каждого заголовка в пакете.
* Парсеры. Описывают разрешенные последовательности заголовков в полученных пакетах.
* Таблицы. Связывают пользовательские ключи с действиями. Таблицы P4 обобщают традиционные таблицы переключателей, которые могут использоваться для реализации таблиц маршрутизации, таблиц поиска потоков, списков контроля доступа и других пользовательских типов таблиц, включая сложные решения с несколькими переменными.
* Действия. Фрагменты кода, которые описывают манипуляции с полями заголовка пакета и метаданными. Действия могут включать в себя данные, предоставляемые сетевым уровнем во время выполнения.
* Юниты. Выполняют последовательность операций:

1. Создают ключи поиска в полях пакета или вычисленных метаданных
2. Выполняют поиск в таблице, используя созданный ключ, выбирая действия для выполнения (включая связанные данные)
3. Выполнение выбранного действия.

* Контроль потока. Выражает императивную программу, описывающую цели обработки пакетов, включая зависящую от данных последовательность вызовов каждого одинакового действия. Депарсинг (повторная сборка пакета) также может выполняться с использованием сетевого уровня.
* Внешние объекты. Специфичны для архитектуры конструкциями, которые могут управляться программами P4 через четко определенные API, но внутреннее поведение жестко связано (например, единицы контрольной суммы) и, следовательно, не программируется с использованием P4.
* Пользовательские метаданные. Определяются пользователем структуры данных, связанны с каждым пакетом.
* Внутренние метаданные. Метаданные, предоставляемые архитектурой, связаны с каждым пакетом - например, порт ввода, где был получен пакет.

Производители обеспечивают и предоставляют аппаратную или программную среду реализации, определением архитектуры занимается и компилятор P4. Программисты P4 пишут программы для конкретной архитектуры, которая определяет набор программируемых P4 целевых компонентов, а также их внешние интерфейсы канального уровня. А значит все что необходимо знать программисту об устройстве, на котором будет работать программа P4 – это задача, для которой предназначена программа P4.

Компиляция программ P4 порождает:

* Конфигурацию канального уровня, реализующую логику пересылки, описанную в программе ввода.
* API для управления состоянием объектов канального из сетевого.

P4 - это предметно-ориентированный язык, который предназначен для реализации самых разных целей, включая программируемые сетевые карты, FPGA (представляют из себя микросхему с большим количеством логических блоков соединенных в одну цепь. Конфигурируется в любое время под любую из двоичных операций. Состоит из программируемых логических блоков, содержащих таблицу LUT, четыре входа и триггер; блоков ввода и вывода для связи констант с сигнальными линиями; внутренних линий , которые занимаются управлением путей соединения блоков ввода и вывода с программируемыми логическими блоками), программные и аппаратные коммутаторы ASIC (микросхемы, которые могут очень быстро делать ограниченное число операций по перекладыванию кадра из одного буфера в другой в соответствии с заданным алгоритмом. Имеют интерфейсы: Ethernet, интерфейс процессора управления, высокоскоростной интерфейс). Язык может быть эффективен в конструкциях, которые могут быть эффективно реализованы на всех этих платформах.

Предполагается фиксированная стоимость операций поиска в таблице и взаимодействия с внешними объектами. Все программы P4 (анализаторы и элементы управления) выполняют постоянное число операций для каждого байта входного пакета, полученного и проанализированного. Хотя синтаксические анализаторы могут содержать циклы, учитывается что заголовок извлекается в каждом цикле, а сам пакет имеет ограничение на общее выполнение синтаксического анализатора. Другими словами, при этих допущениях вычислительная сложность программы P4 является линейной к общему размеру всех заголовков и никогда не зависит от количества состояний, во время обработки данных (например, количества потоков или общего количества обработанных пакетов). Эти необходимо (но не достаточно) для обеспечения быстрой обработки пакетов.

P4 обеспечивается и определяется следующим образом: если конкретная цель T поддерживает только подмножество языка программирования P4, P4T программы написанные на P4T. Выполнение программ должно обеспечивать поведение, описанное в документации. Следует обратить внимание, что P4-совместимые цели предоставляют произвольные расширения языка P4 и внешние элементы.

**Преимущества P4** по сравнению с современными системами обработки пакетов (например, основанными на записи микрокода на пользовательском оборудовании), P4 обеспечивает ряд существенных преимуществ:

* Гибкость. P4 занимается пересылкой пакетов выраженной в виде программы, в отличие от традиционных коммутаторов, предоставляющие механизмы пересылки с фиксированными функциями.
* Выразительность. P4 выражает сложные, аппаратно-независимые алгоритмы обработки пакетов, используя операции общего назначения и поиск таблиц. Такие программы совместимы между устройствами, использующими одну и ту же архитектуру (при условии наличия достаточных ресурсов).
* Управление ресурсами. Программы P4 описывают ресурсы абстрактно (например, адрес источника IPv4); компиляторы сопоставляют эти пользовательские поля с доступными аппаратными ресурсами и управляют деталями низкого уровня, такими как распределение и планирование.
* Разработка программного обеспечения. Программы P4 имеют преимущества, такие как проверка типов, скрытие информации и повторное использование программного обеспечения.
* Библиотеки компонентов: Библиотеки компонентов, поставляемые производителями, используются для объединения аппаратных функций в переносимые высокоуровневые конструкции P4.
* Разделение изменений аппаратного и программного обеспечения: Производители могут использовать абстрактные архитектуры для дальнейшего разделения низкогоуровневого программирования от высокоуровневой разработки.
* Отладка. Производители могут предоставить программную модель архитектурыс целью помощи в разработке и отладке программ P4.

