**BC/NW 2021 №1 (37):2.1**

**ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ПРОГРАММНО-ОПРЕДЕЛЯЕМОЙ СЕТИ ДЛЯ ОЦЕНКИ ВРЕМЕНИ ОБРАБОТКИ СЛУЖЕБНОГО ТРАФИКА**

**Абросимов Л.И., Игнатьев М.А.**

**Введение**

Традиционные компьютерные сети (ТКС) являются инфраструктурой, комплексно использующей современные информационные технологии (ИТ) , однако в настоящее время архитектура ТКС устарела и уже не всегда способна адекватно и эффективно соответствовать новым требованиям.

Актуальность работы связана с ключевыми тенденциями и требованиями рынков, а именно: быстрый рост трафика, рост сложности сети из-за использования дополнительных функций: поддержка мобильности, VPN, туннелирование, управление сетями, и т.д. Растущие объемы передаваемого трафика приводят к тому, что сети становится динамическими и зачастую требуют быстрых реакций на изменение их состояния. Этому препятствует высокая сложность управления сетью ввиду наличия большого числа производителей оборудования, каждый из которых разрабатывает собственную аппаратную платформу, команды конфигурирования, а также проприетарные протоколы взаимодействия устройств.

В результате этапы проектирования и пуско-наладочных работ т.н. мультивендорских сетей занимают довольно существенные временные промежутки, поскольку требуется учитывать все особенности взаимодействия устройств и протоколов.

Каждый сетевой узел поддерживает в актуальном состоянии собственное локальное представление о сети с помощью специализированных протоколов. Однако в случае необходимости изменения конфигурации сети глобальный администратор лишен инструмента, предоставляющего глобальный взгляд на сеть, и вынужден менять настройки каждого отдельного устройства. Существующие протоколы, подобные SNMP и NetFlow, призваны решать скорее задачи мониторинга, а не управления сетью.

Рост количества и разнообразия мобильных устройств, развитие различных технологий беспроводной связи привели к тому, что сегодня число их пользователей превысило число пользователей сетей с некоммутируемыми каналами связи. Однако рост мощности мобильных терминалов стимулирует увеличение вычислительной емкости приложений, что, в свою очередь, требует увеличения пропускной способности каналов связи — объем мобильного трафика растет в геометрической прогрессии, а виды трафика становятся все более разнообразными. По данным ведущих производителей сетевого оборудования, трафик удваивается примерно каждые девять месяцев, что в ближайшие несколько лет приведет к увеличению нагрузки на несколько порядков. В то же время сегодня эффективность доступного спектра частот для мобильных сетей уже близка к насыщению. [4]

Развитие микропроцессорной техники и телекоммуникаций привело к тому, что сейчас на каждого человека приходится в среднем около 40 чипов, однако появляются все новые сетевые устройства, внесение любых изменений в их существующие конфигурации трудоемко, затратно и практически невозможно без привлечения производителя. Нельзя гарантировать, что программно-аппаратные средства производителя содержат только ту функциональность, которая описана в документации, а в сетях ситуация может быть еще сложнее — такая функциональность может быть распределенной. Средства построения сетей сегодня проприетарны, их основной функционал реализован аппаратно и закрыт для изменений со стороны владельцев сетей.

Рост количества и разнородности контента, развитие сервисов и масштабов их охвата привели к изменению парадигмы организации вычислений — на место клиент-серверной архитектуры пришли ЦОД и облака, а файловые системы и базы данных трансформировались в сети хранения данных.

Одновременно с ростом количественных показателей нагрузки на сети усложнились задачи управления сетями — увеличились их перечень, значимость и критичность, при повышении требований к безопасности и надежности.

Итак, можно выделить следующие проблемы TKC:

• *технологические* — сегодня невозможно управлять и надежно прогнозировать поведение таких сложных объектов, как глобальные компьютерные сети;

• *экономически*е — сети дороги, сложны и требуют для своего обслуживания высококвалифицированных специалистов;

• *развития* — в архитектуре современных сетей имеются существенные барьеры для экспериментирования и создания новых сервисов.

Ответом на кризис компьютерных сетей стало появление принципиально нового подхода к их построению — программно-определяемых сетей (ПОС) (от англ. Software-defined Networking – SDN, также программно-конфигурируемые сети.)

Программно-определяемая сеть – сеть передачи данных для управления, в которой плоскость управления сетью отделена от устройств передачи данных (коммутаторы, маршрутизаторы) и реализуется программно, контроллером.

Контроллер, который (или сетевая ОС с приложениями) является ключевым элементом ПОС, выполняет функции управления элементами сетевой инфраструктуры и потоками данных в сети.

В результате - логическое централизованное управление, отделенное от передачи данных, реальное управление качеством передачи данных, инжинирингом трафика и маршрутизацией, удобный механизм абстракции и виртуализации сетевых ресурсов и сервисов, повышение эффективности распределения ресурсов и размещения сервисов в сети.

Освоение ПОС требует существенных усилий и затруднительно без практического опыта. Для освоения и экспериментального исследования установления функциональных характеристик ПОС требуется разработать экспериментальный стенд. Экспериментальный стенд включает в себя эмулятор компьютерной сети, контроллер программно-определяемой сети, который на основании параметров состояния сети «вырабатывает» управляющие команды составляющие управляющий трафик, программу для перехвата и анализа параметров сетевого трафика и программу для генерации трафика.

Наличие в составе ПОС введенных структурных модулей, реализующих дополнительные функции, позволит успешно разрешить технологические, экономические и проблем развития компьютерных сетей.

Цель исследований – повысить качество функционирования ПОС, используя для интегральной оценки качества временные параметры обработки данных

Для достижения поставленной цели необходимо решить следующие задачи:

* провести анализ функций дополнительно выполняемых средствами ПОС,
* выбрать состав программных и аппаратных модулей,
* разработать стенд для проведения исследований,
* провести тестирование и анализ результатов

1. **Особенности программно-определяемой сети (ПОС)**

Концептуальные особенности, положенные в основу ПОС:

• разделение процессов управления компьютерной сетью и обработки трафика данных;

• единый, унифицированный, независящий от поставщика интерфейс между уровнем управления и уровнем передачи данных;

• логически централизованное управление компьютерной сетью, осуществляемое с помощью контроллера с установленной сетевой операционной системой и реализованными поверх сетевыми приложениями;

• виртуализация физических ресурсов сети.

В архитектуре ПОС можно выделить три уровня (рис. 1) [4]:

* *инфраструктурный уровень*, предоставляющий набор сетевых устройств (коммутаторов и каналов передачи данных);
* *уровень управления*, включающий в себя сетевую операционную систему, которая обеспечивает приложениям сетевые сервисы и программный интерфейс для управления сетевыми устройствами и сетью;
* *уровень сетевых приложений*для гибкого и эффективного управления сетью.

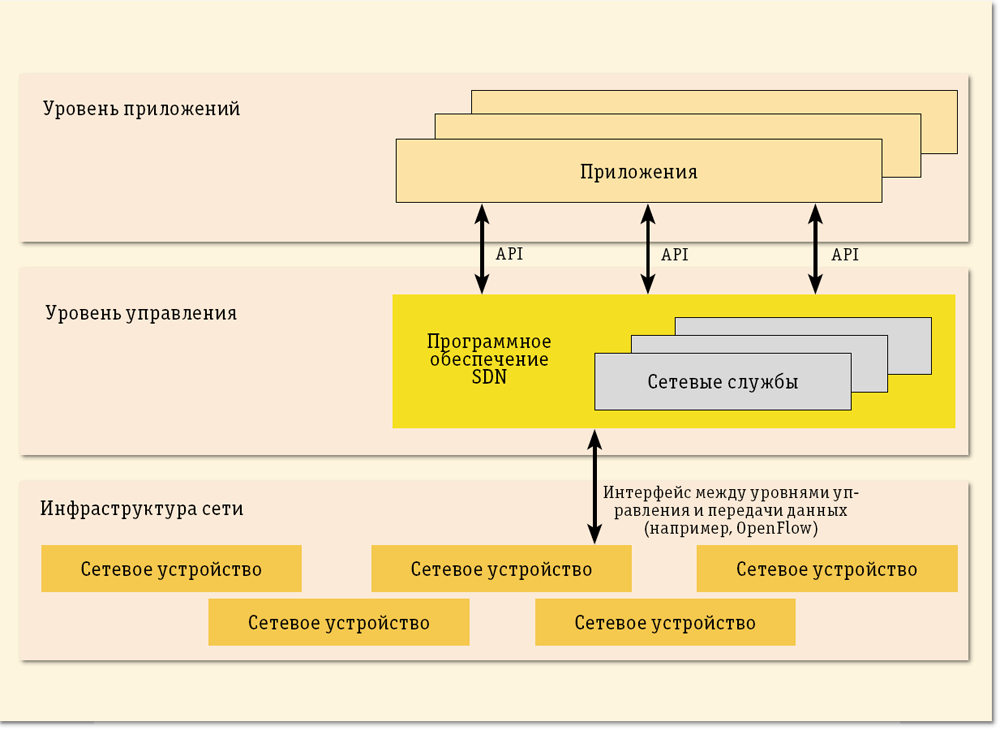


Рис. 1 Уровни архитектуры ПОС

Программно-определяемая сеть имеет три определяющие характеристики.

Во-первых, возможность отделить плоскость данных (то есть, пересылать пакеты в соответствии с решением, принятым плоскостью управления) от плоскости управления (то есть решение о маршрутизации или анализ полученных пакетов и определение решения, каким образом обрабатывать трафик в маршрутизаторах и коммутаторах).

Во-вторых, ПОС обеспечивает единую плоскость управления таким образом, чтобы плоскость данных мог управляться с помощью одной программы. Плоскость управления ПОС расширяет прямой контроль над элементами плоскости данных сети, т.е. коммутаторами и управление интерфейсами и плоскостью данных через OpenFlow, использующий интерфейс прикладного программирования (API).

В-третьих, этот способ обеспечивает сетевому администратору глобальный взгляд на всю сеть и возможность вносить изменения в глобальном масштабе вместо внесения изменений в каждый отдельный блок (ориентированный на устройство конфигурации).

Эта инновационная технология и концепция были первоначально предложены Nicira Networks, которая была основана на их предыдущих разработках в UCB, Standford, CMU, Princwton [10,17]. Последние работы в области ПОС ориентированы на исследование применения и расширения для широкого спектра сетей, которые могут включать в себя сети дома, базовые сети сотовой связи, корпоративные сети, сотовую связь, Wi-Fi сети и т. д.

Программно-определяемая сеть (ПОС) как программируемая сеть, имеющая функции «предоставление услуг на лету», вызывает большой интерес, как в академическом мире, так и в промышленности. [24]

Предлагаемая архитектура ПОС, реализованная на ресурсах программно-определяемой гетерогенной сети использует новые технологии, позволяющие открыть новые перспективы в области сетевых технологий, которые облегчат обработку огромного интернет-трафика и помогут инфраструктуре и поставщикам услуг динамически настраивать свои ресурсы.

Рост мультимедийного контента и растущий спрос на обработку больших данных требует более высоких скоростей сетевого соединения для удовлетворения потребностей растущего населения мира.

Рост трафика, а также будущие интернет-приложения с эффективной и экономичной доставкой пакетов являются большой проблемой для администратора сети.

Программно-определяемая сеть возникла, как хорошо организованная сетевая технология в этом быстро меняющемся сценарии сетей, что достигается путем оказания поддержки динамическими характеристиками будущих сетевых (FN) приложений при уменьшенном количестве ресурсных затрат и эксплуатационных расходах, благодаря простоте управления оборудованием и упрощенному управлению программным обеспечением.

