ФЕДЕРАЛЬНОЕАГЕНТСТВОСВЯЗИ

ФЕДЕРАЛЬНОЕГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА»
(СПБГУТ)

Б.С. Гольдштейн, В.С. Елагин, А.Е. Селиванов

ПРОГРАММНО-КОНФИГУРИРУЕМЫЕ СЕТИ (SDN). ПРОТОКОЛ OPENFLOW

Учебное пособие

СПб ГУТ)))

САНКТ-ПЕТЕРБУРГ
2017

УДК 94(075.8)
ББК 63.3я73
Г44

Рецензент
доктор технических наук, профессор *Н.А. Соколов* (ЛОНИИС)

Рекомендовано к печати
редакционно-издательским советом СПбГУТ

Гольдштейн Б.С., Елагин В.С., Селиванов А.Е..
Г44 Программно-конфигурируемые сети (SDN). Протокол OpenFlow: учебное пособие / Б.С. Гольдштейн, В.С. Елагин, А.Е. Селиванов; СПбГУТ. – СПб., 2017. – 76 с.

Содержит учебный материал по технологическим основам реализации технологии SDN и архитектуры построения этих сетей. Материал учебного пособия представлен в виде теоретической и практической частей, включает в себя планы проведения адаптируемого к слушателю интерактивного обучения, практических и лабораторных занятий.

Предназначено для студентов профильных специальностей: 11.03.02, 11.04.02 «Инфокоммуникационные технологии и системы связи», а также специалистам работающих на сетях связи различного назначения.

УДК 94(075.8)
ББК 63.3я73

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

API – Application Programming Interface (интерфейс программирования приложений)

BSS – Business Support System (система поддержки бизнеса)

CPE – Customer Premise Equipment (оборудование на стороне пользователя)

CPU – Central Processing Unit (центральный процессор)

CSP – Communication Service Provider (провайдер телекоммуникационных услуг)

DHCP – Dynamic Host Configuration Protocol (протокол динамической маршрутизации)

E2E – end-to-end (из конца в конец, сквозная услуга)

EMS – Element Management System (система управления элементами)

ETSI – European Telecommunications Standardization Institute (Европейский Институт Стандартизации в Области Телекоммуникаций)

GRE – Generic Routing Encapsulation (общая инкапсуляция маршрутов)

IETF – Internet Engineering Task Force (Инженерный Совет Интернета)

IP – Internet Protocol

IRTF – Internet Research Task Force (исследовательская группа интернет-технологий)

LTE – Long Term Evolution

MPLS – Multiprotocol Label Switching (многопротокольная коммутация по меткам)

NAT – Network Address Translation (преобразование сетевых адресов)

NF – Network Function (сетевая функция)

NFV – Network Functions Virtualization (виртуализация сетевых функций)

NFVI – Network Functions Virtualisation Infrastructure (инфраструктура виртуализации сетевых функций)

NFVI-PoP – Network Functions Virtualisation Infrastructure Point of Presence (точка присутствия инфраструктуры виртуализации сетевых функций)

NIC – Network Interface Card (сетевая карта)

NMS – Network Management System (система управления сетью)

ONF – Open Networking Foundation

OSS – Operation Support System (система поддержки эксплуатации)

QoS – Quality of Service (качество обслуживания)

REST – Representational State Transfer (передача репрезентативного состояния)

SDN – Software Defined Networking (программно-конфигурируемая сеть)

VLAN – Virtual Local Area Network (виртуальная локальная сеть)

VNF – Virtual Network Function (виртуальная сетевая функция)

VoIP – Voice over IP

VXLAN – Virtual Extensible LAN

МСЭ-Т – Международный Союз Электросвязи-Телекоммуникации

ПКС – программно-конфигурируемая сеть

ПО – программное обеспечение

ЦОД – центр обработки данных

ЧНН – час наибольшей нагрузки

1. ВВЕДЕНИЕ В ТЕХНОЛОГИЮ SDN

1.1 Определение и история развития SDN

В современном мире, при таком разнообразии поставщиков телекоммуникационного оборудования, прогресс, модификация и сложность различных технологий стремятся к росту количества устройств и объемов передаваемых данных. Классическая архитектура сети, основы которой закладывались в конце 60-х годов, в связи с быстрым ростом многообразия, сложности и важности решаемых задач, устарела и не всегда способна эффективно реагировать на новые потребности рынка. Чтобы полностью преобразовать исторически устоявшуюся архитектуру, необходимо затратить не малые средства, а модернизация отдельных элементов сети не может длиться бесконечно.

Сегодня инфокоммуникационные сети, несмотря на повсеместное использование и масштабное развитие, встречают на своем пути множество проблем:

- Рост количества пользователей и трафика приводит к перегрузке сетевых устройств. На сегодняшний день, пользователь глобальной сети генерирует больше трафика, чем вся всемирная паутина 30 лет назад. В 2014 году интернет-трафик вырос, по сравнению с 1984 годом, в 2.7 миллиардов раз (Cisco).
- Огромное количество протоколов (более 700) и их стеков, число которых с каждым днем только увеличивается;
- Устаревшие протоколы, особенно протоколы транспортного и сетевого уровней;
- Традиционные сети проприетарны, ограничены для исследований и практически любых изменений. Интерфейс настройки сетевого оборудования, иногда зависит даже от модели у одного и того же производителя – следовательно, специалисты в области сетевых технологий

должны уметь анализировать системы разных вендоров и работать с большим количеством протоколов.

- Оборудование разных производителей довольно часто может конфликтовать между собой, несмотря на универсальность сетевых протоколов;
- Настройка сети с большим количеством узлов требует много времени, а управление такими сетями осуществляется путем конфигурирования их элементов через специализированные интерфейсы.
- Освоение системы оборудования конкретного производителя, требует переподготовки специалистов [1];

Таким образом, большое количество проблемных факторов ведет к усложнению сетевого оборудования и структуры самих информационных сетей. И, как следствие, удорожанию сетевого оборудования. В качестве рецепта устранения некоторых сетевых проблем, ряд экспертов называют переход к архитектуре программно-конфигурируемых сетей, которые позволяют перевести сетевые элементы под контроль настраиваемого ПО (контроллера). Идеология SDN позволяет упростить сетевые устройства до предела, сделать их «слабыми», а значит и дешевыми.

Программно-конфигурируемые сети (SDN от англ.Software-DefinedNetworks) – это новая технология построения архитектуры компьютерных сетей, в основе которой лежит перенос функций управления (маршрутизаторов, коммутаторов) в отдельные программные приложения, которые функционируют на отдельном сервере. Таким образом производится физическое разделение уровней управления и передачи данных. Компоненты данной сети разделены на два класса – коммутаторы и контроллеры. Проприетарные традиционные коммутаторы не позволяют изменять и вносить желаемую функциональность. SDNдаёт такую возможность. Вся вычислительная нагрузка сосредоточена на контроллерах, которые решают задачи, связанные с прокладкой маршрутов и управлением передачи данных,

тем самым разгружаются все остальные сетевые устройства, оставляя им только функцию пересылки трафика. Интерфейсы общения между всеми центральными серверами и контроллерами – открытые, что позволяет попробовать свои идеи и разработки. Допустимость самостоятельно писать приложения для контроллера, предоставляет возможность использования некоторых функций, которые ранее были доступны только в качестве аппаратных решений.