*Эволюция языка P4: сравнение с предыдущими версиями (P4 v1.0 / v1.1)*

По сравнению с P414, более ранними версиями языка, P416 вносит ряд существенных, обратно несовместимых изменений в синтаксис и семантику языка. Большое количество языковых функций были исключены из языка и перенесены в библиотеки, включая счетчики, единицы контрольной суммы, счетчики и т.д.

Следовательно, язык был преобразован из сложного языка (более 70 зарезервированных ключевых слов) в относительно простой базовый язык (менее 40 зарезервированных ключевых слов) с библиотекой основных конструкций, которые необходимы для написания большей части программ P4.

Версия P4 v1.1 представила языковую конструкцию под названием extern , которая может быть использована для описания элементов библиотеки. Таким образом, конструкции, определенные в спецификации языка v1.1, будут преобразованы в элементы библиотеки (включая конструкции, исключенные из языка, такие как измерители и счетчики). Extern ожидает, что объекты будут стандартизированы и включены в новый документ, описывающий стандартную библиотеку элементов P4.

Одна важная цель P416 это пересмотр языка, для обеспечения стабильного определения языка. Другими словами, разработчики стремятся к тому, чтобы все программы, написанные на P416 оставались синтаксически правильными и вели себя одинаково при работе программ в следующих версиях языка. Если следующая версия языка потребует отказа от обратной совместимости разработчики постараются предоставить легкий путь для миграции P416 программы в новую версию.

Таким образом, язык P4 предназначен для описания сетевых протоколов. Программы на этом языке не зависят от уровня передачи данных. Компилятор P4 подходит для написания архитектур под разные архитектуры процессоров. Этот язык является удобным средством для программирования протокола динамической маршрутизации в сетях IPv6.

## Централизованное управление Ansible

В крупных сетях необходима организация централизованного управления. Централизованное управление облегчает задачу системного администратора и позволяет исключить ошибки человеческого фактора. Централизованное управление в территориально распределенных сетях требует надежных каналов связи и требования к отказоустойчивости электороэнергии.

К централизованной системе управления должны решать следующие задачи:

* Поддержка различных версий операционных систем;
* Управление производительностью серверов при росте или сокращении количества пользователей;
* Работа в неоднородных сетях (разные устройства, разные операционные системы);
* Быстрая установка критических обновлений при обраружении уязвимостей 0-day (уязвимости, информация о которых опубликована раньше, чем выпущено обновление, перекрывающее данную уязвимость);
* Управление доступом пользователей к ресурсам сети.

Ansible — это программное решение для удаленного управления конфигурациями. Оно позволяет настраивать удаленные машины. Главное его отличие от других подобных систем в том, что Ansible использует существующую инфраструктуру SSH. Это делает Ansible более универсальным средством управления, в отличии, например, от Chef или Puppet.

**Список управляемых хостов, Inventory.** Ansible позволяет хранить и группировать список управляемых хостов. Так же хосты можно группировать по «похожести». (Например разделить роутеры вендора Cisco от роутеров вендора Mikrotik. Управляющие команды CLI для них будут разными). Список хостов можно задавать как по IP адресу, так и по имени. Пример оформления списка хостов приведен в листинге (НОМЕР)

[Mikrotik]

host0.example.org

host1.example.org

host2.example.org

[Cisco]

host0.example2.org

host1.example2.org

host2.example2.org

**Выполнение команд, Playbook.** Playbook- это выполнение команды, только если есть что выполнять. Так, например плейбук перезапуска веб-сервера будет выполнен на конечном хосте, только если на нем есть служба вебсервера. И запуск данного плейбука оставит хост в прежнем состоянии если на нем не установлено службы веб-сервера. Это позволяет избегать оишбок заполнения Inventory и попадания туда хостов, на которых данную команду выполнять не нужно. Ansible позволяет отслеживать успешность выполнения плейбука, и в случае возникновения ошибки либо делать очередную попытку запуска, либо откатывать в состояние до выполнения данного плейбука.

**Роли Ansible.** Роли имеют следующую структуру:

roles

|

|\_some\_role

|

|\_files

| |

| |\_file1

| |\_...

|

|\_templates

| |

| |\_template1.j2

| |\_...

|

|\_tasks

| |

| |\_main.yml

| |\_some\_other\_file.yml

| |\_ ...

|

|\_handlers

| |

| |\_main.yml

| |\_some\_other\_file.yml

| |\_ ...

|

|\_vars

| |

| |\_main.yml

| |\_some\_other\_file.yml

| |\_ ...

|

|\_meta

|

|\_main.yml

|\_some\_other\_file.yml

|\_ ...

Файл main.yml является обязательным. В нем лежат наши плейбуки. Данный подход позволяет одновременно проводить проверку наличия того, что выполнять на хосте, выполнять, проверять успешность и в случае неуспеха откатывать обратно сделанные изменения.

Таким образом, подход Ansible заключается в том, что нам нужно все устройства привести в состояние, описанное в файле (идеальная конфигурация). Ansible позволяет сравнивать текущие конфигурации устройств с идеальной и применять к ним команды для приведения их в состояние идеальной конфигурации. Это удобно с точки зрения администрирования большого парка устройств и распределенной инфраструктуры.

## 1.7. Язык YAML

YAML Ain't Markup Language («YAML — не язык разметки»)- язык описания предметной области, близкий по своей сути к языкам разметки, но не являющийся таковым. Ориентирован на удобство ввода-вывода типичных структур данных многих языков программирования.