***1.1 Сравнение программно-определяемой сети с ТКС***

Традиционная компьютерная сеть (ТКС) выполняет различные специальные алгоритмы и следует ряду установленных правил, которые запрограммированы в интегральных схемах специального назначения (ASICs) для контролирования и управления потоком данных по сети, маршрутизации и за настройку различных сетевых элементов друг с другом в сети. Когда в ТКС пакеты принимаются устройствами маршрутизации, то используется набор правил, которые уже заложены в его программно-аппаратном обеспечении, чтобы определить маршрут для этих пакетов, а также адрес устройства получателя в сети.

Обычно пакеты данных обрабатываются аналогичным образом, который может быть направлен в один и тот же пункт назначения, и все это происходит в недорогом устройстве маршрутизации. Кроме того, специальное устройство маршрутизации, к примеру, маршрутизатор Cisco, может иметь возможность обрабатывать различные пакеты в зависимости от их природы и содержимого. Это позволяет администратору выделять приоритеты различных потоков с помощью настраиваемого программирования локального маршрутизатора.

Таким образом, размер очереди в каждом маршрутизаторе может управлять потоком пакетов напрямую. Такая настраиваемая настройка локального маршрутизатора позволяет операторам более эффективно обрабатывать трафик с точки зрения перегрузки и контроля приоритетов.

Существующие сетевые устройства имеют ограничение на производительность сети из-за высокого сетевого трафика, что снижает производительность сети с точки зрения скорости, масштабируемости, безопасности и надежности. Существующим сетевым устройствам не хватает динамизма в работе, что связано с различными типами пакетов и их содержимым. Это может быть связано с невозможностью перепрограммирования работы сети из-за скрытой реализации правил маршрутизации и различных протоколов.

В связи с разделением задач коммутационного узла на два различных класса в архитектуре принято выделять две плоскости, реализующих различный функционал сетевого коммутационного узла:

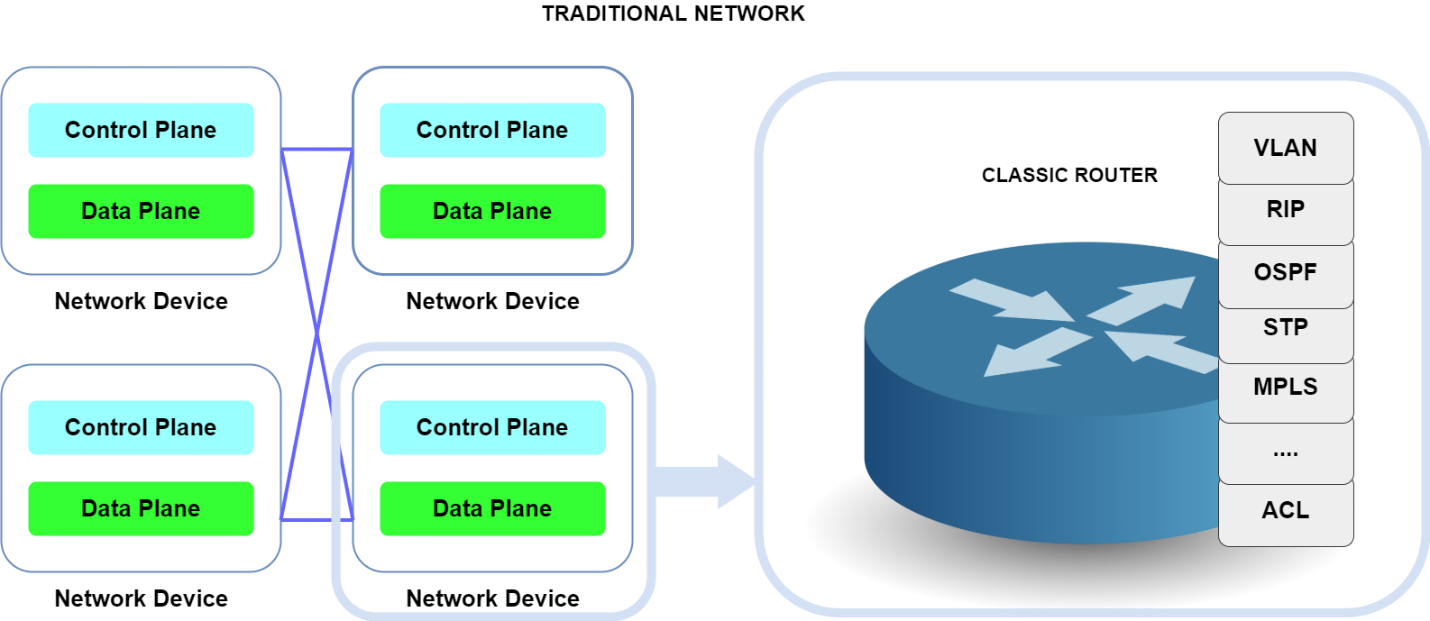
- плоскость управления (control plane);

- плоскость данных (data plane).

Плоскость управления реализует логику работы сетевого устройства, определяя с помощью различных служебных протоколов (например, VLAN, RIP, OSPF, STP и др.) правила для дальнейшего продвижения пакета данных.

Задача плоскости данных заключается в передаче полезного трафика через сетевое устройство в соответствии с установленными правилами.

Исходя из особенностей решаемых каждой плоскостью задач, производители систем коммутации достаточно давно [8] осознали, что разделение данных плоскостей и реализация их на различных аппаратных компонентах позволит повысить производительность. Сетевое устройство ТКС представлено на рисунке 2.

  
Рис. 2. Сетевые устройства в ТКС

Требуется соответствующая обработка правил данных в виде программного модуля для преодоления недостатков, описанных выше.

Централизованное управление с помощью программного модуля поможет улучшить контроль над сетевым трафиком за счет эффективного использования сетевых ресурсов, что может привести к современной технологии, известной как ПОС.

Централизованное управление также позволяет пользователю облака использовать облачные ресурсы, такие как хранилище, обработка (вычисления), пропускная способность и виртуальные машины (ВМ), или проводить научные эксперименты, создавая более эффективные срезы виртуальных потоков.

Цель ПОС – предоставить платформу с открытым, управляемым пользователем управлением для устройства переадресации в сети. В нем, в зависимости от масштаба сети, плоскость управления может иметь один или несколько контроллеров. В случае среды с несколькими контроллерами, высокоскоростное, надежное распределенное управление сетью может быть сформировано одноранговой (P2P) конфигурацией.

В крупномасштабных высокоскоростных вычислительных сетях, отделение плоскости данных от плоскости управления играет важную роль в ПОС, причем коммутаторы используют таблицу потоков для пересылки пакетов на плоскости данных. Таблица потоков содержит список записей потоков, и каждая запись имеет три поля, т.е. сопоставление, счетчик и инструкцию, что приводит к улучшению производительности сети по отношению к обработке данных, контролю и управлению сетью. В результате программный модуль (приложение) помогает администратору контролировать поток данных вместе с желаемым изменением характеристик устройств коммутации и маршрутизации в сети централизованно, не имея дело с каждым устройством отдельно в сети [10]. Сетевые устройства в ПОС представлены на рисунке 3.

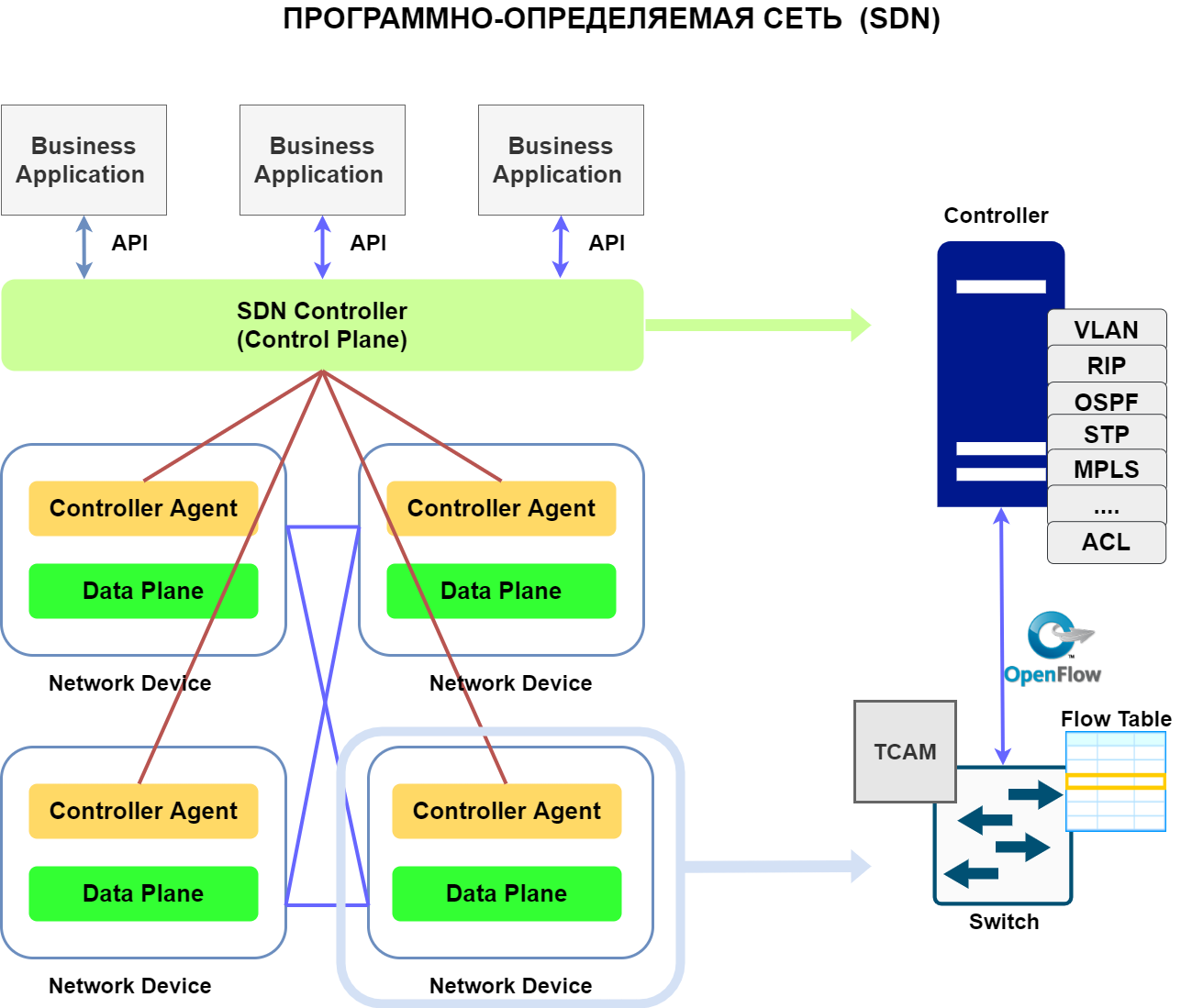


Рис. 3. Сетевые устройства в программно-определяемой сети

Можно также сказать, что продвижение в ПОС должно оставаться необычным эволюционным этапом, при котором стандарты OpenFlow также используются наряду с новыми услугами за счет использования виртуализации, в частности, для оптической транспортной сети, контроль и управление, для дальнейшего улучшения его емкости и эффективности.

Ввиду технологических достижений интернета, вовлечены сложные процессы и прилагаются усилия для решения разнообразных социальных проблем. В настоящее время ведутся исследования и разработки для реализации NGN. Полностью оптическая сеть является перспективной технологией для FN. В этой оптической пакетно-схемной интегральной сети (OPCInet) предлагаются разнообразные услуги, повышение функциональной гибкости и эффективности использования энергии потребление с высокой скоростью коммутации в системе ПОС на основе пакетов в сети метро / базовой сети [9].

***1.2 Результаты сравнения ПОС и ТКС***

1) Увеличение пропускной способности существующей ТКС без использования архитектуры и средств ПОС **– *неэффективно.***

*Обоснование* Для масштабирования ТКС в связи с быстро растущим трафиком потребуется дополнительные инвестиции в сетевую инфраструктуру, поэтомусеть ТКС становится огромной по размеру и для небольших по размеру организаций требуется от 100 до 1000 устройств.