На Рисунке 1 ETSI представила упрощенную лента времени: развитие технологии SDN

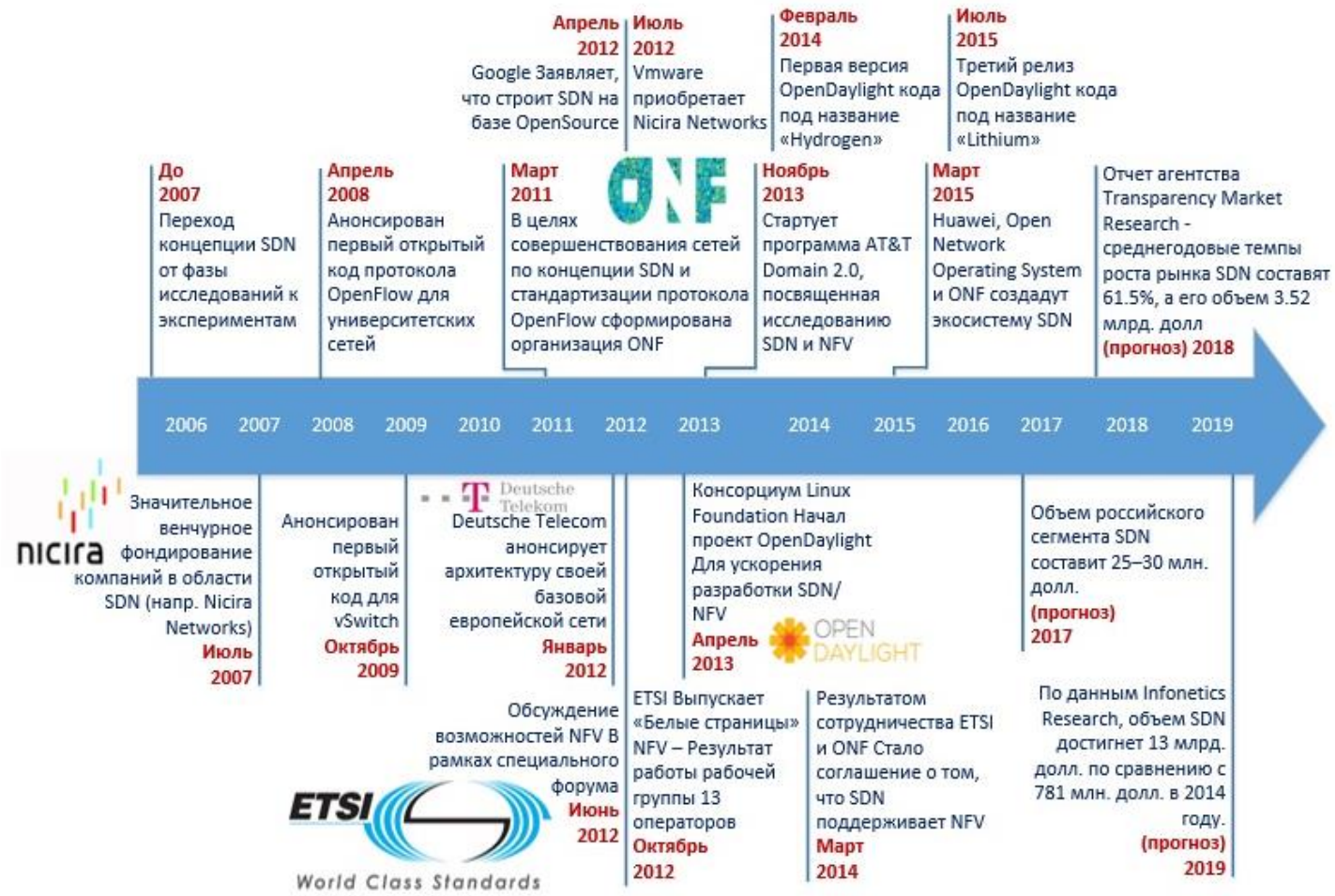


Рисунок 1. Лента времени: развитие технологии SDN (ETSI)

Сложно представить общую картину истории развития технологии программно-конфигурируемых сетей, если учитывать большое количество стартапов, «поглощений» и инвестиций на мировом рынке SDN, поэтому

лучше будет поэтапно рассмотреть самые яркие события на пути становления концепции.

1993-2011

Сама концепция новой сетевой архитектуры, которая позднее получила название SDN, появилась еще в далеком 1993 году, но возможность успешно её реализовать появилась только в 2007 году, когда американскими преподавателями (Мартином Касадо, Ником МакКеоном, Скоттом Шенкером) был разработан новый открытый стандарт общения компьютерных сетей – OpenFlow. Они понимали, что современная сетевая архитектура устарела и, скорей всего, в ближайшем будущем она не сможет обеспечить требуемого качества обслуживания. К сожалению, принципиальные изменения в сетевом оборудовании и протоколах весьма трудное дело и невозможно без привлечения производителя. Переход с оборудования одного вендора на оборудование другого – большая головная боль и огромные затраты ресурсов. В этой ситуации удачно «выстрелила» идея преподавателей разделить уровни управления и передачи данных. Проблему зависимости от оборудования одного производителя Касадо предложил решить с помощью нового сетевого протоколасоткрытым кодом, который позволяет уйти от «ручного» управления сетью (OpenFlow).Научное сообщество благоприятно приняла предложенную концепцию. Позже был основан исследовательский центр OpenNetworkingResearchCenter. Программное обеспечение и все исследовательские программы в рамках ПКС-подхода являются открытыми. Параллельно с этим Кассадо, МакКиннон и Шенкер создают коммерческий проект Nicira, это был первый успешный коммерческий проект в области ПКС который занимался созданием собственной платформы в сфере виртуализации сетей — networkvirtualizationplatform (NPV).Стартап был продан в июле 2012 года за рекордные $1,26 млрд. при венчурных инвестициях в $50 млн., его можно

считать точкой отсчета для глобальной ПКС-индустрии. После сделки с Nicira на рынке ожидалось, что следующим стартапом, который купит один из гигантов ИТ-индустрии, будет BigSwitchNetworks. Компания, выпустившая контроллер Floodlight и средство тестирования OFTest, была основана экс-профессором Стэнфорда Гвидо Аппензеллером и бывшим сотрудником Cisco Кайлом Форстером. Основанный в 2010 году стартап за 2,5 года собрал почти $50 млн. венчурных инвестиций. Инвесторы полагают, что решения BigSwitch способны изменить образ современных центров обработки данных и того, как в них происходит процесс сетевых коммуникаций. Озвученные выше стартапы являются далеко не единственными, кто занимался разработкой и исследованиями в этой тематике. Множествокомпаний (Vyatta, PluribusNetworks, Plexxi, LineRateSystems, ContrailSystems) внесли свои решения в разработку платформ и программного обеспечения программно-конфигурируемых сетей. Весной 2011 года был сформирован консорциум OpenNetworkingFoundation (ONF) с целью развития технологий SDN в целом и протокола OpenFlow в частности. Сегодня членами ONF являются практически все основные поставщики сетевого оборудования (более 130 членов).

Январь 2012

Технологической новинкой заинтересовалась одна из ведущих ISP (InternetServiceProvider) компаний мира – Google. Компания начала тестировать коды OpenFlow в 2009 году еще до выхода официальной версии протокола. Для того чтобы быть глобальной пользовательской базой с высокой скоростью и доступностью, большие объемы данных должны быть перемещены из одного региона в другой. Эту задачу и должна была решить интеграция сетей SDN в инфраструктуру сети Google. Компания разработала свой собственный высокопроизводительный коммутатор (10 Гбит/с) и открытый стек маршрутизации с поддержкой протокола OpenFlow 1.0. В 2010 году Google,объединив двенадцать центров обработки данных (G-scale сеть),

начала внедрение технологии программно-коммутируемых сетей и на сегодняшний день полностью перестроила свою глобальную сеть, адаптировав её под OpenFlow. Благодаря этому была улучшена управляемость, производительность, использование и рентабельность в глобальной сети.