Целями данного языка являются:

* быть легко понятным человеку;
* поддерживать структуры данных, родные для языков программирования;
* быть переносимым между языками программирования;
* использовать цельную модель данных для поддержки обычного инструментария;
* поддерживать потоковую обработку;
* быть выразительным и расширяемым;
* быть лёгким в реализации и использовании;

Отличием данного языка от языков разметки является максимально гибкое описание данных. Он позволяет наиболее полно отображать основные типы данных, используемых современными языками программирования, сохраняя при этом максимальную простоту как для человека, так и для программного анализа.

YAML может использоваться как язык написания конфигурации. Именно он используется как язык конфигураций в системе Ansible. YAML имеет множество различных модулей. Основным его недостатком является неудобство использования с любыми кодировками, отличными от Unicode. Отличные от Unicode кодировки сегодня использовать нецелесообразно.

YAML начичная с версии 1.2 – это надмножество JSON (т.е. все что справедливо для JSON справедливо и для YAML). JSON – это простой текстовый формат обмена структурированными даными, основанный на JavaScript. Во многих ситуациях JSON рассматривается как альтернатива XML, но в этих ситуациях JSON предоставляет более протой и компактный формат представления данных.

Таким образом, языки YAML и JSON являются удобными для передачи данных и пригодня для использования при написании конфигураций для оборудования.

# Глава 2 реализация протокола динамической маршрутизаци

IPv6 адреса имеют длину 128 бит и сложны для запоминания и записи. В связи с этим маршрутзация должна быть полностью динамической во избежания ошибок в указании сетей и подсетей. В связи с «громоздкостью» IPv6 адресов статическая маршрутизация при работе с ними не целесообразна.

Протоколы маршрутизации OSPFv3 и OSPFv2 решают задачу управления сетевым трафиком за несколько этапов: а) установление соседских отношений (adjacency) с маршрутизаторами (узлами), б) распространение информации о структуре сети, в) построение оптимальных маршрутов на графе, г) внесение изменений в таблицы маршрутизации узлов сети.

## Моделирование протокола OSPFv3

В процессе установки соседских отношений маршрутизаторы обмениваются пакетами hello. Маршрутизаторы считаются установившими соседские отношения (adjacency) когда они закончили синхронизацию LSDB. LSDB (link state database) это список всех записей о состоянии каналов ( учитывается не только состояние канала link up или link down, но и стоимость и пропускная способность каждого из каналов)

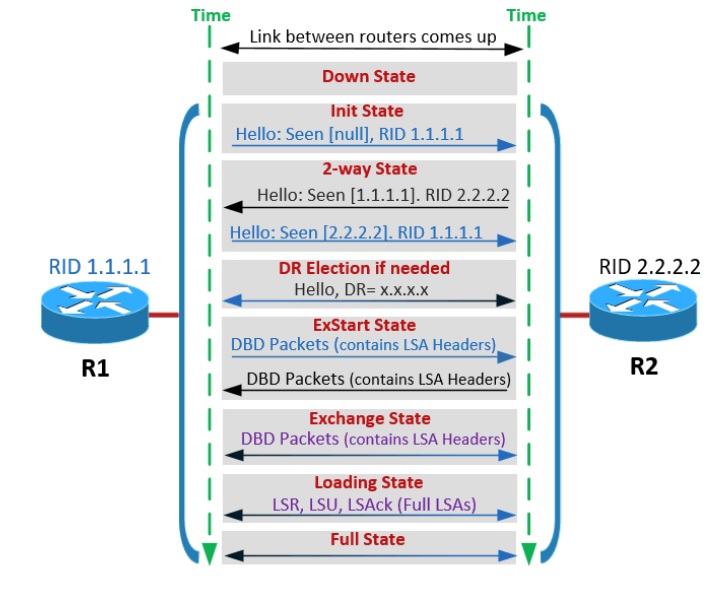
Для синхронизации базы LSDB между маршрутизаторами используются пакеты DBD. Для уменьшения количества трафика в сети DBD, содержащие полную базу LSDB рассылаются только маршрутизатором DR в нешироковещательных сетях общего доступа и только для новых устройств в сети. DBD пакеты между маршрутизаторами, сфомировавшими соседские отношения пересылаются только в случае изменения топологии. В этом случае в пакетах DBD содержится только отличающаяся информация. Например, если маршрутизатор, через который был маршрут в сеть 192.268.0.0/24 перестал отвечать на пакеты hello в течении HelloInterval и RouterDeadInterval в пакете DBD будет содержаться информация о недостижимости маршрута в сеть 192.168.0.0/24.

В нешироковещательных сетях общего доступа DBD пакеты могут пересылать все маршрутизаторы на мультикастовый адрес 224.0.0.5 (это адрес маршрутизатора DR) и 224.0.0.6 (адрес маршрутизатора BDR). Зачачей DR в данном случае будет являться разослать DBD всем маршрутизаторам сети для синхронизации LSDB. Каждый маршрутизатор, получивший DBD строит на основе полученных DBD пакетов свою таблицу маршрутизации.

Таким образом, для работы протокола OSPFv3 маршрутизаторы сначала должны составить граф, в котором будет работать алгоритм Дейкстры, затем произвести обмен маршрутами. Если на маршрутизатор подключен к OSPFv3, но не анонсирует ни одной подсети, он будет в списке маршрутизаторов, принимающих участие в построении графа.Так же этот маршрутизатор сможет построить свой собственный граф для дальнейшего поиска кратчайшего пути.

## Состояния маршрутизаторов в процессе установки соседских отношений OSPFv3

В процессе установления соседства OSPF маршрутизаторам необходимо пройти следующие состояния (рис. 1):



*Down*. Это начальное состояние OSPF-соседа. Для обнаружения соседей в этом состоянии рассылаются специальные hello-пакеты, сообщающие, что с данной стороны канала существует OSPF-маршрутизатор. Данное состояние означает, что от соседей ни было получено никакой информации (hello-пакетов). В состоянии *Full* (см. далее), если роутер не принимает hello-пакеты в течение времени RouterDeadInterval или если сосед был удален из конфигурации, тогда состояние соседа изменяется на *Down*.