Природа ТКС неоднородна, из-за оборудования, приложений и услуг, которые предоставляются различными производителями, поставщиками и провайдерами, поэтому управление сетями требует повышенной квалификации.

Человеческий фактор также способствует простоям сети (неисправностям), поэтому ручная настройка сетевого оборудования может привести к отключению сетевых устройств.

Перечисленные трудности традиционного подхода к настройке, оптимизации и устранению неисправностей станет неэффективным, а в некоторых случаях недостаточным.

Только решения, основанные на ПОС позволит преодолеть указанные проблемы.

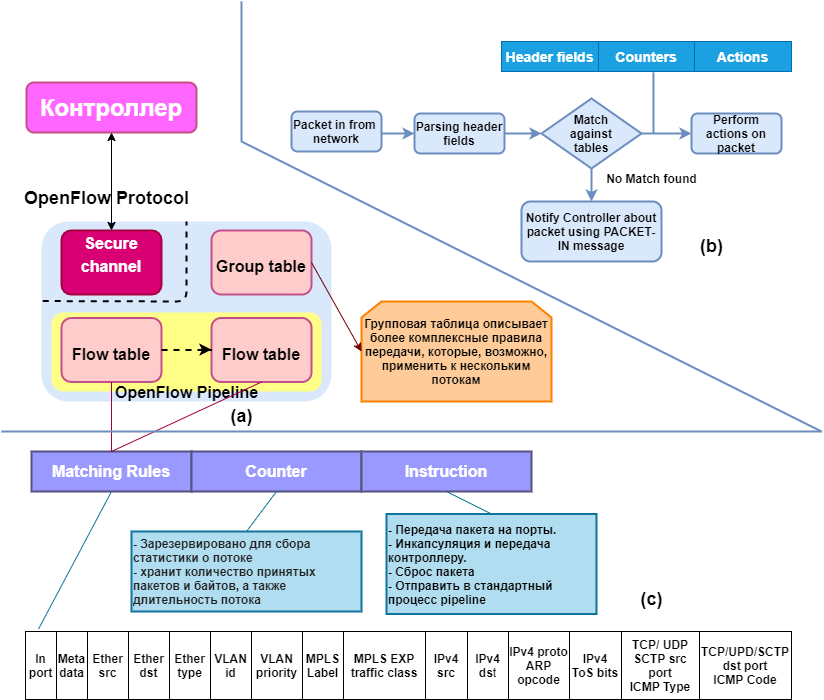
2) Масштабируемость, надежность и производительность ТКС являются основными задачами для эффективной работы программно-определяемой оптической сети (SDON). Из исследования в большой эмулированной сети с 100 000 конечными узлами и 256 коммутаторами наблюдалось, что по крайней мере 50000 новых запросов потока OpenFlow в секунду управляются различными реализациями контроллера, такими как NOX, Maestro, Beacon и т. д. [1]. Это подтверждает, что одним контроллером может быть обработано большое количество новых запросов потока. Таким образом, проблемы масштабируемости, надежности и производительность сети могут быть решены эффективно.

3) При потребности мобильности и перемещении сервисов, на основе оптических технологий ПОС, могут быть развернуты устройства управления и балансировки нагрузки для упрощения реализации программируемого потока трафика внутри ЦОД. При этом, приняты во внимание пропускная способность и возможные временные затраты для разных приложений [7].

**2 Протокол OpenFlow для стандартизации интерфейсов между коммутатором и программным контроллером в архитектуре ПОС**

Протокол OpenFlow был разработан для стандартизации связей между коммутатором OpenFlow и программным контроллером ПОС.

Архитектура OpenFlow состоит из трех основных компонентов, как показано на рис. 4(а), где OpenFlow Protocol - совместимые коммутаторы, составляющие плоскость данных. Плоскость управления имеет один или несколько контроллеров OpenFlow. Интерфейс OpenFlow, по которому контроллер связан с коммутаторами через безопасный канал управления.

  
Рис.4. Архитектура OpenFlow: a) – компоненты архитектуры OpenFlow, b) – алгоритм передачи данных в совместимых коммутаторах OpenFlow, c) – таблица потоков

Алгоритм передачи пакетов с OpenFlow представлен на рисунке 4(b). Таблица потоков, как показано на рисунке 4(c), содержит список записей потоков. Каждая запись имеет поля совпадения, счетчики и инструкции,

Когда пакет получен коммутатором, он анализирует заголовок пакета и выполняет сопоставление с записями в таблице потоков коммутатора. Если запись таблицы потоков сопоставляется с заголовком пакета, то эта конкретная запись рассматривается. При нахождении нескольких таких записей, в этом случае пакеты сопоставляются на основе приоритетов, то есть наиболее конкретная запись будет иметь самый высокий приоритет, как показано на рисунке 4(c). После завершения процесса сопоставления и выбора, счетчик записи в таблице потоков обновляется. В результате, коммутатор выполняет действие над пакетом в соответствии с записью в таблице потоков. Поле действий служит для применения определенных действий к пакетам при появлении совпадения. Выделяют два вида действий (actions) – обязательные (required) и опциональные (optional). К обязательным действиям относятся:

1. Отправка пакетов на физические или виртуальные порты

А) All – отправить пакет со всех интерфейсов, за исключением входящего (порт, на который пакет прибыл);

Б) Local – отправить пакет на локальный порт;

В) Controller – отправить пакет контроллеру;

Г) In\_Port – отправить пакет на входящий порт;

2) Сброс пакета в том случае, если с совпавшим правилом не ассоциировано никакое действие.

К опциональным действиям относятся:

1) Отправка пакетов на виртуальные порты

А) Normal – обработка пакета с использованием коммутации второго и третьего уровней L2 и L3, а также VLAN;

Б) Flood – рассылка пакетов по топологии связного дерева STP;

2) Enqueue – используется для обеспечения качества обслуживания QoS, пересылает пакет через очередь на порту;

3) Modify Field – это опциональное действие, но оно позволяет во много раз увеличить полезность протокола OpenFlow. С помощью данного действия можно изменить VLAN или MAC-адрес источника/назначения, номер TCP/UDP порта и т.д.

В случае, если заголовок пакета не находит соответствия с записью таблицы потоков, коммутатор уведомляет контроллер, инкапсулирует пакет и отправляет его в контроллер с сообщением PACKET-IN в качестве первого байта пакета. Получив уведомление PACKET-IN от стороны коммутатора, контроллер находит точное действие для пакета и устанавливает одну или несколько подходящих записей в запрашивающей таблице потоков коммутатора, а затем пакеты пересылаются в соответствии с правилами. Это вызвано явными сообщениями PACKET-OUT. Обычно контроллер прокладывает полный путь для пакета, изменяя записи в таблицах потоков всех коммутаторов на пути в сети. Программное обеспечение, называемое контроллером, все записи таблицы потоков обрабатываются и заполняются контроллером путем вставки, удаления записей потока и изменений.

Протокол OpenFlow, независимо от версии, поддерживает три типа сообщений:

• контроллер-коммутатор;

• асинхронные;

• симметричные.

Сообщения типа контроллер-коммутатор инициируются контроллером и используются для управления и контроля состояния коммутатора. Сообщения данного типа могут использоваться контроллером для установления и запроса параметров конфигурации OpenFlow коммутатора, для сбора статистических данных с коммутаторов, для добавления, удаления и модификации записей в таблицах потоков [4]. Для версии OpenFlow 1.0 существует шесть видов сообщений коммутатор-контроллер:

• Read state - используется контроллером для чтения счетчиков (counters) потоков (per flow), портов (per port) и очередей (queue);

• Modify state – используется преимущественно для добавления, удаления или изменения потоков в таблицах потоков на коммутаторе;

• Send packet – используется для отправки пакетов из определенного порта коммутатора;

• Barrier request/replies – используется для получения подтверждений о завершении операций коммутатором. Если коммутатор получил сообщение о запросе блокировки от контроллера, то он должен завершить все текущие задачи. Коммутатор может перейти к следующей задаче только после завершения текущей. Как только коммутатор завершит все задачи, он отправляет подтверждение блокировки контроллеру и начинает выполнять следующую задачу;

• Features – это сообщения, с помощью которых контроллер запрашивает сведения о поддерживаемых коммутатором функциях;

• Configuration – используется контроллером для получения и установки параметров коммутатора.

В версии протокола OpenFlow 1.3 существует восемь видов сообщений:

• Read state, Modify state, Barrier, Features, Configurations остались без изменений;

• Role-Request – используется для выбора роли контроллера в кластере;

• Packet out используется взамен Send packet и расширяет возможности: данные сообщения используются контроллером для отправки пакетов из указанного порта на коммутаторе и для пересылки пакетов, полученных через сообщения Packet-in. Сообщения Packet out должны содержать полный пакет или идентификатор буфера, указывающий на пакет, хранящийся в коммутаторе. Сообщение должно также содержать список действий, которые должны применяться в том порядке, в котором они указаны;

• Asynchronous-configuration - используется контроллером для установки дополнительного фильтра для асинхронных сообщений или для запроса этого фильтра. В основном используется при организации кластера контроллеров.

Сообщение асинхронного типа инициируются коммутатором для оповещения контроллера о сетевых событиях (прибытии пакетов или удалении записи из таблицы по тайм-ауту), а также об изменениях состояния коммутатора или ошибках. Они бывают четырех разных видов (как в версии протокола 1.0, так и 1.3):

• Port status – сообщения о любых изменениях статуса порта отправляются контроллеру. Под изменением статуса порта понимается не только отключение порта, но и изменение топологии связующего древа STP;

• Packet in – это сообщение отправляется контроллеру только в тех случаях, когда для входящего пакета на коммутаторе не найдено совпадений в наборе записей (в этом случае пакет должен быть отправлен на контроллер) или пакет соответствует правилу и должен быть отправлен на контроллер;

• Flow removed – для наборов записей, добавленных в таблицу потоков существует два тайм-аута: тайм-аут простоя (idle timeout) и жесткий тайм-аут (hard timeout). Всякий раз при удалении потока из-за превышения тайм-аута, коммутатор отправляет контроллеру сообщение об изменении потока «flow removed»;

• Error – отправляется коммутатором контроллеру для информирования о любых ошибках.

Сообщения симметричного типа могут инициироваться коммутатором или контроллером без запроса и используются при установлении соединения. Помимо этого, такие сообщения могут также использоваться для измерения 85 задержек, пропускной способности соединения контроллер-коммутатор или проверки состояния соединения. Выделяют три вида сообщений:

• Hello – как и во многих других протоколах, обмен приветственными сообщениями происходит между контроллером и коммутатором при запуске;

• Echo – эти сообщения служат для измерения времени соединения между контроллером и коммутатором;

• Vendor – сообщения, несущие в себе информацию о дополнительной функциональности коммутаторов. В версии OpenFlow 1.3 сообщение Vendor заменено на Experimenter с аналогичным функционалом.