Апрель 2013

LinuxFoundation – открытый консорциум участником которого может быть любой человек или любая компания. Организация является гидом для компаний, которые хотят строить свои решения на базе OpenSource, и предоставляет пространство для обсуждения насущных вопросов техническим сообществам, разработчикам приложений, промышленным заказчикам и конечным пользователям. В апреле 2013 консорциум начал работу над новым проектом OpenDaylight, в рамках которого ряд ведущих производителей отрасли объединили свои усилия для создания единой открытой платформы для организации работы программно-конфигурируемых сетей. В итоге совместной работы был подготовлен открытый фреймворк, на базе которого заинтересованные компании и отдельные разработчики могут создавать готовые продукты и сервисы для SDN-сетей. Например, наиболее востребованным применением SDN являются облачные системы, в которых виртуализация сетевого уровня позволяет динамически предоставлять клиентам желаемые виртуальные сетевые ресурсы. Используя уже готовый унифицированный фреймворк, поддерживаемый всеми производителями оборудования, появится возможность существенно упростить и ускорить создание конечных приложений и сервисов для SDN, основанных на уже сформированном едином высокоуровневом стеке, не привязанном к решениям отдельного вендора. Первый выпуск OpenDaylight кода под названием «Hydrogen» включает реализацию контроллера для SDN-сети, систему для создания виртуальных сетей, набор плагинов для поддержки различных протоколов и набор улучшений для виртуальных коммутаторов.

Таким образом OpenDaylight выступает в роли полнофункциональной SDN-платформы, пригодной для прямого развёртывания в различных сетевых окружениях, без необходимости использования каких-либо дополнительных компонентов. Расширения и дополнительные возможности поставляются в форме плагинов. Названия релизов соответствуют элементам таблицы Менделеева: в феврале 2013 года вышел Hydrogen (водород), в сентябре 2014 года – Helium (гелий), третий релиз вышел в 2015 году под названием Lithium (литий).

Ноябрь 2013

Программа SupplierDomain 2.0, которую AT&T ввела в ноябре 2013, является релизом начальной версии Domain 1.0. Основной стратегией является переход крупнейших операторов связи и телекоммуникационных компаний к крупномасштабному использованию SDN и виртуализации сетевых функций (networkfunctionvirtualization, NFV). Задача компаний к 2020 году трансформировать 75 процентов своей сетевой архитектуры под архитектуру SDN. К концу 2014 года 40 процентов приложенийAT&T уже было переведено на новую структуру сети, что позволило на 50 процентов увеличить эффективность пропускной способности по сравнению с традиционной архитектурой

Март 2015

OpenNetworkOperatingSystem(ONOS) – это проект, который был основан в ноябре 2014 года некоммерческой группой OpenNetworkingLab (ON.Lab), которая в свою очередь состояла из пионеров в разработке и стандартизации технологии SDN. Проект занимается разработкой открытого исходного кода для SDN и виртуализации сетевых функций. Платформа имеет отличную архитектуру, ориентированную на операторские сети и сервисные требования, что делает её высокопроизводительной, надежной и безопасной. В марте 2015 года ONOS совместно с компанией Huawe и фондом ONF объявила о создании открытой инновационной экосистемы типа

SDN, причем приставка «эко» никак не связана с окружающей обстановкой. Целью сотрудничества было поставлено создание единой, открытой и программируемой сетевой архитектуры SDN, которая будет отвечать соответствующим потребностям операторов связи и принесет наибольшую выгоду. Сегодня ONOS является частью LinuxFoundation.

Прогноз к 2020 году

Осознав все преимущества, мобильные операторы и операторы широкополосного доступа начали делать инвестиции в технологию программно-конфигурируемых сетей, это позволяет увидеть детальный прогноз развития и сделать вывод о дальнейшем положении SNDна мировом рынке. Согласно отчету агентства ResearchandMarkets инвестиции сервис-провайдеров в программно-конфигурируемые сети и виртуализацию сетевых функций к концу 2020 года составят более $20 млн., а среднегодовой темп роста инвестиций между 2015 и 2020 годами составит 54 процента.

1.2 Стандартизация в SDN?

Технология SDN приковала огромное внимание не только со стороны поставщиков, но и со стороны бизнеса. В следствие этого, все большее количество компаний начинают проявлять интерес к использованию ПКС-сетей, что требует разработки системы однородных стандартов для открытого применения. На сегодняшний день существуют несколько некоммерческих организаций, рабочих групп и сообществ, которые предлагают свои определения и стандарты в этой области:

OpenNetworkingFoundation(ONF, фонд открытых сетевых технологий) - одна из первых некоммерческих организаций, целями которой было продвижение и стандартизация принципиально нового подхода к передаче данных.Определение программно-конфигурируемых сетей, предложенное консорциумом ONF:

- Динамичная, управляемая и адаптируемая сетевая архитектура, в которой разделены уровни управления сетью и передачи данных, что позволяет обеспечить программное управление сетью и абстрагирование/изоляцию (уровня) сетевой инфраструктуры от (уровня) приложений и сетевых услуг/сервисов.

Основной задачей ONF является продвижение стандарта OpenFlow, который позволяет осуществлять удаленное управление уровнем передачи данных. Стандарт OpenFlow – первый стандарт SDN. В настоящее время в ONF продолжается работа по анализу требований к SDN, развитию стандарта OpenFlow в соответствии с запросами, возникающими при коммерческом развертывании SDN, а кроме того, создаются новые стандарты в целях расширения возможностей SDN.
Структура ONFсостоит из технических сообществ, которые организованы в области, советы и группы. Каждая из областей заключает в себе определенный набор рабочих групп, которые отвечают на конкретные вопросы, связанные с программно-конфигурируемыми сетями (концепция SDN, архитектура, сертификация, программные платформы, программное обеспечение), и сотрудничают с мировыми экспертами в области разработки стандартовSDN и OpenFlow. Советы разрабатывают стратегию, контролируют операционное исполнение и технические направления организации. Группы работают помощниками в обеспечении бесперебойной работы консорциума, руководят и консультируют ONFдля достижения общих целей организации.

InternetEngineeringTaskForce(IETF, группа по решению задач проектирования Интернета) – открытое международное сообщество сетевых проектировщиков, операторов и провайдеров, которые заинтересованы исследованиями и эволюцией сети Интернет. Определение программно-конфигурируемых сетей, предложенное группойIETF:

- SDN — подход к построению сетей, обеспечивающий прямое управление ресурсами и сетями, а также их распределение за счет добавления собственных средств обработки, администрирования и программного управления посредством открытых сетевых интерфейсов и абстракции (абстрагирования, изоляции) уровня сети.

Архитектура функционирования сообщества состоит из нескольких рабочих групп, действия которых распределены согласно основным выделенным тематикам, таким как маршрутизация, транспорт, безопасность и пр.

Разработка аналитических и стандартизирующих документов IETF, относящихся к SDN, стартовала в конце 2012 г. и сейчас находится на начальных этапах. Следует отметить, что ONFи IETFопределяют два совершенно разных подхода к реализации SDN. ONF определяет OpenFlow как интерфейс между двумя уровнями и считает, что весь потенциал программно-конфигурируемых сетей можно раскрыть благодаря протоколу, который осуществляет перенос данных с одного уровня на другой. IETF предлагает свой элемент архитектуры сети SDN – PCE (PathComputationElement), задача которого вычислять маршруты LSP (LabelSwitchPath). Элемент может быть реализован как роутер, часть OSS-системы или как виртуальный элемент, поднятый в облачной среде. В целом принцип работы подхода IETF выглядит так: сетевой элемент при прокладке маршрута обращается к PCE, тот, зная всю сетевую топологию и статус сети, вычисляет оптимальный маршрут. Вычислив, PCE возвращает узлу данные для LSP, которые передаются по сети с помощью сигнального протокола. Исследования по SDN в IETF представляет исследовательская группа Software-DefinedNetworkingResearchGroup(SDNRG) – группа по SDN, сформированная в составе исследовательской группы интернет-технологий IRTF. Целью SDNRG является изучение различных аспектов SDN для определения, распространения и применения подходов к реализации

концепции в краткосрочной перспективе, а также выявление вопросов, требующих дальнейшего изучения. В частности, целевыми вопросами для РГ являются масштабируемость, абстракции, языки и парадигмы программирования, полезные при реализации SDN. В настоящий момент SDNRG представила документы, в которых рассматриваются вопросы сокращения объемов контрольной информации в сети SDN, уровни SDN и терминология архитектуры SDN, концепция SDN с точки зрения поставщика услуг, анализ кластеров контроллеров SDNв крупномасштабных сетях.