*Init.* Это состояние указывает, что маршрутизатор получил hello-пакет от своего соседа, но идентификатор RouterID соседнего маршрутизатора не был включен в этот hello-пакет, т.е. сосед не знал еще о существовании другого (принимающего) соседа. В ответ на первое сообщение посылается ответный hеllo-пакет соседу, но при этом теперь добавляется идентификатор соседа.

*2-Way.* Это состояние означает, что между маршрутизаторами установлено двунаправленная связь, т.е. что соседи получили hello-пакеты друг друга с соответствующими идентификаторами. Такое состояние устанавливается, когда узел, принимающий hello-пакет, видит свой собственный RouterID в приходящем hello-пакете. В этом состоянии маршрутизатор решает, будет ли через этот узел маршрутизироваться трафик. В широковещательной (broadcast) среде или в среде точка-мультиточка (point-to-multipoint), маршрутизатор переходит в состояние *Full* только с основным роутером (Designated Router, DR) и с резервным (Backup Designated router, BDR). Со всеми другими соседями он остается в состоянии *2-Way*. Если в сети нет DR- или BDR-узлов, то запускается процедура выбора DR и BDR среди соседних узлов.

*Exstart.* Как только выбраны DR и BDR, между маршрутизаторами и их DR и BDR начинается процесс обмена информацией о состоянии каналов. В этом состоянии, узлы и их DR и BDR устанавливают отношения ведущий-ведомый, создают соединение как специальным образом пронумерованную последовательность пакетов. Маршрутизатор с более высоким RouterID становиться ведущим, начинает обмен данными по соединению, увеличивая каждый раз номер пакета в последовательности.

*Exchange.* В этом состоянии, OSPF-маршрутизаторы обмениваются DBD-пакетами (дескрипторами базы данных). BDB содержат только заголовки LSA (объявления о состоянии канала) и описывают содержимое всей базы данных о состоянии каналов. Каждый DBD-пакет пронумерован, и считается переданным по соединению, если получен пакет-ответ. Узлы также посылают пакеты–запросы и пакеты–обновления о состоянии канала. Содержимое принимаемого DBD-пакета обновляет информацию в базе данных состояния каналов узла.

*Loading.* В этом состоянии происходит непосредственно обмен информацией о состоянии канала. Основываясь на информации полученной при помощи DBD-пакетов, маршрутизаторы посылают пакеты-запросы о состоянии канала. Затем сосед предоставляет запрошенную информацию в пакетах-обновлениях.

*Full.* В этом состоянии, маршрутизаторы являются полностью связанными друг с другом. Узлы обменялись сетевыми и маршрутными LSA, и их базы данных полностью синхронизированы. В этом состоянии модель (топология) сети известнавсем маршрутизаторам.

Состояние Fullявляется нормальным состоянием для OSPF-роутера. Если роутер не переходит в это состояние за приемлемое время, то это указывает на проблему в формировании связности. Исключением из этого является состояние 2-Way, которое является обычным для широковещательных сетей. Роутеры достигают состояния Full только со своими DR и BDR. С остальными соседями состояние всегда остается 2-Way.

Кроме перечисленных выше основных состояний для вручную настроенных соседях в NBMA-среде (нешироковещательная сеть множественного доступа - non-broadcast multiple access network) задается специальное состояние *Attempt.* В состоянии Attempt, маршрутизатор посылает юникастовые hello-пакеты через определенный интервал опроса соседу, от которого не были получены hello-пакеты в течение Dead-интервала.

Таким образом, состояния, описанные в данном разделе должен пройти каждый маршрутизатор, на котором запущен процесс OSPF. Нормальными состояниями для запущенного процесса могут являться состояния Full для канала, связывающего с DR и BDR и состояние 2-Way для канала с маршрутизаторами Drother.

## Обмен маршрутами OSPFv3

При динамической маршрутизации происходит обмен маршрутной информацией между соседними маршрутизаторами, в ходе которого они сообщают друг другу, какие сети в данный момент доступны через них. Информация обрабатывается и помещается в таблицу маршрутизации.

Устройства, подключенные к одному каналу и участвующие в процессе обмена информацией протокола OSPF называются соседними маршрутизаторами. Для обнаружения OSPF-устройств маршрутизаторы рассылают многоадресатные Hello-пакеты через все интерфейсы, на которых настроен протокол OSPF. В запросе содержится следующая информация:

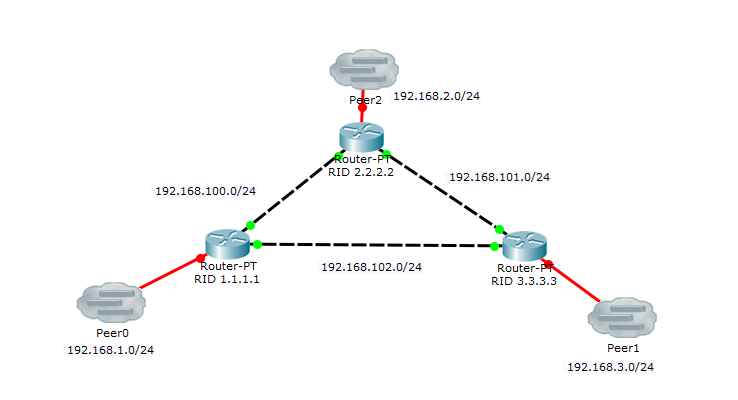
* идентификатор маршрутизатора-отправителя Router ID – RID,
* идентификатор зоны OSPF Area ID,
* Hello-интервал,
* интервал обнаружения неработоспособности устройства (dead interval),
* приоритет маршрутизатора (router priority),
* идентификатор RID выделенного маршрутизатора (designated router DR),
* идентификатор RID резервного выделенного маршрутизатора (backup designated router BDR)
* список соседних устройств, обнаруженных маршрутизатором-отправителем.