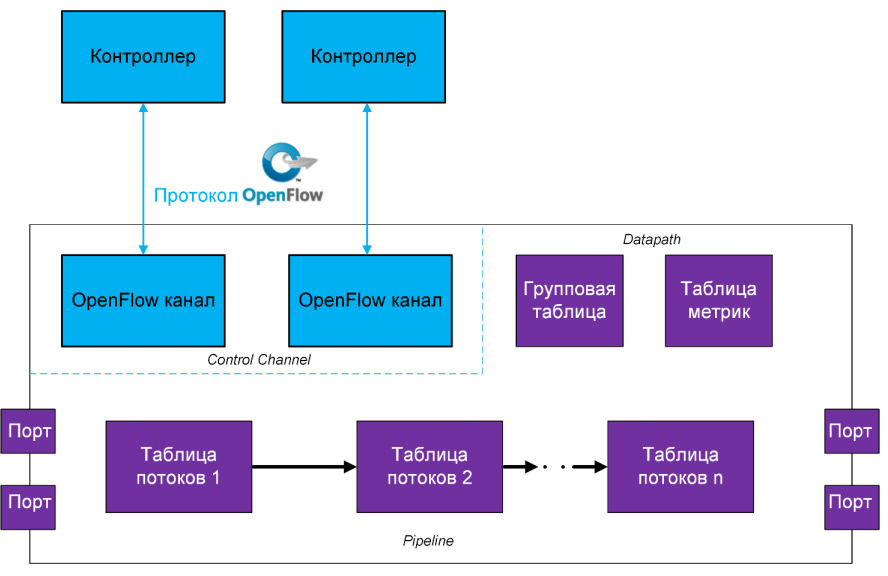
**3. Коммутатор с поддержкой протокола OpenFlow**

В ПОС-коммутаторе с поддержкой протокола OpenFlow можно выделить следующие логические компоненты (рис.5.):

* OpenFlow канал – отвечает за связь с внешним ПОС-контроллером и приём от него управляющих команд; по сути, организует связь плоскостей управления и данных;
* Таблица потоков;
* Групповая таблица;
* Таблица метрик;
* Порты.

Порты, таблицы потоков, групп, метрик (и прочие функции, определяемые спецификацией протокола) являются элементами data plane.

В их задачи входит обеспечение продвижение пакета с входного на выходной порт. Сами же таблицы программируются командами в ПОС-контроллера.

Рис.5. Устройство ПОС-коммутатора с поддержкой OpenFlow

OpenFlow-совместимые коммутаторы делятся на два типа: чистые OpenFlow коммутаторы и гибридные. Коммутаторы первого типа поддерживают работу только по протоколу OpenFlow. В таких устройствах все пакеты обрабатываются потоками OpenFlow и не могут быть обработаны иначе.

Гибридные OpenFlow коммутаторы поддерживают оба режима работы: OpenFlow и обычную Ethernet-коммутацию (традиционную L2-коммутацию, VLAN, маршрутизацию и т.д.). Например, коммутатор может использовать тег VLAN или входной порт пакета для принятия решения о его продвижении, или может направить все пакеты на обработку по протоколу OpenFlow. Этот механизм классификации выходит за рамки данной спецификации. Гибридные коммутаторы могут также пропускать пакеты от обработки OpenFlow к нормальной обработке через обычные или потоковые зарезервированные порты.

В спецификации OpenFlow определяется следующий набор стандартных портов – physical, logical и reserved. Physical –физические порты коммутатора. 87 логических (logical) портов напрямую не привязаны к физическим портам оборудования (MPLS LSP, Null интерфейсы). Зарезервированные порты зачастую используются в гибридных коммутаторах (OpenFlow + традиционная сеть).

Любой OpenFlow-коммутатор, в зависимости от поддерживаемой версии протокола, содержит одну или более таблиц потоков, каждая из которых содержит множество записей потоков. Процессы обработки OpenFlow определяют, как пакеты взаимодействуют с этими таблицами потоков. Коммутатор OpenFlow нуждается хотя бы в одной входящей таблице потоков. Поточная обработка на коммутаторе с единственной таблицей потоков существенно упрощается. Таблицы потоков последовательно нумеруются, начиная с нуля. Поточная обработка проходит в два этапа: обработка на входе и выходе. Разделение этих двух этапов обозначается первой выходной таблицей. Все таблицы, номера которых ниже первой выходной таблицы должны использоваться как входные, а таблицы с номерами выше, либо равные первой выходной таблице, так же могут быть использоваться как выходные.

На рисунке 6 представлен алгоритм действий, совершаемых с пакетом внутри коммутатора с поддержкой OpenFlow 1.5 до момента его отправки через исходящий порт (т.н. pipeline).

Поточная обработка всегда начинается с входящей обработки на первой таблице потоков: пакет должен сначала сопоставиться с записями потоков таблицы под номером 0. Остальные входные таблицы могут быть использованы исходя из результатов сравнения с первой таблицей. В результате совпадения входной обработки пакет отправлен на выходной порт, коммутатор может выполнить выходную обработку в контексте данного выходного порта.

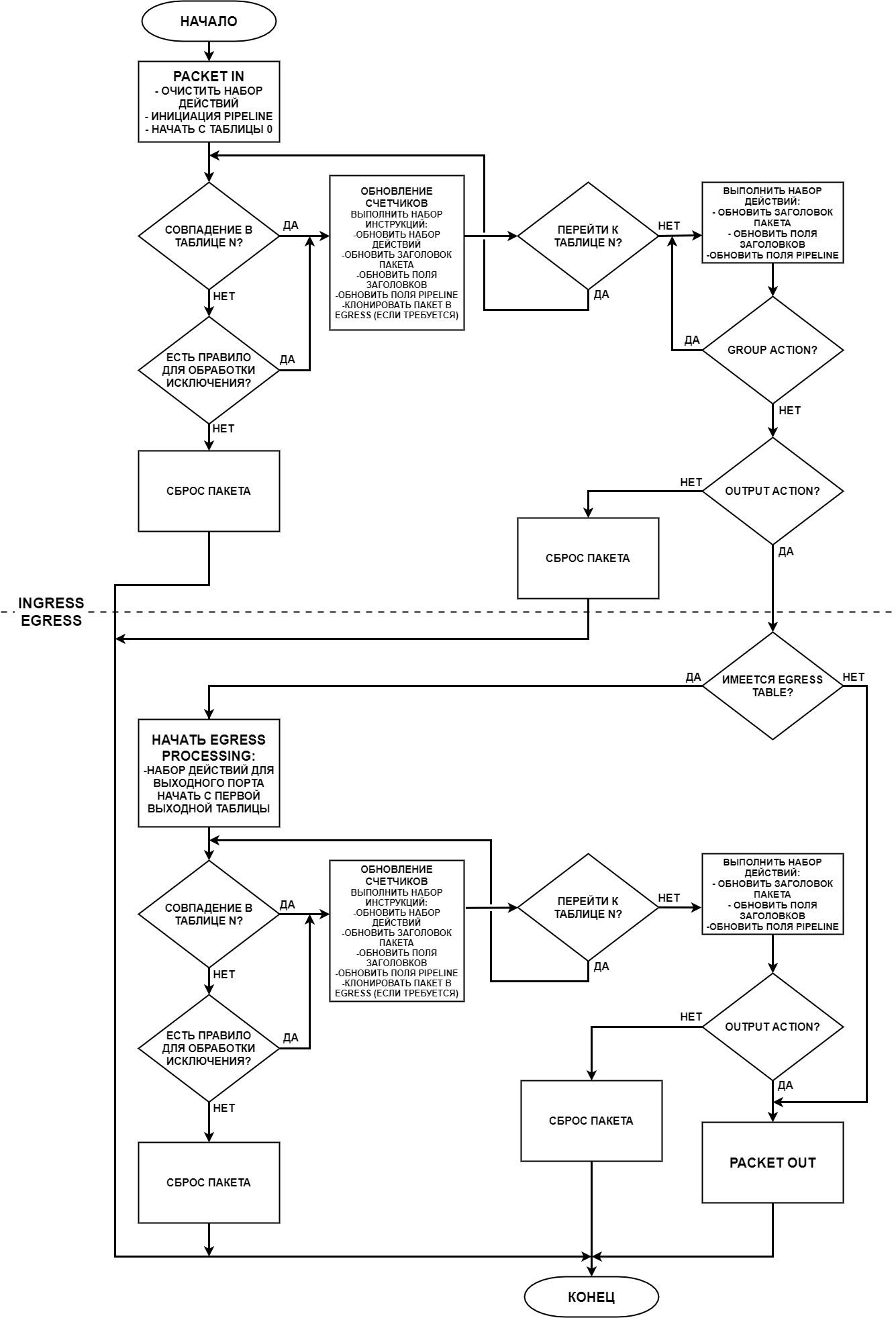
Обработка исходящих данных не обязательна и коммутатор может не поддерживать выходные таблицы или может не использовать их. Если ни одна выходная таблица, не установлена как первая выходная таблица, тогда пакет обязан быть обработан выходным портом и в большинстве случаев отправлен. Если выходная таблица установлена как первая выходная, пакет обязан быть сравнен с записями потоков этой таблицы и остальные выходные таблицы могут быть использованы исходя из результатов сравнения пакета с этой таблицей.

При обработке таблицей потоков, пакет сравнивается со всеми записями потоков данной таблицы. Если запись найдена, применяется инструкция, занесенная с этой записью. Эти инструкции могут строго направить пакет в другую таблицу потоков, где тот же процесс повторится снова. Запись потока может направлять пакет только в таблицы с большим номером, чем номер таблицы, к которому она принадлежит. Если соответствующая запись потока не отправляет пакеты к другой таблице, то на данном этапе конвейерная обработка данных останавливается, пакет обрабатывается соответствующим набором действий и, как правило, отправляется.

Если пакет не совпадает с записями потоков в таблице, коммутатор расценивает данный процесс как несовпадение с данной таблицей потоков. Поведение таблицы при несовпадениях зависит от конфигурации таблицы.

Инструкции, заложенные в отсутствующие записи потоков таблицы, могут гибко указывать как поступить с несогласованными пакетами. Полезными опциями являются отбрасывание пакетов, проверка их с другой таблицей или отправка пакетов на контроллер по каналу управления в качестве packet-in сообщений.

Существует несколько случаев, когда пакет не полностью обработан записью потока и поток обработки останавливается без установки действия для пакета или направления его к другой таблице. Если не существует несовпадений с таблицей, пакет отбрасывается. Если найдено недопустимое значение TTL, пакет может быть отправлен на контроллер.

  
Рис.6. Алгоритм обработки пакета на OpenFlow-коммутаторе

**4. Контроллер программно-определяемой сети с открытым исходным кодом**

***4.1 Контроллер программно-определяемой сети***

За последние годы технология ПОС быстро развивается. В связи с активным развитием, существует множество контроллеров ПОС, доступного для использования.

Базовая концепция ПОС состоит в том, чтобы отделить плоскость управления (т.е. маршрутизацию) от элементов передачи (т.е. коммутаторов) и сосредоточить контроль над управлением и работой сети в логически централизованном компоненте - контроллере ПОС.

Несмотря на то, что существует много контроллеров ПОС, приводится сравнение уровня зрелости популярных контроллеров ПОС с открытым исходным кодом в промышленности и научных кругах, включая: операционную систему с открытыми сетями (ONOS), OpenDayLight (ODL), OpenKilda, Ryu. Эти контроллеры ПОС можно оценить по следующим критериям оценки:

1. Архитектура
2. Модульность
3. Масштабируемость
   1. Кластерная масштабируемость
   2. Архитектурная масштабируемость
4. Интерфейсы
   1. Поддержка северного API
   2. Поддержка южного API
5. Телеметрия
6. Отказоустойчивость
7. Язык программирования
8. Сообщество

Важно понимать мотивы доступных платформ. Каждый проект имеет разные варианты использования, поскольку использование зависит не только от ряда возможностей, но также от культурного соответствия организации и проекта. Контроллеры ПОС имеют множество различных вариантов использования, включая: управление трафиком, маршрутизацию сегментов, интеграцию и автоматизированное управление трафиком.