European Telecommunications Standards Institute (ETSI, европейский институт по стандартизации в области телекоммуникаций) – некоммерческая организация по разработке стандартов в области информационных и коммуникационных технологий, официально признанная Европейским союзом. Институт разрабатывает стандарты фиксированной, подвижной, радио, конвергентной связи, а также телевещания и интернет-технологий. В составе ETSI была выделена группа отраслевой спецификации (IndustrySpecificationGroup, ISG) по разработке концепции виртуализации сетевых функций (служб) (NetworkFunctionsVirtualisation, NFV). Концепция NFV предполагает замещение разнообразных сетевых устройств стандартизированными высокопроизводительными серверами, коммутаторами и системами хранения данных с реализацией сетевых функций (служб) программным обеспечением.

Предполагается, что ISGNFV разработает требования и архитектуру NFV, рассмотрит вопросы управления, оркестровки служб, архитектуры ПО, производительности и переносимости, надежности и устойчивости, безопасности и миграции к NFV. Концепция NFV сходна с концепцией SDN, однако на текущем, начальном, этапе работы степень их взаимного соотношения не является точно определенной. Согласно ETSI, концепция NFV в большой степени дополняет SDN, но концепции независимы, каждая из них может быть реализована отдельно. Некоторые считают, что

совместное использование технологий открывает широкие возможности для обеспечения связности инфраструктуры. Использование их создаст беспрецедентные условия для быстрого развертывания отказоустойчивой, ориентированной на сетевые приложения архитектуры с высоким уровнем безопасности и возможностью обеспечения гарантированного качества услуг. В январе 2015 года ISGпредставила 11 спецификаций, завершив, таким образом, первый этап работы.

International Telecommunication Union (ITU, международный союз электросвязи) – специализированное учреждение Организации Объединенных Наций в области информационно-коммуникационных технологий. Организация занимается распределением радиочастотного спектра и спутниковых орбит в глобальном масштабе, разрабатывает технические стандарты на международном уровне, обеспечивающие возможность эффективного присоединения сетей и технологий, и стремится улучшить доступ к ИКТ для недостаточно обслуживаемых сообществ всего мира. МСЭ-Т занимается исследованием технических, эксплуатационных, тарифных и других вопросов, выпускает рекомендации с целью стандартизации электросвязи на международном уровне. В ходе Всемирной ассамблеи МСЭ по стандартизации электросвязи в Дубае в 2012 г., где обсуждались вопросы SDN, было высказано согласованное мнение, что программно-коммутируемые сети коренным образом преобразуют отрасль электросвязи и ИКТ в ближайшие десятилетия, обеспечат отрасли многочисленные преимущества. В условиях быстро растущего интереса к SDN со стороны значительного количества компаний необходима система стандартов для широкого применения SDN.По результатам обсуждения тематики SDN была принята Резолюция 77 «Работа по стандартизации в области организации сетей с программируемыми параметрами в Секторе стандартизации электросвязи МСЭ», поручившая Исследовательской комиссии (ИК) 13 в следующем исследовательском периоде расширить и

ускорить работы в области архитектуры и требований к SDN, а также представить рекомендации Консультативной группе по стандартизации электросвязи (КГСЭ) относительно порядка рассмотрения вопросов, выходящих за рамки мандата ИК 13.КГСЭ поручено изучить проблему, рассмотреть вклады ИК 13 и других ИК и принять необходимые меры для активизации деятельности по стандартизации SDN в МСЭ-Т. Определение программно-конфигурируемых сетей, предложенное международным союзом электросвязи:

- SDN: Технология построения сетей, которая позволяет реализовать централизованный, программируемый уровень управления и изоляцию (абстракцию) уровня данных, при этом уровни управления и данных разделены, благодаря чему операторы сетей связи могут напрямую управлять своими виртуальными ресурсами и сетями.

В настоящее время исследованиями в области SDN в МСЭ-Т занимаются ИК 13 (архитектуры и функциональные требования к SDN) и ИК 11 (эталонные архитектуры сигнализации SDN, требования к сигнализации и протоколам SDN, включая протоколы взаимодействия, а также тестирование на соответствие и взаимодействие). Интерес к SDN проявляют ИК 15 (транспорт в SDN) и ИК 17 (безопасность в SDN).

Существует некоторое количество представителей, которые акцентируют свое внимание на решении частных вопросов построения и эксплуатации ПКС-сетей:

Optical Internetworking Forum (OIF, оптический межсетевой форум) – некоммерческая организация, основанная в 1998 году. Участники форума разрабатывают соглашения по реализации для оборудования оптических сетей, а также занимаются демонстрацией, тестированием и построением требований к SDNco стороны операторов и поставщиков услуг в части

транспортных сетей, проводят исследования на соотносимость архитектуры SDNс архитектурой оптических сетей с автоматической коммутацией.

BroadbandForum (BBF, форум широкополосных сетей) – некоммерческая организация, сосредоточенная на разработке спецификаций для«умных» и высокоскоростных широкополосных сетей, спектр которых простирается от DSL до технологий создания максимально эффективных и управляемых широкополосных сетей, объединяющих операторское и абонентское оборудование в единую IP-платформу предоставления услуг. В рамках рабочей группы «Инновационные услуги и рыночные требования» основан проект SD-113 – коммерческие требования и структура SDN в широкополосных телекоммуникационных сетях, на основе которого предполагается проведение исследований в области вариантов перехода к сетям SDN, включая способы внедрения функциональности и поддержки ПКС-сетей оборудованием при обновлении программного обеспечения.

Качественное решение проблем интеграции и стандартизации любой технологии невозможно без проведения испытаний разработанных решений в области исследования данной тематики. На рисунке 2 представлено общее распределение стандартизирующих организаций по направлениям разработки архитектуры SDN Испытание разрабатываемых стандартов и разработкой открытых решений SDN занимаются несколько организаций:

1) OpenDaylight — объединение отраслевых производителей, включая IBM, Juniper Networks, Cisco, Red Hat, VMware, Citrix, Ericsson, Microsoft, NEC, Big Switch Networks, Brocade Communications Systems. Целью проекта является создание единой открытой платформы SDN, разработка открытой структуры классов (фреймворка) в качестве основы для создания готовых продуктов и сервисов различными участниками рынка;

2) Open vSwitch — проект по разработке программного коммутатора SDN с открытым исходным кодом для применения в виртуализированной

сетевой среде. В проекте участвуют компании Citrix, Red Hat, Canonical, Oracle, FreeBSD Foundation, Nicira;

3) OpenStack — проект по разработке комплекса бесплатных открытых программных средств для создания облачной вычислительной инфраструктуры и хранилищ данных, включающий поддержку SDN. В проекте представлены компании AT&T, AMD, Brocade Communications Systems, Canonical, Cisco, Dell, EMC, Ericsson, Groupe Bull, HP, IBM, Inktank, Intel, NEC, Rackspace Hosting, Red Hat, SUSE Linux, VMware, Yahoo!