Каждому маршрутизатору присваивается уникальный номер – идентификатор маршрутизатора RID. Он представляет собой 32-битное число, форма записи похожа на адрес IPv4. В сетях IPv4 настройка RID не является обязательной, в случае если не настроен RID в качестве него будет использоваться наибольший IP адрес на Loopback интерфейсе, в случае если не настроен ни один Loopback интерфейс будет использоваться наибольший IPv4 адрес, настроенный на любом интерфейсе маршрутизатора. В OSPFv3 настройка RID является обязательным условием работы протокола.

Обмен маршрутами происходит следующим образом:

Маршрутизаторы ведут обмен информацией о состоянии каналов, тем самым строят граф, для работы алгоритма Дейкстры. После того как алгоритм сошелся, маршрутизаторы начинают строить маршруты согласно LSDB базе, которая является общей для всех маршрутизаторов.

Например, в сети нарисованной на рисунке (НОМЕР)маршрутизаторы после установления соседских отношений (перехода в состояние adjacency) роутеры будут обмениваться пакетами Hello и LSA. Hello пакеты необходимы для поддержания состояния соседства, в LSA содержится информация о состоянии каналов каждого роутера.



Например маршрутизатор с идентификатором 2.2.2.2 будет рассылать приветственные сообщения маршрутизаторам с идентификаторами 1.1.1.1 и 2.2.2.2. Получая такие сообщения смежные маршрутизаторы с идентификаторами 1.1.1.1 и 2.2.2.2 будут проверять настройки области, номера процесса OSPF, если настроена аутентификация проверять подлинность смежного роутера.

После проверки, если настройки совпадают, маршрутизаторы могут обмениваться анонсами состояния каналов (Link-State Advertisements – LSA).

После установления двустороннего канала маршрутизаторы продолжают периодически обмениваться Hello-сообщениями. Если связь отсутствуют в течение времени, которое определяется dead-интервалом, то считается, что связь с соседним устройством потеряна. Стандартно в протоколе OSPF интервал рассылки Hello-сообщений равен 10 с, dead-интервал – 40 с.

В анонсах LSA содержится подробная информация о топологии сети. Процесс рассылки этих анонсов называется лавинной рассылкой (flooding), при которой маршрутизаторы пересылают анонсы LSA своим соседям, которые, в свою очередь, рассылают их своим соседям, и так до тех пор, пока все устройства в сети не получат информацию из анонса. Анонсы LSA рассылаются периодически (по умолчанию один раз в 30 мин). По окончании процесса рассылки у всех маршрутизаторов в домене маршрутизации появится общая одинаковая информация о сети. Информация хранится в виде структуры, называемой базой данных состояния каналов link-state database – LSDB.

После того как маршрутизаторами будет построен граф, маршрутизаторы будут обмениваться данными базы LSDB (DBD пакеты). DBD база для всех маршрутизаторов одинакова. В ней содержатся идентификаторы маршрутизаторов одной области, а так же маршруты, которые эти маршрутизаторы анонсируют (сети, которые доступны этим маршрутизаторам). Пример LSDB базы приведен в таблице (НОМЕР).

|  |  |
| --- | --- |
| Router RID | Network |
| 3.3.3.3 | 192.168.3.0 |
| 3.3.3.3 | 192.168.101.0 |
| 3.3.3.3 | 192.168.102.0 |
| 2.2.2.2 | 192.168.2.0 |
| 2.2.2.2 | 192.168.100.0 |
| 2.2.2.2 | 192.168.102.0 |
| 1.1.1.1 | 192.168.1.0 |
| 1.1.1.1. | 192.168.100.0 |
| 1.1.1.1 | 192.168.102.0 |

Когда у каждого маршрутизатора в домене маршрутизации есть идентичная копия базы LSDB, то используется технология протоколов маршрутизации с учетом состояния каналов. Устанавливаются маршруты в таблицу IP-маршрутизации: создаются записи, содержащие адрес подсети, маску, выходной интерфейс и адрес следующего транзитного устройства (next-hop). Для выполнения данной задачи используется алгоритм поиска первого кратчайшего пути Дейкстры.

Протокол OSPF выбирает маршрут между маршрутизатором и какой-либо сетью с наименьшей стоимостью. С каждым интерфейсом на маршруте связано некоторое значение стоимости. Стоимость всех интерфейсов (каналов), через которые пролегает путь к сети, суммируется и выбирается путь, стоимость которого минимальна. Таким образом, каждый маршрутизатор строит маршруты подобно древовидной структуре, в корне которой ставит себя.

Таким образом, передача маршрутов является основной задачей для процесса динамической маршрутизации. Передача маршрутов и маршрутная информация функционируют одинакво для маршрутизации OSPFv2 и OSPFv3.