***4.2 Контроллер OpenDayLight (ODL)***

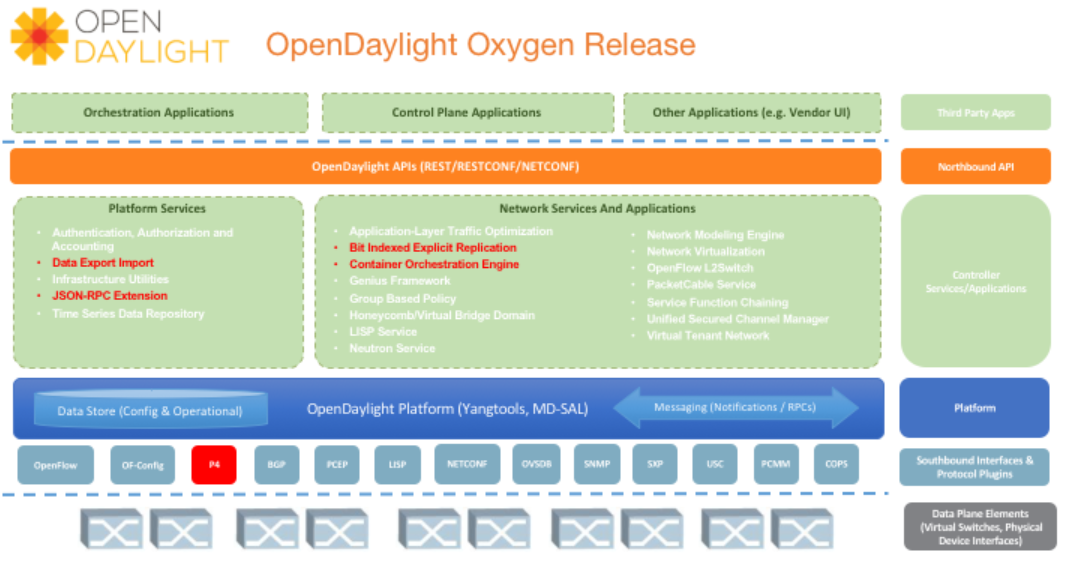
Контроллер OpenDayLight (ODL) ориентирован на SD-LAN и облачные интеграционные пространства. OpenDaylight — это модульная открытая платформа для настройки и автоматизации сетей любого размера и масштаба. Проект OpenDaylight возник в результате движения ПОС с четким акцентом на программируемость сети. С самого начала он был разработан как основа для коммерческих решений, которые предназначены для различных вариантов использования в существующих сетевых средах.

АрхитектураODL, представленная на рисунке 7, состоит из трех уровней:

- подключаемые модули на южном уровне для связи с сетевым устройствами,

- базовые сервисы, которые можно использовать с помощью Service Abstraction Layer (SAL), который основан на OSGi, чтобы помочь компонентам входить и выходить из контроллера во время его работы,

- северныеинтерфейсы (например, REST / NETCONF), которые позволяют операторам применять политики высокого уровня к сетевым устройствам или интеграции ODL с другими платформами.

Рис.7. Архитектура OpenDayLight

Встроенные механизмы, предоставляемые ODL, упрощают подключение модулей кода. Контроллер использует преимущества контейнеров OSGi для загрузки пакетов во время выполнения, что позволяет очень гибко подходить к добавлению функциональности.

В контроллере ODL использует основанный на модели подход, который подразумевает, что для выполнения логических вычислений требуется глобальное представление сети в памяти. Последний выпуск ODL дополнительно расширяет масштабируемость и надежность платформы благодаря новым возможностям, поддерживающим развертывание на нескольких площадках для обеспечения географического охвата, производительности приложений и отказоустойчивости.

Кластерная масштабируемостьODL содержит внутреннюю функциональность для обслуживания кластера, AKKA в качестве распределенного хранилища данных разделяет текущее состояние ПОС и позволяет контроллерам переходить на другой ресурс в случае разделения кластера.Однако по мере роста кластера активность в области связи и координации быстро увеличивается, что ограничивает прирост производительности на каждого дополнительного элемента кластера.

Архитектурная масштабируемостьODL включает собственные возможности маршрутизации BGP для координации потоков трафика между островами ПОС.Введение OpenDaylight в OpenStack обеспечило многосайтовую (multi-site) сеть, одновременно повышая производительность сети.

На уровне проекта ODL имеет ограниченную функциональность, связанную с телеметрией. В последнем выпуске разработки появились шаги к предоставлению телеметрических каналов на верхнем уровне, но они находятся на ранней стадии разработки и вряд ли будут готовы к производству в ближайшей перспективе.

Механизм отказоустойчивости ODL аналогичен ONOS с нечетным количеством контроллеров ПОС, необходимых для обеспечения отказоустойчивости в системе. В случае отказа главного узла будет выбран новый лидер, который возьмет под контроль сеть. Механизм выбора лидера немного отличается в этих контроллерах - в то время как ONOS фокусируется на согласованности, ODL фокусируется на высокой доступности.

Контроллер ODL написан на java и является вторым из контроллеров ПОС под зонтом Linux Foundation Networking. Этот проект пользуется наибольшей поддержкой сообщества среди всех открытых ПОС-контроллеров на рынке, при этом несколько крупных компаний активно участвуют в его разработке.

Контроллер OpenDayLight — это наиболее распространенный ПОС-контроллер с открытым исходным кодом с обширными северным и южным API. В дополнение к отказоустойчивости и масштабируемости модульная архитектура ODL делает его подходящим выбором для различных вариантов использования. Вот почему OpenDayLight был интегрирован в другие открытые ПОС / NFV-решения для управления, такие как OpenStack, Kubernetes, OPNFV и ONAP, которые являются очень популярными платформами в телекоммуникационных средах.

**5 Разработка экспериментального стенда**

Разрабатываемый стенд для исследования функциональных характеристик должен иметь набор инструментальных средств, осуществляющих генерацию варьируемых параметров, сбор, хранение и обработку измеренных данных, а также интерфейс с исследователем.

***5.1 Сравнение эмуляторов и симуляторов компьютерной сети***

В разделе решается задача выбора эмулятора для выполнения функций поддержки исследований и обучения GJC в ПОС. Приведены результаты сравнения трех сетевых симуляторов/эмуляторов, таких как Mininet, Cisco Packet Tracer, GNS3.

Подчеркнём разницу между симулятором и эмулятором.

Эмулятор запускает точную копию реальной сетевой операционной системы.

Симулятор – это программная реализация набора функций, которые будут имитировать функции реальной сетевой операционной системы.

Основная проблема исследований состоит в том, как проанализировать полученные результаты производительности передачи данных с помощью выбранных инструментов.

Сложности возникают при масштабировании вычислительных сетей, при проверке правильности оценки производительности и при реализации возможности перехода на реальную систему с минимальными изменениями.

GNS3 является «симулятором», но он способен имитировать целые сети, а не только сетевые операционные системы. Большинство пользователей используют GNS3 для эмуляции Cisco IOS, а также других поставщиков. Отличие GNS3 от других симуляторов состоит в его способности эмулировать маршрутизацию и коммутацию, а также включать реальные виртуальные машины и соединять их вместе через систему логического туннелирования (оверлейная сеть). GNS3 сделал успехи в эмуляции реальных производственных сетей, в отличие от других программных обеспечений.

Packet Tracer — это сетевой симулятор, который включает в себя только ограниченные возможности реального оборудования.

В настоящее время спрос на инструменты для оценки сетевых решений растет из-за необходимости тестирования их перед использованием в реальных условиях. Эта область нуждается в развитом сообществе, которое бы помогало в поддержке и разработке новых сетевых технологий. По этой причине изучение возможных сред для проведения экспериментов становится важным в современных условиях функционирования сетей.

5.1.1 Эмулятор компьютерной сети Mininet

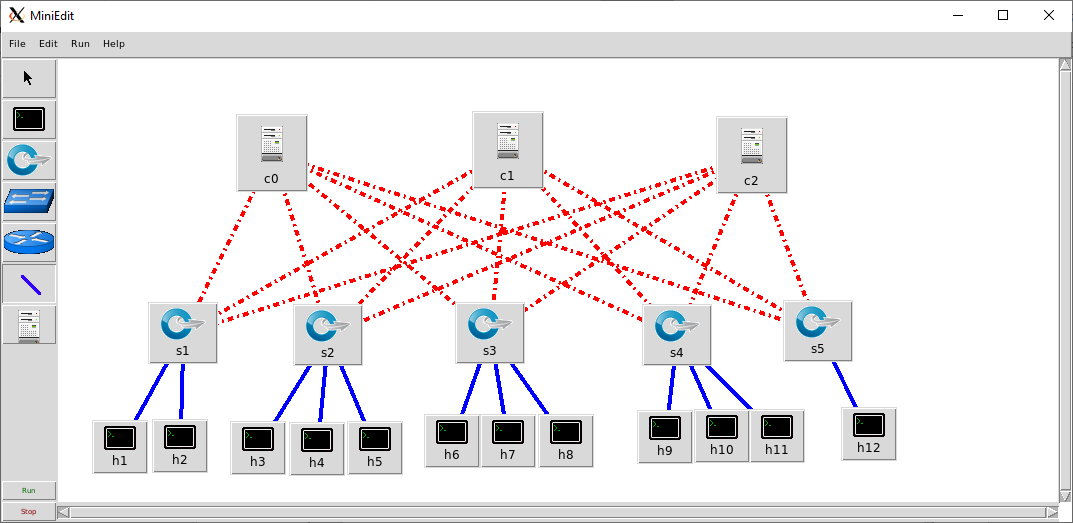
Эмулятор компьютерной сети MiniNet создает виртуальную сеть, с поддержкой OpenFlow, с контроллером, коммутаторами, конечными узлами и подключениями (link), а также позволяет разрабатывать собственные топологии с использованием скриптов Python. Пока что Mininet не обеспечивает определение истинных значений производительности для реальной сети, хотя используемый в ней код работает в реальной сети на основе NetFPGA и на основе коммерческих коммутаторов.

Эмулятор Mininet – эмулятор с открытым исходным кодом был создан для использования в качестве инструмента для исследований и преподавания сетевых технологий. \

Mininet предназначен для создания виртуальных программно-определяемых сетей, состоящих из контроллера OpenFlow, сети Ethernet с несколькими коммутаторами с поддержкой OpenFlow и несколькими конечными узлами, подключенными к этим коммутаторам. Он имеет встроенные функции, которые поддерживают использование различных типов контроллеров и коммутаторов. Существует возможность создавать сложные пользовательские сценарии, используя API Mininet Python.

Эмулятор Mininet эмулирует сети, включая стандартные сетевые приложения Unix / Linux, а также реальное ядро Linux и сетевой стек (включая любые расширения ядра, которые могут быть доступны, если они совместимы с сетевыми пространствами имен). Из-за этого код, который разрабатывается и тестируется в Mininet для контроллера OpenFlow, модифицированного коммутатора или конечного узла, может перемещаться в реальную систему с минимальными изменениями для реального тестирования, оценки производительности и развертывания.

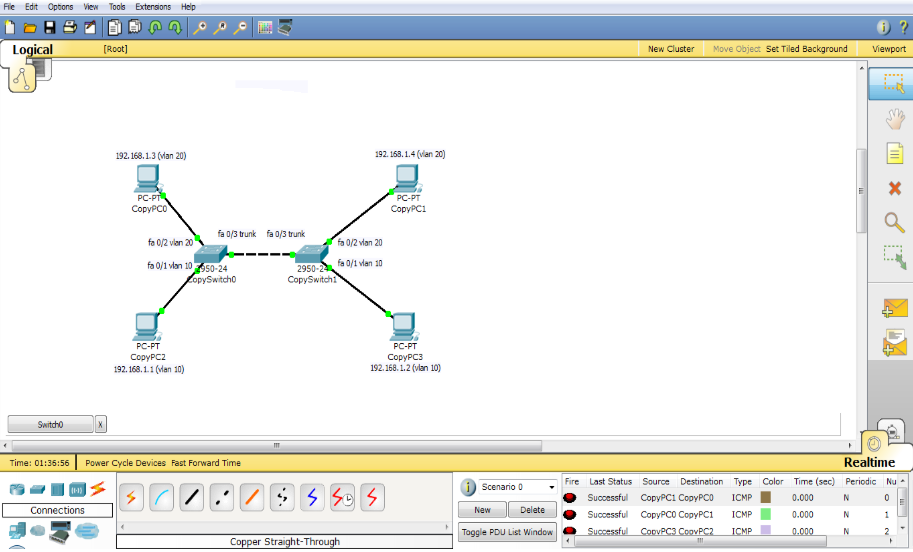
Важно отметить, что конструкцию, которая спроектирована в Mininet, обычно можно переносить непосредственно к аппаратным коммутаторам. Графический интерфейс Mininet показан на рисунке 8.