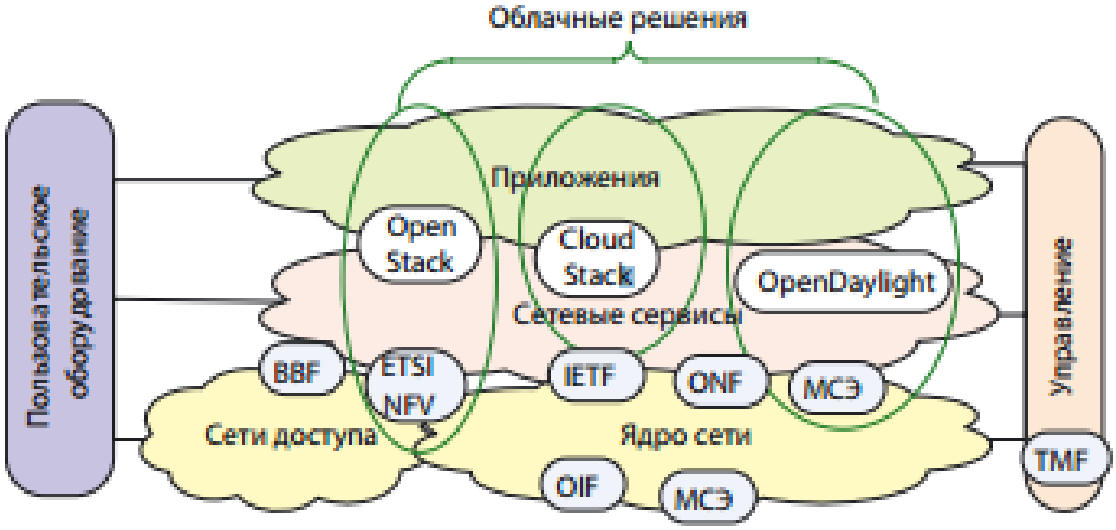


Рисунок 2. Распределение стандартизирующих организаций по направлениям разработки архитектуры SDN.

1.3 Область применения SDN

Основными областями применения SDN является коммутация, контроллеры, виртуализация облачных приложений и виртуализация средств безопасности сетевых решений. Отсюда, основные направления, где программно-коммутируемые сети пытаются найти свое место:

1) ЦОДы
2) облачные технологии
3) сети провайдеров
4) корпоративные сети

5) локальные сети (домашние)

6) безопасность

В локальных сетях модернизация будет довольно нескоро, но разработчики о такой возможности говорят. Если модернизация произойдет, то можно будет применять для настройки локальных сетей с управлением трафиком внутри них, также с возможностью для операторов удаленно настраивать такие сети.

К тому же, архитектура SDN работает в реальной сети сотовой связи и может использоваться в mesh-сетях.

Основные преимущества внедрения программно-коммутируемых сетей в компаниях со сложной ИТ-инфрастуктурой:

1) Стоимость. За счет снижения расходов на управление, уменьшается стоимость владения компьютерными сетями, и как следствие - увеличение прибыли.

2) Повышение производительности. Из-за того, что с коммутаторов снижаются нагрузки по обработке линии управления, программно-коммутируемая сеть дает возможность этим устройствам направить все свои ресурсы на ускорение перемещения трафика.

3) Реализация и тестирование новых сервисов. Программные средства программно-коммутируемых сетей позволяют администраторам добавлять новые функциональности к уже имеющейся сетевой архитектуре.

4) Администрирование. На централизованном контроллере программно-коммутируемой сети системный администратор может наблюдать сеть в едином представлении, что способствует повышению удобства управления, обеспечения безопасности и выполнения других задач.

5) Безопасность. Так как программно-коммутируемая сеть предоставляет возможность администратору четко видеть все потоки трафика, то ему будет гораздо легче замечать вторжения, определять

приоритеты различным типам трафика, разрабатывать правила реагирования сети при заторах и проблемах с оборудованием.

6) Облачные технологии. В облаках данные и приложения размещены на компьютерах, которые взаимодействуют по сети, и данная технология способна дать требуемый облакам уровень «интеллектуальности» сетей.

1.4 Архитектура программно-конфигурируемой сети

Существуют две независимые модели развития архитектуры SDN. Одна из них продвигается консорциумом ONF, другая – IETF. Кроме этого, некоторые открытые комьюнити пытаются разработать протоколы взаимодействия между NFV и SDN. Например, группой ETSI NFV ISG представлена модель MANO (Management And Orchestration), а консорциумом MEF (Metro Ethernet Forum) была разработана концепция LSO (Lifecycle Service Orchestration), которая является дополнением модели MANO и неким связующим звеном между вышеописанными архитектурами (ONF SDN и ETSI NFV MANO). Наибольшую популярность и экономические показатели имеет модель ONF на базе протокола OpenFlow. Консорциум продвигает трехуровневую модель, состоящую из уровней инфраструктуры, управления и приложений, если подробно рассмотреть информационные потоки в архитектуре SDN (рис. 3), то можно увидеть два направления обмена информацией. Первый между бизнес-приложениями конечных пользователей и уровнем управления, второй между уровнем физических сетевых устройств и опять же уровнем управления. Первый поток получил название «северный интерфейс» (northbound API), а второй «южный интерфейс» (southbound API). В качестве первого выступает протокол на основе REST API (REST – общие принципы организации взаимодействия приложения с сервером, API – интерфейс программирования приложений, состоящий из набора готовых кодов, предоставляемых приложением для

использования во внешних программных продуктах), а в качестве «южного интерфейса» протокол OpenFlow.

Рисунок 3. Трехуровневая модель SDN консорциума ONF

Концепция обладает свойством изолированности функций управления от функций форвардинга данных (disaggregation), в связи с этим, каждый из уровней модели может быть рассмотрен по отдельности, а не в качестве единой интегрированной системы.

1.4.1 Уровень приложений

Совокупность приложений, которые напрямую взаимодействуют с SDN контроллером, запрашивая необходимые ресурсы посредством API, и решают высокоуровневые задачи по управлению сетью. Примером таких приложений могут служить средства мониторинга, аналитики или бизнес-приложения. Например, конкретное бизнес-приложение Microsoft Lynk и его основная роль – изменение сети в режиме реального времени под текущие нужды

обслуживаемой программы (изменение Quality of Service или создание VPN тоннеля между двумя абонентами).

2.2.2 Уровень управления

Осуществляет низкоуровневое логически-централизованное управление, которое обеспечивает форвардинг трафика на инфраструктурном уровне с помощью открытого интерфейса OpenFlow. Уровень управления постоянно осуществляет мониторинг всей сети и способен управлять всеми сетевыми устройствами. Поэтому приложениям и сетевым политикам сеть представляется как единый логический коммутатор (рис.4).