## Реализация маршрутизации OSPFv3

Декларативное программирование подразумевает описание программистом какую именно задачу нужно решать. В основе декларативных языков лежит формализованная человеческая логика. Поиском решения согласно описанию решаемой задачи занимается система программирования. Такой подход к программированию может применяться в областях:

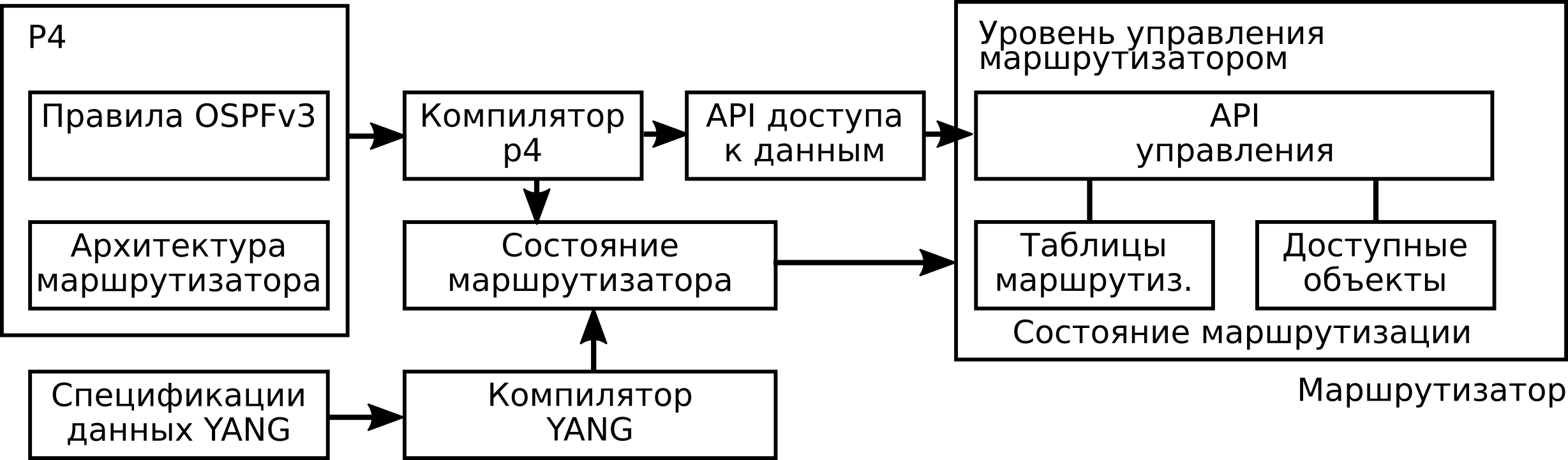
* создание систем искусственного интеллекта;
* автоматическое доказательство теорем;
* разработка экспертных систем и оболочек экспертных систем;
* создание систем поддержки принятия решений;
* разработка систем обработки естественного языка;
* построение планов действий роботов.

Основным преимуществом декларативного программирования- это сокращение количества кода, создание более понятных программ (по сравнению с программами, написанными на императивных языках программирования). Подход декларативного программирования проще и прозрачнее формализуется математически, что облегчает тестирование и верифицирование написанных программ.

Для реализации протокола OSPFv3 предлагается использовать декларативные спецификации языков P4- это язык высокого уровня, не зависящий от протокола канального уровня. Предназначен для реализации протоколов сетевого уровня не зависимо от среды передачи данных, а так же производителя оборудования. Так как язык P4 не зависит от протоколов канального уровня он подходит для программирования протоколов маршрутизации в виртуальной среде вне зависимости от средств, которые используются виртуальной средой. Язык P4 предоставляет API для подключения устройств, реализующий нижние уровни сетевой модели OSI.

Язык P4 предлагает API для доступа к данным, но не является удобным средством для описания структур и таблиц, которые необходимы для реализации протоколов маршрутизации. Для этих целей был выбран язык структур данных YANG.

Предлагаемая реализация протокола базируется на использовании декларативных спецификаций структур данных YANG, опубликованных в Интернет, а также реализации динамической модели протокола при помощи языка P4. На рисунке 1 представлена технология синтеза модулей, реализующих протокол OSPFv3 в предлагаемом подходе.



**Рис. 1. Синтез структур данных логического уровня управления маршрутизатором**

Ключевым блоком в схеме является “Состояние маршрутизатора” - набор объектов, представляющих состояние узла как элемента структуры сети (высокий уровень представления). Состояние маршрутизатора интерпретируется (загружается) в виде записей таблицы маршрутизации и состояния доступных (external) объектов маршрутизатора (низкий уровень представления). Интерпретация осуществляется при помощи запросов к API уровня управления маршрутизатором.

Структуры данных “Состояния маршрутизатора” формируются из спецификаций YANG и структур данных “Правил OSPFv3”: модель динамики маршрутизатора задается в виде правил “Если … то …”, где с левой стороны правила задается условие на свойства входного пакета, а справа - некоторое действие, например, переход в новое состояние. Для формирования “Копилятором p4” “API доступа к данным” необходимо задать на языке p4 “Архитектуру маршрутизатора”. Алгоритм и API SPF реализуются на языке C и в схеме находятся в блоке “Уровень управления маршрутизатором”. Реализация алгоритма создается на основные функции модуля ospf6d пакета FRRouting, обеспечив взаимодействие с p4 и структурами YANG.

Приведем пример спецификации IETF структуры - “состояние маршрутизатора” в процессе установки соседских отношений.

|  |  |
| --- | --- |
| grouping neighbor-state {  description  "OSPF neighbor state.";  leaf address {  type inet:ip-address;  config false;  description "Neig. IP";  } | leaf dr-router-id {  type rt-types:router-id;  config false; }  leaf dr-ip-addr {  type inet:ip-address;  config false;  description "Neig's DR IP.";  } … } |

Спецификация P4 должна включать архитектуру маршрутизатора (перечень физических и виртуальных портов), API уровня управления, процедуры трансляции/генерации заголовков пакетов и набор правил обработки пакетов согласно их заголовкам. Приведем пример правил для нашей задачи (типы данных опущены).

|  |  |
| --- | --- |
| control neighbor\_state(…h, …inp, …err, …out, …rou) {  table down\_hello {  key = {h.ospf.nei.id==none;  rou.st == DOWN;}  actions = {set\_hello;}}  table init\_hello {  key = {h.ospf.nei.id!=none;  rou.st == DOWN}  actions = {set\_hello;  set\_nrID;}} | action set\_hello{  out.ospf.my.id =  API.get\_my\_id();}  action set\_nrID {  out.ospf.nei.id=rou.nei.id;  rou.st = INIT;}  apply {  down\_hello.apply();  init\_hello.apply();  … } … } |

Аналогичным образом на P4 задается весь перечень функций маршрутизатора. В примере переменная API задает интерфейс к системным функциям узла, в частности, функциям вычисления контрольных сумм данных, доступа к тексту конфигурации, а также алгоритму вычисления кратчайших путей.