Рис.8. Графический интерфейс Mininet

5.1.2 . Симулятор компьютерной сети Cisco Packet Tracer

Симулятор Cisco Packet Tracer — это кроссплатформенная программа визуального моделирования, разработанная компанией Cisco Systems, которая позволяет пользователям создавать сетевые топологии и имитировать современные компьютерные сети. Cisco Packet Tracer — это мощная программа моделирования сети, которая позволяет студентам экспериментировать с сетевым поведением. Packet Tracer предоставляет возможности моделирования, визуализации, разработки и совместной работы. Цель Packet Tracer состоит в том, чтобы предложить студентам инструмент для изучения принципов сетевого взаимодействия, а также развить специфические навыки технологии Cisco.

Симулятор Cisco Packet Tracer можно скачать бесплатно, если у вас есть учетная запись Netacad. По рекомендациям Cisco, лучший способ узнать о сети — это смоделировать сеть. Практическое оборудование позволяет студентам начать работу, но ограничивается количеством устройств в лаборатории. Кроме того, преподаватели используют Packet Tracer для демонстрации сложных технических концепций и сетевых систем. Студенты используют Packet Tracer для выполнения заданий, работая самостоятельно или в команде. В работе бакалавра, я использовал для моделирования простых сценариев Packet Tracer, но были случаи, когда маршрутизаторы и коммутаторы, которые предоставлены в программе не поддерживали какие-то возможности, которые есть в реальных устройствах. Интерфейс Cisco Packet Tracer показан на рисунке 9.

Рис.9. Интерфейс Packet Tracer

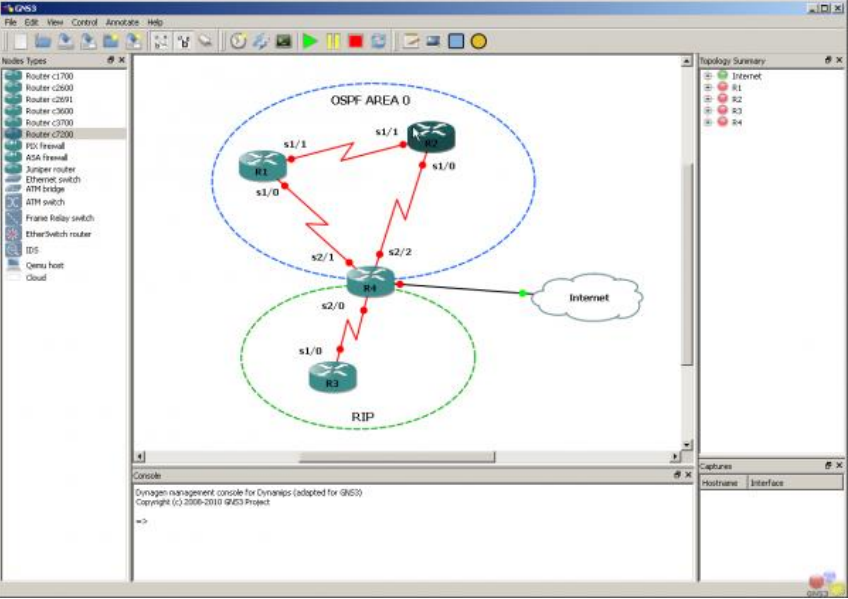
Симулятор Packet tracer-это симулятор Сisco IOS идеально подходит для практических лабораторий, для начинающих лабораторий (например, для исследований CCNA) Packet Tracer эффективен, но для продвинутых лабораторий целесообразно использовать GNS3, потому что большинство продвинутых лабораторий и команд не поддерживаются в Packet Tracer.

5.1.3 Эмулятор компьютерной сети GNS3

Эмулятор компьютерной сети GNS3 — это графический сетевой эмулятор, позволяющий пользователю запускать несколько эмулируемых систем, включая маршрутизаторы Cisco, Juniper, Vyatta, виртуальные машины Linux, Windows. Заставить GNS3 действительно выполнить это моделирование - не самая простая задача, особенно если вы хотите выйти за рамки простой топологии сети. GNS3 способен эмулировать Cisco IOS, что дает запускать Cisco IOS в виртуальной среде на компьютере. GNS3 — это графический интерфейс для продукта под названием Dynagen. Dynamips — это основная программа, которая позволяет эмулировать IOS. Dynagen работает поверх Dynamips, создавая более удобный для пользователя интерфейс командной строки (CLI). Пользователь может создавать сетевые топологии, используя простые файлы типа Windows INI-файлов с Dynagen, работающим поверх Dynamips. GNS3 делает этот шаг еще дальше, предоставляя графическую среду.

Эмулятор GNS3 позволяет эмулировать Cisco Ios на компьютере под управлением Windows или Linux. Эмуляция возможна для длинного списка платформ маршрутизаторов и брандмауэров PIX. Используемые карты EtherSwitch в маршрутизаторе и коммутаторе, также могут быть эмулированы в зависимости от степени поддерживаемой функциональности карты. Это означает, что GNS3 является бесценным инструментом для подготовки к сертификациям Cisco, таким как CCNA и CCNP.

Интерфейс GNS3, представленый на рисунке 11, отлично подходит для эмуляции физической сети поскольку позволяет эмулировать аппаратную часть сетевых устройств, непосредственно загружая и взаимодействуя с реальными образами Cisco IOS. GNS3 имеет свои недостатки, такие как, высокие системные требования из-за того, что в память загружается реальный образ IOS и не имеет возможности полноценно эмулирования коммутаторов Catalyst, а также некоторых моделей маршрутизаторов, из-за наличия в них большого количества ASIC (интегральная схема специального назначения).

Рис.10. Интерфейс GNS3

5.1.4 Выводы по результатам сравнения

В результате проведенных исследований установлено.

Packet Tracer является сетевым симулятором и встраивает только ограниченные функции реального оборудования,

GNS3-это сетевой эмулятор, основанный на Dynamips и QEMU, работающих с реальными образами IOS, виртуальными машинами, основным ограничением GNS3 является количество ресурсов (CPU / memory), доступных на ПК для запуска эмулированных образов IOS лаборатории и виртуальных машин для моделирования клиентов и серверов. Основные характеристики сетевых симуляторов/эмуляторов в сравнительном анализе показаны в таблице 1.

По результатам сравнительного анализа автором принято решение использовать эмулятор Mininet, т.к. это уникальный сетевой эмулятор с открытым исходным кодом, который разработан для поддержки исследований и обучения в программно-определяемой сети. Интерфейс командной строки Mininet очень прост в использовании, так как Mininet использует сетевые пространства имен в качестве своей технологии виртуализации. Он может поддерживать большое количество виртуальных узлов без замедления моделирования.

Таблица 1. Основные характеристики Mininet, Packet Tracer, GNS3

|  |  |  |  |
| --- | --- | --- | --- |
|  | Mininet | Packet Tracer | GNS3 |
| Бесплатное программное обеспечение | Да | Да | Да |
| Открытый исходный код | Да | - | Да |
| В свободном доступе | Да | - | Да |
| Поддержка Windows | - | Да | Да |
| Поддержка Linux | Да | Да | Да |
| Симулятор | - | Да | Да |
| Эмулятор | Да | - | Да |
| Совместим с реальным контроллером | Да | - | - |
| Правильность результата | Зависит от ресурсов | - | Да |
| Поддержка Gui | Да | Да | - |

***5.2 Разработка экспериментального стенда***

Экспериментальный стенд создается последовательным выполнением следующих шагов:

* Тестовый стенд запускается на персональном компьютере, характеристики которого зависят от масштабов исследуемой ПОС (Например, для стенда, на котором производились исследования и получены результаты, представленные в разделе 6, персональный компьютер снабжен центральным процессором Core i3-3110M 2,4 ГГц (4 ядра), ОЗУ 8Гб, Windows 10 64 bit.)
* В среде Mininet моделируется топология ПОС**:** задается общее количество узлов; формируется исследуемая топология, в которой конечные узлы и контроллер моделируемой ПОС могут выполнять функции как целевых узлов, так и источников трафика. (Например, для стенда, на котором производились исследования и получены результаты, была смоделирована топология single, состоящая из одного коммутатора и двух конечных узлов.)
* Коммутатор для взаимодействия с контроллером использует протокол OpenFlow.(с набором команд управления).
* С помощью программы Wireshark выполняется перехват и анализ пакетов передачи данных
* По результатам анализа дампа трафика строятся диаграммыобработки сообщений.

**6. Экспериментальные исследования**

***6.1 Построение диаграммы обмена сообщениями при работе протокола ARP***

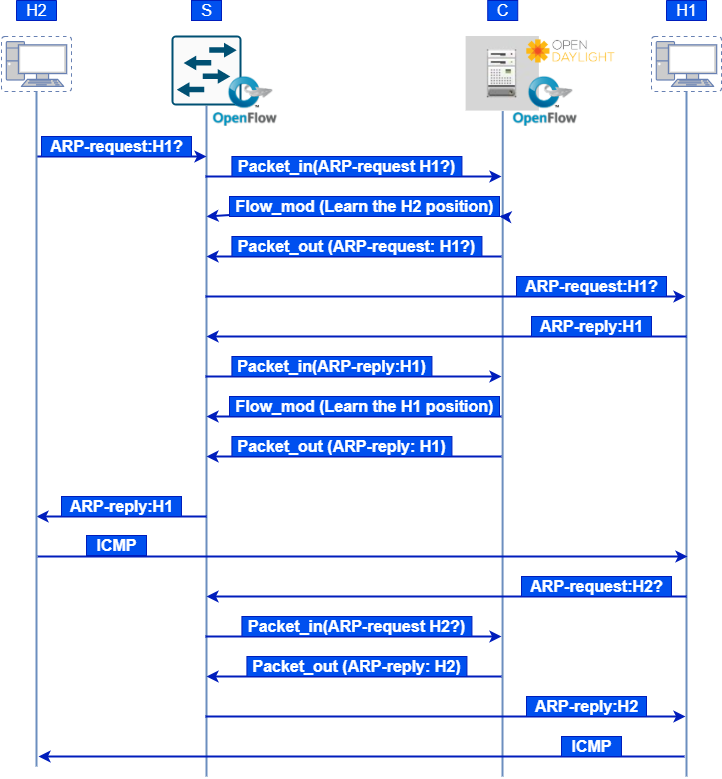
Для исследования служебного трафика в ПОС был использован тестовый стенд, с включением с среды Mininet и контроллера OpenDaylight. Схема стенда показана на рисунке 10. Тестовый стенд запускался на персональном компьютере, снабженном центральным процессором Core i3-3110M 2,4 ГГц (4 ядра), ОЗУ 8Гб, Windows 10 64 bit.

В среде Mininet была смоделирована топология single, состоящая из одного коммутатора и двух конечных узлов.

Коммутатор для взаимодействия с контроллером использовал протокол OpenFlow. С конечного узла H2 с помощью команды ping генерировались ICMP-запросы до конечного узла H1. С помощью программы Wireshark для перехвата и анализа пакетов передачи данных были проанализированы пакеты служебного OpenFlow-трафика, которые возникали в ходе данной процедуры определения адреса.

Рис. 10. Топология исследуемой ПОС

По результатам анализа дампа трафика построена диаграмма сообщений, представленная на рисунке 11.