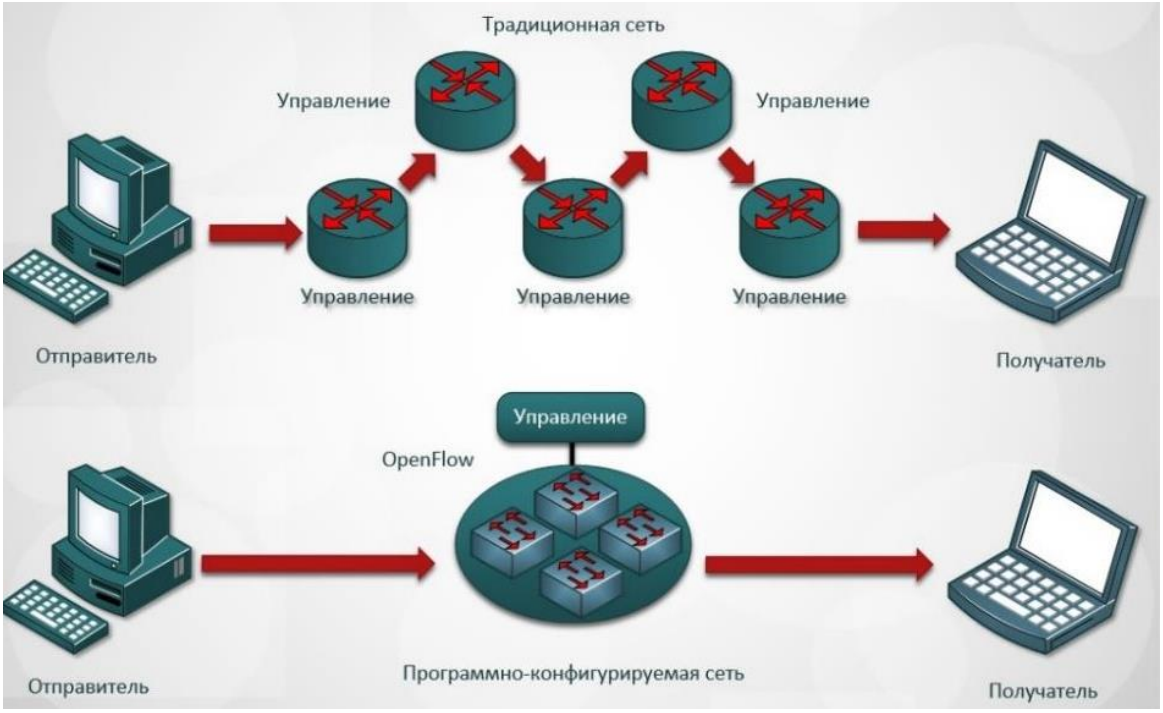


Рисунок 4. Управление сетью в традиционной сети и сети SDN

2.2.3 Уровень инфраструктуры

Состоит из среды передачи данных и коммутаторов SDN (далее Open-Flow-коммутаторы), которые могут быть как логическими, так и физическими элементами сети.

2.2.4 Функции конфигурирования оркестратором

Вертикальная сквозная в модели, которая представлена не как уровень архитектуры SDN, а как среда определения и контроля уровней управления и инфраструктуры.

1.5 Компоненты программно-конфигурируемой сети

На протяжении многих лет традиционные сети стремились уйти от централизации. Сегодня они являются децентрализованными, каждое устройство изучает сеть вокруг себя само по себе. Ключевая особенность традиционной сети как управляемой структуры – распределенность. Существующая парадигма ПКС-сетей повторно приводит к истокам развития, а точнее к идее централизованного контроля. Рассмотрим второй подход реализации SDN сетей на базе специальных аппаратных коммутаторов, которые взаимодействуют с контроллерами по протоколу OpenFlow.

Компоненты архитектуры сети:

1. Оркестратор – представленные в виде аппаратного устройства или программного обеспечения платформы автоматизации. Используются приложениями для конфигурирования устройств уровней управления и инфраструктуры (напр. корректируют работу нескольких контроллеров) и выполняют конкретные сервисные запросы. Существует несколько моделей взаимодействия контроллеров и оркестратора.

2. Контроллер – вычислительное устройство (напр. сервер), на котором функционирует специализированная платформа, состоящая из сетевых управляющих приложений и сетевой операционной системы (СОС).

- СОС – готовый фреймворк, задача которого предоставить интерфейс прикладного программирования для сетевых приложений по контролю и управлению сетью целиком, а также реализация механизмов управления таблицами одного или

нескольких OpenFlow-коммутаторов (добавление, удаление, модификация правил, сбор статистики). Разработано более 30 сетевых операционных систем для контроллеров SDN сетей: Ryu, Maestro, MUL, Beacon, Floodlight, POX, NOX, In-kernel и т.д.

- SDN приложение – программная реализация различных сетевых функций и сервисов (маршрутизация, балансировка нагрузки, фильтрация трафика, шлюзы, сетевые экраны, шифрование, DPI, NAT, DHCP, DNS и т.д.). На данный момент реализованы целые онлайн-магазины SDN-приложений, которые позволяют загружать приложения на контроллер и сразу же их использовать.

Архитектура контроллера – многоуровневая. Жестких требований к структуре пока не стандартизировано и по сути он является «черным ящиком». На рисунке 5 приведена обобщенная возможная схема архитектуры компонентов контроллера.

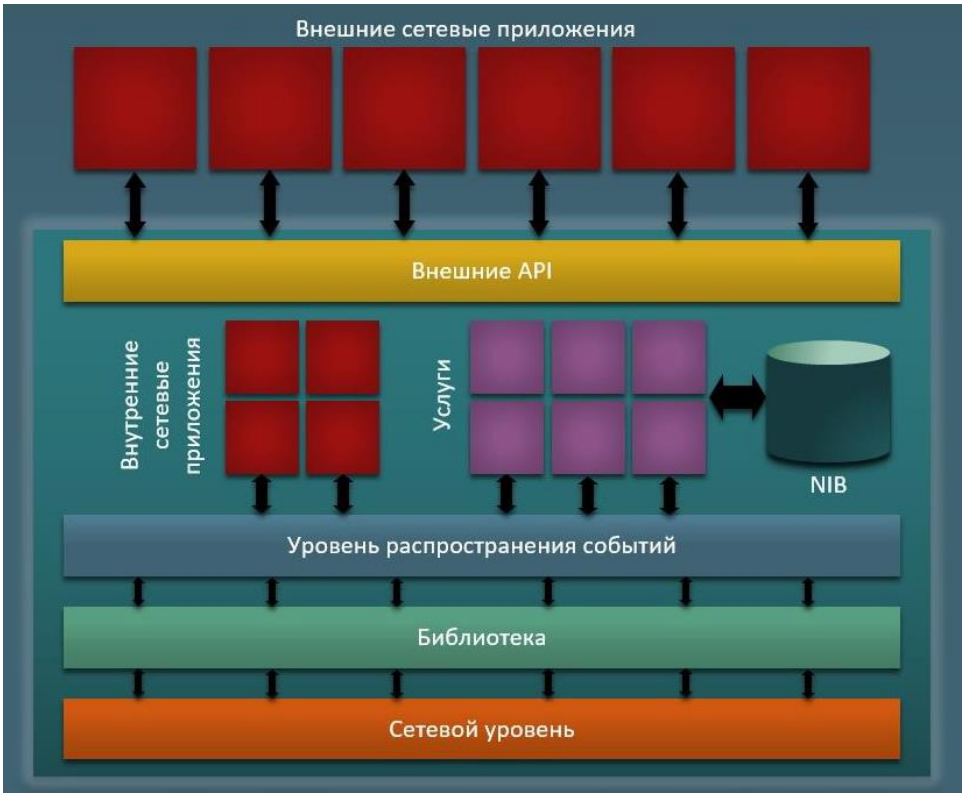


Рисунок 5. Обобщенная архитектура SDN контроллера

Краткий алгоритм работы выглядит так: сначала необходимо достоверно принять пакеты через защищенный канал, далее следует выделить OpenFlow-сообщения, понять от какого коммутатора они приходят, разобрать, сгенерировать события, на которые подписаны приложения и сервисы контроллера. После приложение получает сообщение «packet-in», обрабатывает его и отправляет ответ обратно на уровень взаимодействия с коммутатором. Важно отметить, что приложения можно интегрировать внутрь контроллера или же запускать их с отдельного сервера и «общаться» с контроллером через унифицированный интерфейс. Выбор зависит от требований к производительности системы. Важный компонент структуры контроллера Network Information Base (NIB) – это сетевая информационная база, которая содержит всю информацию о графе сети, то есть информацию о всех устройствах, которые находятся под управлением данного контроллера. База строится с помощью протокола OpenFlow для полного представления топологии сети и мониторинга за сетевыми узлами в режиме реального времени.

3. OpenFlow-коммутатор – простое программируемое сетевое устройство, выполняющее лишь функции коммутации данных (forwarding) согласно инструкциям контроллера. Основные составляющие OpenFlow-коммутатора, принцип работы и уровни устройства описаны в пункте «Уровни сетевого устройства» данной главы.

4. Программный интерфейс (northbound API) [5] – открытый интерфейс программирования, позволяющий программировать контроллер извне. Предназначен для создания экосистемы приложений. Пользователями данного API являются все разработчики сетевых приложений. Новая модель программных интерфейсов базируется на нескольких компонентах:

- NETCONF (Network Configuration Protocol) – протокол определяет простой механизм управления сетевым устройством.