Таким образом, в докладе представлены основные идеи подхода к реализации этапа установления соседских отношений в проектировании программного обеспечения ДМ на основе протокола OSPFv3 для сетей IPv6; показаны примеры спецификаций структур данных и части автомата, моделирующего состояние маршрутизатора и его соседей, а также архитектура процесса проектирования средств ДМ. Следующей задачей, решаемой в данном проекте, является адаптация компилятора P4 к API алгоритмов вычисления кратчайших расстояний Э.Дейкстры.

## Реализация централизованного управления в сети

При помощи Ansible было реализовано распространение следующиъ конфигураций:

* Распространение информации о серверах точного времени на устройства
* Применение единого часового пояса для всех устройств сети
* Создание бекапа и скрипта, восстанавливающего из бекапа устройство через 20 минут (если доступ к устройству не пропадает- удаляется скрипт восстановления из бекапа)
* Настройка отправки логов на syslog сервер
* Скрипт обновления ПО на устройстве

Сервера точного времени (NTP) предназначены для синхронизации часов устройства с серверами точного времени в интернете. Особенность протокола заключается в том, что при синхронизации учитывается время, затраченное на передачу информации от сервера до клиента. Погрешность при синхронизации с удаленными серверами, например Stratum2 состовляет около 0,10 мс. В большинстве случаев этого бывает достаточно.

Применение единого часового пояса на всех устройствах, расположенных в этом часовом поясе облегчает задачу администрирования. Так как любой администратор может вычислить какое время должно быть на устройстве сейчас, и какое время было несколько часов назад. Необходимо это для поиска неполадок. Логи на устройствах имеют привязку ко времени, установленном на устройстве.

Скрипт на языке RouterOS для Ansible синхронизирующий устройство с серверами ntp.org и изменяющий часовой пояс на +8 (Asia/Irkutsk) приведен ниже:

*/system ntp client set enabled=yes primary-ntp=[:resolve ntp1.stratum2.ru] secondary-ntp=[:resolve ntp2.stratum2.ru]*

*/system clock set time-zone-name=Asia/Irkutsk time-zone-autodetect=no*

*Создание* точки возврата (rollback) практически всегда является обязательным требованием для работы с сетями, которые не являются тестовыми. Использование таких точек возврата является хорошим тоном даже после предварительного тестирования в среде, очень близкой к реальной.

Ниже приведен скрипт на языке RouterOS для создания точки возврата и восстановлении из нее через 20 минут:

*:local time1 value=[/system clock get time]; local time2 value=($time1+20m); :local date value=[system clock get date] ; /system scheduler add name=rollback on-event="/system backup load name=flash/backup.backup password=\"\" " policy=ftp,reboot,read,write,policy,test,password,sniff,sensitive,romon start-date=$date start-time=$time2*

И удаление данного скрипта если применение новых конфигураций не привело к потере управления

*/system scheduler remove [find name=rollback]*

Системы мониторинга необходимы для скорейшего обнаружения неполадок в сети. Одним из способов мониторинга является отправка логов устройство на удаленный сервер syslog (журнал системных сообщений). Такой мониторинг прост в настройке, не требует «обратной связи» от сервера, принимающего сообщения. На сервере, на котором просходит сбор логов устройств так же может быть настроена система оповещений, при получении (или не получении) записей с определенным содержанием.

Пример скрипта на языке RouterOS для Ansible приведен ниже

*/system logging action set 3 bsd-syslog=yes remote=syslog.local syslog-facility=syslog syslog-severity=info; :do { :local addrtun [/ip address get [find interface=tunnel ] address ] ; /system logging action add bsd-syslog=yes name=remoteerror remote=syslog.local syslog-facility=syslog syslog-severity=error target=remote; /system logging action add bsd-syslog=yes name=remotecritical remote=syslog.local syslog-facility=syslog syslog-severity=critical target=remote; /system logging action add bsd-syslog=yes name=remotewarning remote=syslog.local syslog-facility=syslog syslog-severity=warning target=remote; /system logging action add bsd-syslog=yes name=remoteinfo remote=syslog.local syslog-facility=syslog syslog-severity=info target=remote; /system logging add action=remoteinfo topics=info prefix=$addrtun; /system logging add action=remotecritical topics=critical prefix=$addrtun; /system logging add action=remoteerror topics=error prefix=$addrtun; /system logging add action=remotewarning topics=warning prefix=$addrtun; /system logging remove [find invalid ]; :log info "It Works! Hello syslog!"} on-error={:log info "exists"};*

Обновление ПО на устройствах необходимо. Обновления могут быть различными: приносящие новый функционал, исправляющие старые ошибки ПО, обновления безопасности. Самыми важными для сетей организаций являются обновления безопасности. Они исправляют уязвимости ПО, которые могут быть использованы вредоносными программами. Такие уязвимости называются 0day (уязвимости, информация о которых была опубликована раньше, чем выпущено обновление ПО, исправляющее данную уязвимость) и устанавливать обновление ПО целесообразно сразу после выхода обновления с исправлением.