  
Рис.11. Диаграмма обмена сообщениями при работе протокола ARP

Рассмотрим подробнее полученную диаграмму обмена сообщениями при работе протокола ARP, показанную на рисунке 11. Для успешного прохождения ICMP-запроса узел H2 должен узнать MAC-адрес узла H1, для чего отправляется ARP-запрос коммутатору S. Полученный ARP-запрос коммутатор инкапсулирует в OpenFlow-сообщение асинхронного типа Packet\_in и далее отправляет контроллеру, поскольку для входящего пакета на коммутаторе не найдено совпадений в наборе записей. Контроллер с помощью OpenFlow сообщения типа контроллер-коммутатор Flow\_mod даёт команду коммутатору запомнить расположение узла H1.

Следующим шагом контроллер рассылает ARP-запрос, инкапсулированный в сообщение Packet\_out, только на те порты коммутатора, к которым подключены узлы. Узел, обнаруживший совпадение искомого MAC-адреса с собственным, отправляет в сеть ARP-ответ, который ретранслируется коммутатором на контроллер.

Контроллер теперь знает расположение узла H1 и устанавливает соответствующее правило в таблице потоков коммутатора, после чего ARP-ответ отправляется узлу H2. После обновления ARP-таблицы, узел H2 может отправлять IP-пакеты узлу H1 (в реализованном эксперименте – ICMP-пакет). Узлу H1 потребуется отправить ответный ICMP-пакет узлу H2.

Особенность механизма определения адреса по протоколу ARP в ПОС состоит в том,, что заголовки Source IP/MAC перенаправляемого контроллером ARP-ответа будут иметь значения, принадлежащие IP и MAC самого ПОС-контроллера.

Таким образом, сгенерированный узлом H1 ARP-запрос будет передан коммутатором контроллеру в виде сообщения Packet\_in. Далее контроллер с помощью сообщения Packet\_out установит правило для коммутатора, который в свою очередь отправит ARP-ответ узлу H1. Узел H1 обновит свою ARP-таблицу и сможет отправить ICMP-пакет узлу H2.

Описанный выше процесс установления соединения для двух сетевых элементов можно условно представить в виде двух этапов:

Этап 1 – запрос на установления соединения от инициирующего сетевого элемента H2 до целевого узла H1;

Этап 2– ответ при установления соединения от целевого сетевого элемента H1 до инициировавшего узла H2.

***6.2. Оценка времени обработки служебного трафика протокола ARP в программно-определяемых сетях***

В локальных ТКС в процессе передачи пакетов данных от одного узла к другому важную роль играет процесс определения канальных адресов, осуществляемый в соответствии с протоколом определения адреса ARP, описанным в RFC 826 [20]. Однако механизм для обработки ARP-запросов и ARP-ответов в программно-определяемых сетях отличается от данного механизма в традиционных сетях IP. В ПОС при отсутствии на коммутаторах записей о направлении ARP-пакетов механизм определения адресов должен быть выполнен, в первую очередь, на контроллере — программно-аппаратной платформе, реализующей интеллектуальные функции сети.

В работе Internet Draft «Address Resolution Delay in SDN» [18], являющейся рабочим документом Инженерного совета Интернета (IETF) и опубликованной в октябре 2014 года, определяются две метрики, которые можно применить в качестве критерия оценки эффективности функционирования для характеристики затрачиваемого времени механизмом определения адреса:

1) Address Resolution Delay No Forwarding Flow Registrations (ARDNFFR) — время определения адреса без установки потока для передачи пакета на коммутаторе;

2) Address Resolution Delay Forwarding Flow Registrations (ARDFFR) — время определения адреса с установкой потока для передачи пакета на коммутаторе.

Дадим определения этим величинам, основываясь на топологии сети, представленной на рисунке 13, где С — контроллер; S — коммутатор; H1 и H2 — узлы в сети.

  
Рис.12. Логическая схема сети

Определение 1: рассмотрим задержку ARDNFFR. Пусть узел *H2* отправляет первый бит пакета ARP-запроса узлу *H1* в момент времени *TH2*, тогда H2 принимает первый бит ARP-ответа от *H1* в интервал момент времени [*TH2 + dτNF ]* при отсутствии требуемой ARP-записи в таблице потоков (кэше) коммутатора. Получаем, что задержка ARDNFFR есть величина dτNF. Единицами измерения данного параметра являются миллисекунды. За эталонное значение принимается время первого прохождения пакета по сети при отсутствии установленных потоков на коммутаторах для ARP-пакетов [18].

С учетом определения 1 и топологии сети, представленной на рисунке 12, предложим формулу для оценки метрики задержки при отсутствии записи в таблице потоков коммутатора:

*dτNF = L + CNF + SNF*, (1)

где *L* — время передачи по в каналам связи; *CNF* — время обработки пакета контроллером; *SNF* — время коммутатором без установленных правил.

Определение 2: рассмотрим задержку ARDFFR. Пусть узел *H2* отправляет первый бит пакета ARP-запроса узлу *H1* в момент времени *TH2*, тогда H2 принимает первый бит ARP-ответа от *H1* в момент времени *TH2 + dτ* при наличии требуемой ARP-записи в таблице потоков (кэше) коммутатора. Получаем, что задержка ARDFFR есть величина *dτ.* Единицами измерения данного параметра являются миллисекунды. За эталонное значение принимается время первого прохождения пакета по сети при наличии установленных потоков на коммутаторах для ARP-пакетов [18].

С учетом определения 2 и топологии сети, представленной на рисунке 12, предложим формулу для оценки метрики задержки при наличии записи в таблице потоков коммутатора:

*dτ = L + S*, (2)

где *L* — время передачи по каналам связи; S — время обработки коммутатором с установленными правилами.

Исходя из определений 1 и 2, а также формул (1), (2) можно сделать вывод, что затрачиваемое время механизма определения адресов в ПОС будет превышать значения затрачиваемого времени в аналогичной традиционной сети с коммутацией пакетов только в том случае, когда на коммутаторе отсутствует запись в таблице потоков.

Применение *dτ* вместо *dτNF* обусловлено следующими причинами:

– определение адреса с установленными потоками на коммутаторах является основной формой механизма определения адреса в ПОС;

– когда время жизни ARP-таблиц на коммутаторе заканчивается, то узел должен вновь инициировать механизм определения адреса, а значит и установку потока на коммутаторах.

Высокое значение параметров *dτ* вместо *dτNF* может свидетельствовать о проблемах на одном или более сегментах сети, а значит, данный параметр может быть использован для диагностики программно-определяемой сети в целом.

Рассмотрим возможность использования двух перечисленных параметров для оценки программно-определяемых сетей.

Введем следующие обозначения:

*H1, H2* — сетевые узлы;

*C0* — контроллер;

*Ti* — временной интервал, в течении которого на коммутаторах существуют записи о потоках, с;

*T0, Tf* — временные границы рассматриваемого интервала, с;

λ — интенсивность потока, 1/с.

В [18] выдвинуто предположение, что поток ARP-пакетов можно рассматривать в качестве Пуассоновского. Имея матожидание значений *T0, Tf* и λ, можно описать псевдослучайный Пуассоновский процесс на временном интервале *[T0, Tf]* с интенсивностью поступления пакетов λ. В каждый момент времени Ti значением задержки для первого процесса механизма определения адреса является параметр *dτNF,* а для последующих процессов механизма определения адреса — значения параметра *dτ.* Далее можно получить значение параметров *dτNF* и *dτ* на промежутке от *T0* до *Tf* c интервалом *Ti*.

Согласно [2], поток событий — последовательность событий, происходящих одно за другим в случайные моменты времени.

Пуассоновский поток — это поток, обладающий двумя свойствами — ординарностью и отсутствием последействия.

Поток называется ординарным, если для малого интервала Δt выполняется условие

*P1(t, Δt) >> P>1(t, Δt),* (3)

где *P1(t, Δt)* — вероятность того, что за Δt произойдёт одно событие, *P>1(t, Δt)* — вероятность того, что за *Δt* произойдёт более одного события.

Поток можно считать ординарным, если за малый промежуток времени может произойти не более одного события или ни одного события, вероятность чего обозначим *P0(t, Δt).* Для любого Δt справедливо

*P0(t, Δt) + P1(t, Δt) + P>1(t, Δt) = 1,* (4)

так как составляющие формулы (4) определяют полную группу несовместных событий.

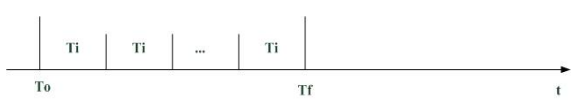
Для ординарного потока

*P0(t, Δt) + P1(t, Δt) ≈ 1*, (5)

потому что P>1(t, Δt) = 0(Δt), где 0(Δt) — величина, порядок малости которой выше, чем Δt [2], т. е.

(6)

Рассмотрим интервал *[T0, Tf] = mTi,* представленный на рисунке 13.

  
Рис.13. Рассматриваемый интервал *[T0, Tf]*

Отсутствие последействия — свойство, когда для двух неперекрывающихся интервалов времени число событий, попадающих в один интервал, не зависит от того, сколько событий попало в другой [2].

Воспользовавшись формулой распределения Пуассона для оценки вероятности возникновения k-запросов от коммутатора к контроллеру на установление правила на временном интервале *[T0, Tf] = mTi*, получим:

. (7)

Тогда вероятность возникновения n-событий в режиме функционирования с установленными правилами на коммутаторе будет определяться формулой

*.* (8)

Таким образом, согласно [18], общее время D, возникающее при функционировании механизма определения адреса может быть вычислена по соотношению

, (9)

где и — число временных интервалов, к которым применимы параметры задержки и dτ соответственно.

Из полученных результатов исследования можно сделать ряд выводов:

1. Если величина *Ti* стремится к бесконечности, то параметр *dτ* будет преобладать над . В противном случае может практически не быть *dτ* .

2. Если величина λ стремится к бесконечности, то параметр ничтожно мал на фоне *dτ* поскольку только одно из всего множества событий на интервале *Ti* будет соответствовать режиму функционирования сети без установленных потоков на коммутаторах.

В противном случае, если λ слишком мала, вероятность возникновения ARP-запросов на интервале *[T0, Tf] = mTi* может стремиться к нулю.

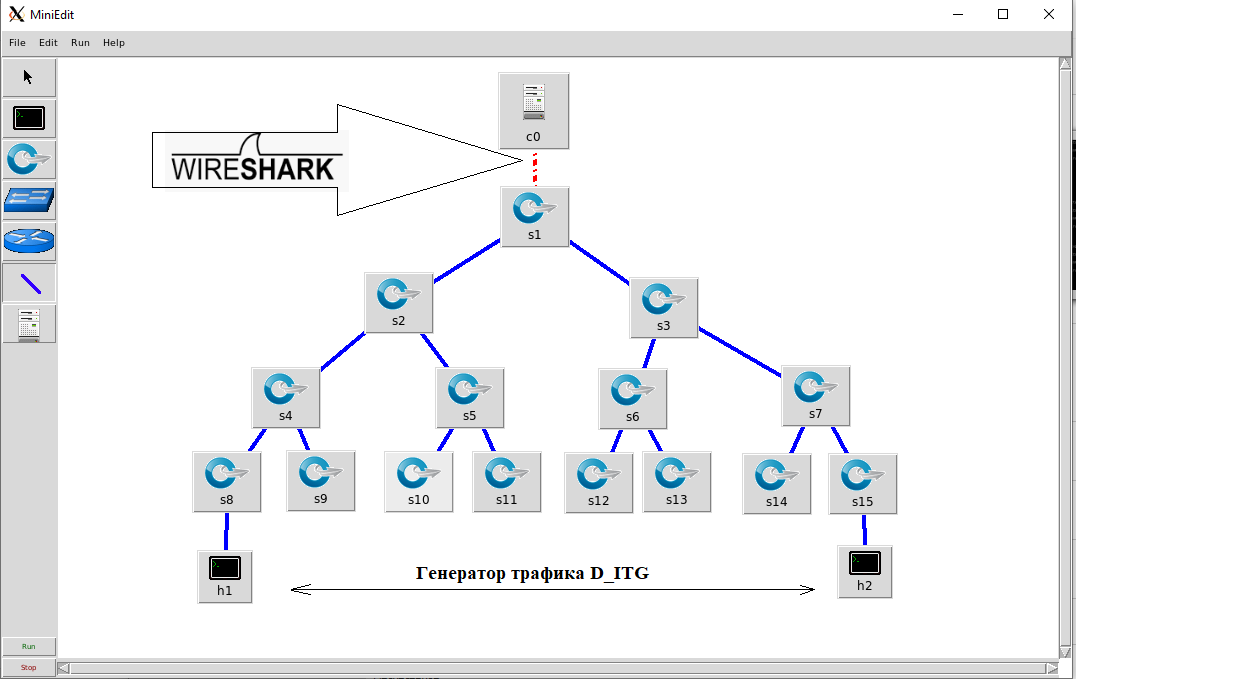
Полученные формулы оценки времени в программно-определяемых сетях представляют собой *математическую модель*, которая имеет практическую ценность, поскольку позволяют сетевому инженеру получить конкретные оценочные значения времени обработки пакетов на каждом из устройств сети.