Пример скрипта, производящего обновление устройства до последней актуальной версии на языке RouterOS для Ansible приведен ниже:

*/system package update install*

*/system routerboard upgrade*

y

Таким образом, реализация централизованного управления при помощи Ansible позволяет экономить время, когда необходимо применить похожую конфигурацию на различные устройства. Так же Ansible позволяет отслеживать выполнение команд на удаленных устройствах и исправлять ошибки. Так же можно создавать точки возврата на устройствах если во время изменения конфигурации будет потеряно управление устройством.

## Тестирование работы протокола

Для тестирования работы протокола будет использована виртуальная среда, которая может использоваться в организации, имеющей 4 расположенных вблизи друг от друга филиала. Пример тестовой сети приведен на рисунке (НОМЕР)

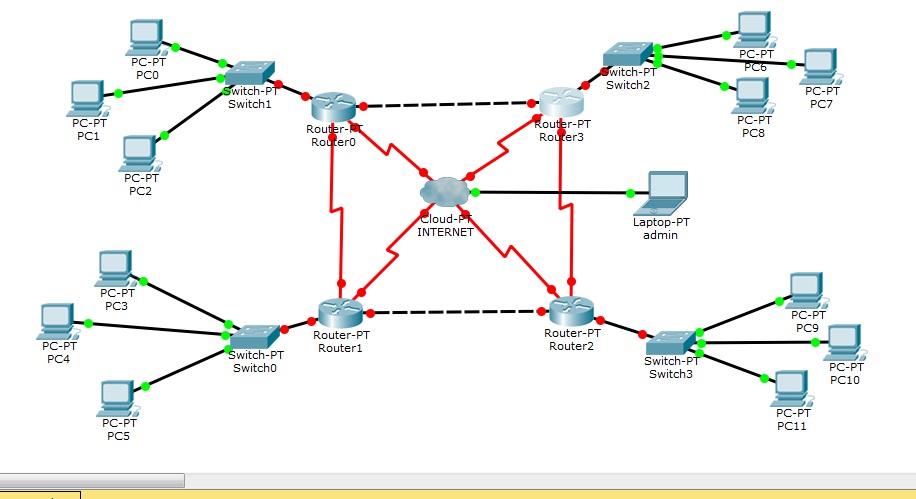


Рис. …. Структура тестовой сети

В данной сети необходим протокол динамической маршрутизации для резервирования маршрутов между филиалами. Так как OSPF может использоваться для балансировки трафика, при неодинаковой пропускной способности каналов он был выбран для реализации. В качестве маршрутизаторов можно использовать компьютеры с установленной операционной системой ArchLinux и установленным ПО FRRouting. FRRouting позволяет организовать динамическую маршрутизацию.

Для установки нового модуля написанного на языке P4 применяется Ansible.

Протокол, реализованный на языке P4 показал хорошую работу в данной тестовой среде.

# Заключение

IPv6 – это новая версия протокола интернета (IP), появившаяся на замену устаревшему протоколу IPv4, разработанный IETF к 1996 году, когда возникла проблема нехватки адресов IPv4. В 2018 году трафик IPv6 составило всего 25% от общего сетевого трафика. На сегодняшний день оборудование, оборудование, которое можно встретить по миру обычно поддерживает работу в dual stack (оба протокола IPv4 иIPv6 используются параллельно).

Маршрутизация служит для приема пакетов от одного устройства и для передачи его по сети через другие сети. Для осуществления маршрутизации маршрутизатор должен обладать следующей информацией: адрес назначения, маршрутизатор через который есть доступ к сети адреса назначения, путь в удаленную сеть через маршрутизатор с доступом к сети адреса назначения. Для передачи пакетов может использоваться статическая и динамическая маршрутизация. Статические маршруты задаются настраивающим данную сеть сетевым администратором, и на маршруты может повлиять только администратор данной сети. Динамические маршруты формируются автоматически, при помощи протоколов динамической маршрутизации. При использовании динамической маршрутизации в задачи сетевого администратора входит только настроить протокол динамической маршрутизации на роутерах, которые находятся в зоне ответственности этого сетевого администратора. Маршруты роутеры смогут передавать друг другу без участия сетевого администратора. Динамическая маршрутизация является удобным средством для резервирования. Протоколы динамической маршрутизации делятся на 2 больших класса- дистанционно-векторные и протоколы по состоянию канала.

Отличительной особенностью протоколов по состоянию канала является использование для определения метрики пропускную способность канала и стоимость каждого маршрута. Оптимальным маршрутом является маршрут с наименьшей метрикой. OSPFv2 (для IPv4 сетей) и OSFv3 (для сетей IPv6) является протоколом динамической маршрутизации по состоянию канала.

В ходе данной работы был запрограммирован протокол динамической маршрутизации OSPFv3 на языке P4. Язык P4 изначально задумывался как язык для программирования коммутаторов. Этот язык может использоваться для создания программно-определяемых сетей (SDN) – сетей передачи данных, в которых уровень управления сетью отделен от устройств передачи данныхз и реализуется программно. Является одной из форм виртуализации сети. Ключевыми принципами является разделение процессов передачи и управления данными, централизация управления сетью при помощи унифицированных программных средств и виртуализации сетевых ресурсов.

В результате проведенных исследований было обнаружено, что YANG и P4 не обладают возможностью реализации алгоритмов, так как они не являются языками программирования, однако неплохо поддерживают описание структур данных и автоматов состояния маршрутизаторов и протоколов.

# Список использованных источников

Материал предоставлен редакцией журнала Системный администратор.

Опубликовано в журнале "Системный администратор" N 8 2008

# Приложение