В случае если затрачиваемое время обработки на контроллере весьма велико, системный администратор может сделать вывод о необходимости оптимизации работы ПОС-контроллера либо путем перенастройки установленных правил, либо путем модернизации аппаратной части.

***6.3******Экспериментальное исследование зависимости общего времени передачи служебного трафика от времени обработки контроллером***

Задача измерения времени обработки пакетов при отсутствии записи в таблице OpenFlow коммутатора представляется наиболее интересной для исследования.

Для выявления зависимости между временем передачи служебного трафика и времени обработки контроллером в сети, функционирующей с использованием протокола OpenFlow в среде Mininet, смоделирована древовидная топология, показанная на рисунке 14. Число коммутаторов последовательно задавалось равным 3, 7, 15 и 31. Выполнялся запуск генератор трафика D-ITG с узла H1 до H2, после чего с использованием Wireshark выполнялся перехват и анализ трафика и высчитывалось время dτNF, время обработки коммутатором SNF и время обработки контроллером СNF , представленные в формуле (1). При этом время обработки коммутатора SNF считалась как среднее арифметическое по всем OpenFlow - коммутаторам.

Рис.14. Исследуемая топология в эксперименте

Результаты измерений, выполненных в ходе тестирования, занесены в таблицу 2.

Таблица 2. Экспериментальные значения задержек

|  |  |  |  |
| --- | --- | --- | --- |
| Количество коммутаторов в сети, шт. | Задержка dτNF, мс | Задержка SNF, мс | Задержка СNF, мс |
| 3 | 0,997 | 0,019 | 0,471 |
| 7 | 1,234 | 0,025 | 0,747 |
| 15 | 1,636 | 0,044 | 0,887 |
| 31 | 2,399 | 0,188 | 1,146 |

По результатам проведенного исследования был построен график, показанный на рисунке 15.

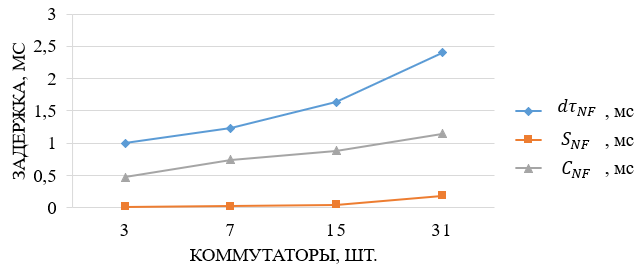
  
**количество коммутаторов**

Рис.15. Зависимость задержек от количества коммутаторов в сети

Исходя из характера кривых на графике, была выдвинута гипотеза о наличии зависимости (корреляции) между исследуемыми параметрами, задержкой передачи служебного трафика и задержкой контроллера в сети. Для проверки этого предположения были осуществлен расчет коэффициентов корреляции попарно для экспериментальных массивов значений dτNF и СNF, dτNF и SNF согласно следующему уравнению:

. (10)

Полученные значения *r(dτNF, СNF)* = 0,935, *r(dτNF, SNF)* = 0,951 указывают на наличие положительной линейной корреляции между исследуемыми параметрами, т.е. с ростом/уменьшением значения одного из этих параметров будет расти/уменьшаться и другой. Таким образом, показано, что снижение величины задержки, вносимой ПОС-контроллером, приведёт к снижению общего времени передачи служебного трафика ПОС.

***6.4 Выводы***

Разработан экспериментальный стенд и экспериментально получен алгоритм обработки служебного трафика протокола OpenFlow, контроллером программно-определяемой сети.

Разработана модель задержки служебного трафика протокола OpenFlow, имеющая практическую ценность, поскольку позволяют сетевому инженеру получить конкретные значения времени обработки пакетов на каждом из устройств сети. В случае если затрачиваемое время обработки на контроллере весьма велико, то системный администратор может сделать вывод о необходимости оптимизации работы ПОС-контроллера либо путем перенастройки установленных правил, либо путем модернизации аппаратной части.

Экспериментальным путем выявлена зависимость задержки обработки служебного трафика от задержки контроллера программно-определяемой сети. Показано, что снижение величины задержки, вносимой ПОС-контроллером, приведёт к снижению общего времени передачи служебного трафика ПОС.

**Заключение**

Для освоения программно-определяемой сети важную роль играет практический опыт. В этой работе описан процесс создания экспериментального исследовательского стенда для освоения и получения практического опыта в программно-определяемой сети. Экспериментальный стенд включает в себя эмулятор компьютерной сети Mininet, контроллер OpenDayLight, программу для перехвата и анализа сетевого трафика Wireshark и программу для генерации трафика D-ITG, которая позволяет создавать поток/потоки данных разной сложности.

Экспериментально установлен алгоритм обработки служебных запросов в ПОС и представлена модель задержки обработки служебного трафика в сетях, базирующихся на протоколе OpenFlow, показан её вывод на основе принципа работы протокола ARP.

На основе исследования полученной модели можно сделать вывод о том, что существует линейная зависимость между временем передачи служебного трафика dτNF, составляющими ее задержкой коммутатора SNF, задержкой ПОС-контроллера CNF. В случае высокого уровня задержки служебного трафика ее можно понизить путем снижения задержки коммутаторов, либо контроллера. Задержка, вносимая коммутаторами, определяется такими факторами, как производительность коммутационной матрицы, размером буфера и наличием очередей пакетов на интерфейсах.

Задержка, вносимая контроллером, зависит как от производительности его аппаратной платформы, так и от эффективности оптимизации программного кода СОС контроллера и приложений. Для снижения задержки контроллера рекомендуется выбирать аппаратную платформу сервера с максимальной производительностью и оптимизировать работу приложений, выполняющих функции управления трафиком, что в свою очередь может привести к снижению пакетной нагрузки на систему.

В результате исследований были получены новые научные результаты.

Разработана модель времени обработки служебного трафика программно-определяемых сетей на базе протокола OpenFlow для обеспечения их эффективного функционирования.

Выявлена зависимость общего времени обработки служебного трафика от времени обработки служебного трафика контроллером программно-определяемой сети.

**Литература**

1. Егоров, В.Б. Системы и средства информатики. Некоторые вопросы практической реализации SDN / В.Б. Егоров. - Москва: ТОРУС ПРЕСС, 2016. – 120 с.

2. Карташевский, В.Г. Основы теории массового обслуживания: учебник для вузов / В.Г. Карташевский. – Москва: Горячая линия-Телеком, 2013. - 130 с.

3. Колечкин, А.О. Программное обеспечение для тестирования контроллеров программно-конфигурируемых сетей. Материалы Девятнадцатой международной научной конференции «Распределенные компьютерные и телекоммуникационные сети: управление, вычисление, связь (DCCN-2016)» / А.О. Колечкин, А.Г.Владыко — Москва: РУДН, 2016. — 263 с.

4. Смелянский, Р.Л. Программно-конфигурируемые сети. URL: <https://www.osp.ru/os/2012/09/13032491/> (Дата обращения: 01.05.2020)

5. Braun, W. Software-defined networking using OpenFlow: protocols, applications and architectural design choices. URL: <https://pdfs.semanticscholar.org/2f7b/c914bad9cde16a2ca52e8470bc2b7a91cb43.pdf?_ga=2.108245259.807113061.1588075318-1160963201.1588075318> (Дата обращения: 06.05.2020)

6. Cbench. URL: <https://github.com/mininet/oflops/tree/master/cbench> (Дата обращения: 08.05.2020)

7. Channegowda, M. Software-defined optical networks technology and infrastructure: enabling software-defined optical network operations. J. Opt. Commun. Netw / M. Channegowda, R. Nejabati, D. Simeonidou. - USA:Institute of Electrical and Electronics Engineers Inc., 2014. – A274 - A282

8. Cisco Router Architecture. URL:  
<https://www.cisco.com/networkers/nw99_pres/601.pdf> (Дата обращения: 20.04.2020)

9. Harai, H. Optical packet and circuit integrated networks and software defined networking extension. J. Lightwave Technol / H. Harai, H. Furukawa, K. Fujikawa, T. Miyazawa, N. Wada, 2020. - 2751–2759

10. Hu, F. A survey on software-defined network (SDN) and OpenFlow: from concept to implementation. IEEE Commun. Surv. Tutor. / F. Hu, Q. Hao, K. Bao, 2020. - 2181–2206

11. Improving network management with software defined networking, 2013. URL: <http://www.princeton.edu/~hyojoonk/publication/SDN_ieeemagazine_Kim.pdf> (Дата обращения: 04.05.2020)

12. Mininet. URL: <http://mininet.org/> (Дата обращения: 07.05.2020)

13. OpenDaylight. URL: <http://www.opendaylight.org/> (Дата обращения: 10.05.2020)

14. ONF TS-001 OpenFlow Switch Specification version 1.0.0 (wire protocol 0x01). URL:

<https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-specv1.0.0.pdf> (Дата обращения: 04.05.2020)

15. ONF TS-006 OpenFlow Switch Specification version 1.3.0 (wire protocol 0x04). URL: <https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wpcontent/uploads/2014/10/openflow-spec-v1.3.0.pdf> (Дата обращения: 04.05.2020)

16. OpenFlow Controller Benchmarking Tool. URL: <https://github.com/No6things/ofc-benchmark> (Дата обращения: 06.05.2020)

17. Ortiz, S. Software-defined networking: on the verge of a breakthrough? IEEE Comput. / S. Ortiz, 2020. – 12 с.

18. Internet-Draft Address Resolution Delay in SDN. URL: <https://tools.ietf.org/html/draftpan-ippm-sdn-addr-resolv-perf-00> (Дата обращения: 30.04.2020)

19. RFC 7426 Software-Defined Networking (SDN): Layers and Architecture Terminology. URL: <https://tools.ietf.org/html/rfc7426> (Дата обращения: 20.04.2020)

20. RFC 826 An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses. URL: <https://tools.ietf.org/html/rfc826> (Дата обращения: 07.05.2020)

21. SDNCentral, LLC, Special Report: OpenFlow and SDN – State of the Union. URL:<https://www.opennetworking.org/images/stories/downloads/sdn-resources/special-reports/Special-Report-OpenFlow-and-SDN-State-of-the-Union-B.pdf> (Дата обращения: 20.04.2020)

22. Tootoonchian, A. On controller performance in software-defined networks. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE’12) / A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, R. Sherwood - USA: Hot-ICE’12, 2020. - 10 с.

23. wiki.opendaylight.org. URL: <https://wiki.opendaylight.org/view/Release/Hydrogen/Service_Provider/User_Guid> (Дата обращения: 25.04.2020)

24. Xia, W. A survey on software-defined networking. IEEE Commun. Surv. Tutor. / W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, 2020. - 51 с.