Описывает процессы манипуляции (извлечение, восстановление, загрузка) с данными конфигурации устройства. Протокол позволяет устройству сформировать программный API-интерфейс, который используется приложениями для отправки и получения частичных или полных наборов данных. NETCONF использует технологию удаленного вызова процедур реализованную на базе XML (RPC-XML) для запуска функций или процедур в другом адресном пространстве;

- YANG – язык моделирования данных. Модули YANG позволяют описать структуру типов данных в формате XML для протокола NETCONF. Описываемые типы данных: конфигурация устройства, данные о состоянии, удаленный вызов процедур, уведомления. YANG модули – это иерархическая структура данных, представленная в виде дерева, в которой каждый узел имеет имя, а также значение или набор дочерних узлов. Язык содержит четкие и краткие описания взаимодействия между узлами;
- RESTCONF – чтобы понять, что это за протокол предварительно немного обобщим вышеописанные понятия. Протокол NETCONF описывает механизм работы четырех базовых функций CRUD (сокр. от англ. create, read, update, delete) над данными конфигурации устройств сети, язык YANG определяет синтаксис и семантику данных для NETCONF, а протокол RESTCONF, работая через HTTP, используется для упрощенного доступа к данных протокола NETCONF. Таким образом, RESTCONF позволяет реализовать CRUD операции без предварительных знаний о протоколе NETCONF.

В консорциуме ONF разработкой и стандартизацией программного интерфейсом занимается рабочая группа Architecture and Framework. Группой представлена схема «северного интерфейса».

1.6 Протокол OpenFlow

OpenFlow – протокол взаимодействия между сетевыми устройствами (OpenFlow-коммутаторами) программно-конфигурируемой сети SDN и централизованным контроллером.

1.6.1 История протокола

История протокола берет свое начало раньше концепции SDN и даже раньше самого протокола OpenFlow. В Стэнфордском университете, в рамках программы «дизайн интернета с чистого листа» (Clean Slate Design for the Internet) Мартином Касадо был разработан проект Ethane. Проект моделирует корпоративную сеть на 400 машин, в которой даже теоретически не могут распространяться вирусы. Машины внутри сети обмениваются информацией, но не свободно, а получая разрешение на обмен трафиком определенного вида. Таким образом, вся сеть находится под управлением единой системы филь-тров. Изначально Ethane разрабатывался как подход к борьбе со спамом и ви-русами, но идея централизации управления сетевыми устройствами и реализа-ция определенного уровня абстракции сети после закрытия проекта были ис-пользованы в дальнейших разработках OpenFlow и SDN. Концепция прото-кола была сформулирована в 2008 году, в декабре 2009-го года вышла первая версия спецификации – OpenFlow 1.0. С 2011 года продвижением протокола занимается ONF.

1.6.2 Версии протокола

Развитие протокола прошло через ряд версий, на сегодняшний день последняя выпущенная консорциумом ONF версия – 1.5.1. Первая версия протокола является широко используемой, однако имеет проблемы с масштабируемостью и большое количество ограничений. Версии 1.1 и 1.2 считаются переходными и практически никто из производителей не

реализует их поддержку. Наиболее перспективной считается версия 1.3 (включена поддержка MPLS тэгов, счетчиков, Provider Backbone Bridging (PBB) и еще некоторых полезных функций). Протокол является полностью open-source источником, однако его разработки контролируются закрытой группой, состоящей примерно из 150 компаний, формирующих ONF. Разработки ведутся в скрытом режиме и их результат виден только после публикации новой версии в качестве стандарта.

1.6.3 Уровни сетевого устройства

Суть концепции программно-конфигурируемых сетей заключается в выносе плоскости управления на отдельное устройство. Протокол OpenFlow – это один из множества инструментов реализации разделения двух плоскостей (Control Plane и Data Plane). Чтобы четко выделить область работы протокола и определить его функции, рассмотрим задачи трех независимых уровней из которых состоит практически каждое сетевое устройство независимо от концепций и архитектур сетей:

› Management plane – предоставление интерфейса управления и мониторинга, с помощью которого производится конфигурация устройства;

› Control Plane – принятие решений о перенаправлении PDU (Protocol Data Unit) путем воздействия на Data plane. Состоит не только из протоколов маршрутизации (OSPF, IS-IS), но и протоколов, контролирующих взаимодействие между смежными узлами (BFD, STP, LACP);

› Data Plane – пересылка PDU согласно правилам, определенным таблицей, которая может состоять из MAC-адресов и соответствующих исходящих портов. Data plane не отвечает за формирование таблицы FIB (Forwarding Information Base).

В концепции SDN, Management plane не так плотно взаимодействует с протоколами типа OpenFlow, в отличие от Control и Data plane. Уровень данных представлен OpenFlow-коммутаторами, которые состоят из следующих компонент (Рис. 6) .

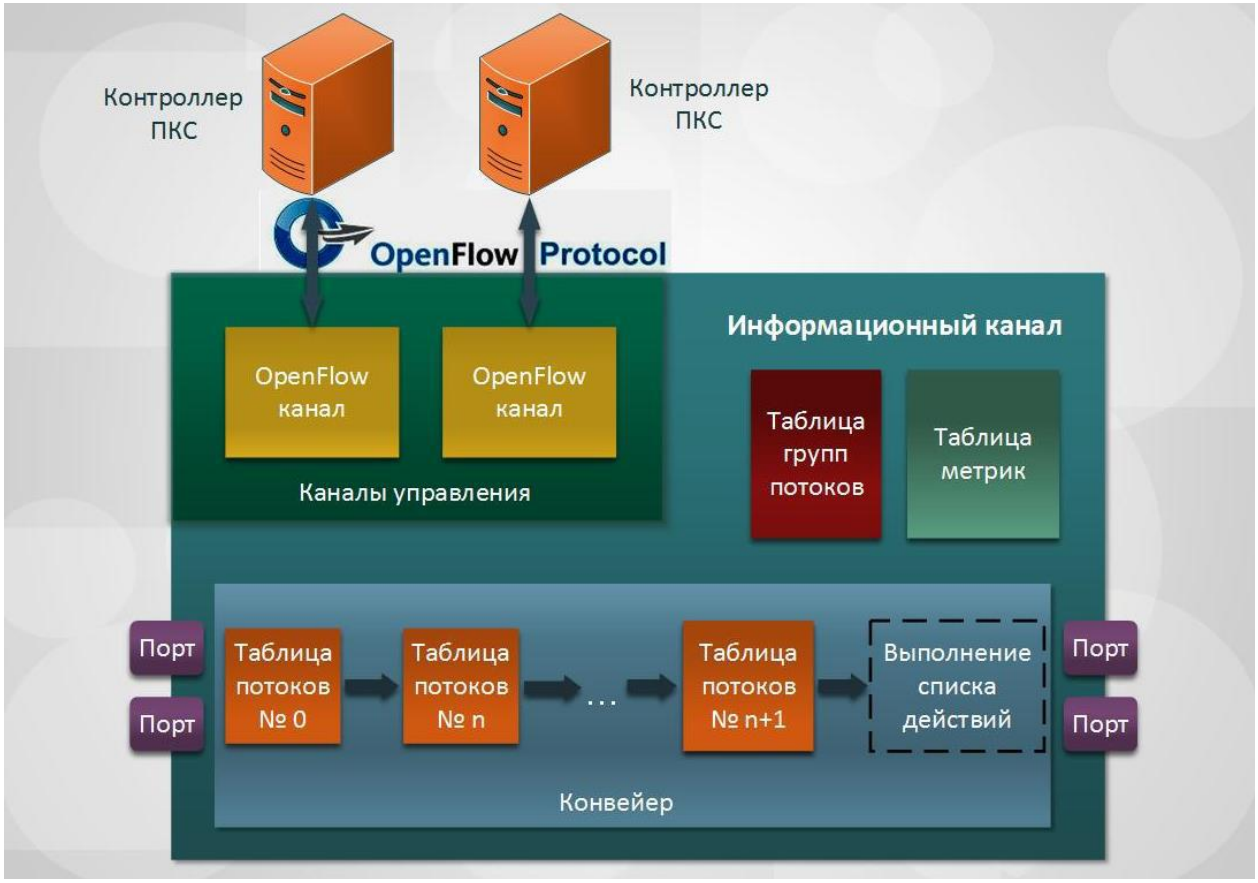


Рис. 12. Основные компоненты OpenFlow-коммутатора

1.6.4 OpenFlow-порты

Ничем не отличаются от портов обычного коммутатора уровня L2, каждый порт имеет входной/выходной буфер и уникальный номер, который выступает в роли имени порта. Протокол OpenFlow определяет набор стандартных портов:

› Физические (physical) – физические (аппаратные) порты OpenFlow-коммутатора;

› Логические (logical) – порты, которые не обязательно соответствуют физическим интерфейсам. Представляют собой более высокий уровень абстракции и могут быть определены без использования протокола OpenFlow (напр. интерфейс кольцевой проверки (loopback), нулевой или туннельный интерфейс);

› Зарезервированные (reserved) – используются для внутренней обработки трафика и для гибридных типов внедрения (OpenFlow + традиционная сеть). Представляют собой специальное действие по пересылке пакета, могут быть опциональными (Local, Normal, Flood) и обязательными:

- All – дублирование пакета на все порты определяемые действием. (используется только для исходящих пакетов);
- Controller – инкапсуляция пришедшего пакета и отправка по OpenFlow каналу на контроллер (используется как для входящих, так и для исходящих пакетов);
- Table – действие по обработке и передаче пакета в первую таблицу в конвейере;
- In port – отсылка пакета, поступившего на вход;
- Any – специальное действие, используемое в некоторых OpenFlow-командах, когда не указан адрес порта.

2.4.5 Таблицы OpenFlow

- Таблица потоков (Flow table)

Каждый OpenFlow-коммутатор содержит одну или несколько уникальных таблиц, которые он заполняет на основе данных, полученных от центрального контроллера. Так как по сети передаются не отдельные пакеты, а потоки данных, то такая таблица получила название Flow table (таблица потоков). Под потоком понимается совокупность пакетов (сеанс связи), определяемая по различным признакам и ограниченная только возможностями реализации самих таблиц (напр. TCP-сессия, пакеты с определенного MAC или IP адреса, пакеты с одинаковым номером порта OpenFlow-коммутатора). Составляющие таблицы потоков:

› Классификатор – наименование поля заголовка пакета (field) и строка из двоичных символов и символа неопределенности (pattern). Запись о потоке применима к пакету, если все двоичные символы строки pattern поля field

совпали с соответствующим полем пришедшего пакета и все классификаторы (список пар (field, pattern)) совместимы с заголовком пакета;

› Приоритет – натуральное число из определенного диапазона, указывающее степень значимости записи о потоке в таблице потоков. Используется для избирательного применения записей о потоках к конкретному пришедшему пакету (для обработки выбирается запись с наибольшим приоритетом);

› Счетчики – содержат статистические данные о работе записи (напр. сколько пакетов было обработано в соответствии с данной записью о потоке);

› Инструкции – операция, которая содержит либо множество действий, чтобы добавить их в список действий (action set), либо действия, которые немедленно применяются к пакету или модифицируют процесс обработки пакета на конвейере, в зависимости от этого различают несколько типов инструкций:

1. Apply-Actions – применяются незамедлительно без изменения списка действий. Могут использоваться для модификации пакета между двумя таблицами или для выполнения нескольких действий одного и того же типа;

2. Clear-Actions – удаляет все действия из списка действий;

3. Write-Action – заносит действия в список действий, если действие данного типа присутствует в списке, то оно перезаписывается, в противном случае добавляется;

4. Write-Metadata metadata / mask: записывает маскированные метаданные в поле метаданных;

5. Goto-Table next-table-id: указывает следующую таблицу в процессе конвейерной обработки.

› Временные метки – метки, которые определяют срок жизни записи в таблице потоков (максимальное время пребывания и максимальный срок простоя правила).

- Таблица групп потоков (Group table)

Кроме таблиц потоков в протоколе OpenFlow, начиная с версии 1.1, реализуется механизм OpenFlow-групп. Группа представляет из себя список подмножеств, состоящих из набора действий и связанных параметров. Каждая группа представлена в виде записи в таблице групп потоков. В зависимости от того какие действия в группе необходимо выполнить для получаемых пакетов, различают несколько типов групп:

› all – выполняются все подмножества действий в группе (multicast);

› select – выполняется одно подмножество действий в группе. На основе определенного алгоритма пакеты посылаются в единственное подмножество в группе (распределение нагрузки);

› indirect – выполняется одно из определенных подмножеств в группе. Используется для указания общего идентификатора группы (простая адресация);

› fast failover – выполняется первое действующее (рабочее) подмножество в группе.

Такой подход позволяет усовершенствовать механизмы форвардинга и реализовать много дополнительных опций. Кроме указанных выше таблиц существуют еще таблицы метрик (Meter table), которые позволяют реализовать примитивные настройки качества обслуживания (QoS), например, ограничение полосы пропускания.

1.6.6 Действия (Actions)

Каждому потоку в таблице ассоциировано одно или несколько действий, которые выполняются, когда поток соответствует записи. Под действием понимается операция, которая изменяет пакет, список действий (action set) и/или процесс обработки конвейером в зависимости от этого различают несколько типов действий:

› Output – передача пакета на указанный порт (физический, логический, зарезервированный);

› Set-Queue – задает идентификатор очереди для пакета;

› Drop – отбросить пакет, принадлежавший потоку;

› Group – обработка пакета в соответствии с указанной группой;

› Push-Tag/Pop-Tag – возможность работать с метками (VLAN, MPLS);

› Set-Field – изменение соответствующих значений поля заголовка пакета;

› Change-TTL – изменение значения поля TTL.

В OpenFlow-коммутаторе действия могут быть представлены не по отдельности, а списком действий. Список действий состоит из набора действий, связанных с пакетом, который накапливается пока пакет обрабатывается таблицами потоков, и выполняется только при выходе пакета из конвейера обработки.

2.4.7 Сообщения протокола

Каждый OpenFlow-пакет начинается с заголовка следующего формата Рис. 7.

Рисунок 7. Заголовок протокола OpenFlow

Компоненты заголовка:

› Version – версия протокола OpenFlow;

› Length – длина сообщения, включая заголовок;

›Transaction ID – идентификатор пакета, используется для сопоставления запроса с ответом;

› Type – тип сообщения, может принимать следующие значения (Табл. 1.):

Таблица 1. Сообщения протокола OpenFlow

№	Тип	Направление передачи
Неизменные сообщения		
1	OFPT_HELLO	Симметричное
2	OFPT_ERROR	Симметричное
3	OFPT_ECHO_REQUEST	Симметричное
4	OFPT_ECHO_REPLY	Симметричное
5	OFPT_EXPERIMENTER	Симметричное
Сообщения конфигурирования OpenFlow-коммутатора		
6	OFPT_FEATURES_REQUEST	Контроллер ↔ Коммутатор
7	OFPT_FEATURES_REPLY	Контроллер ↔ Коммутатор
8	OFPT_GET_CONFIG_REQUEST	Контроллер ↔ Коммутатор
9	OFPT_GET_CONFIG_REPLY	Контроллер ↔ Коммутатор
10	OFPT_SET_CONFIG	Контроллер ↔ Коммутатор
Асинхронные сообщения		
11	OFPT_PACKET_IN	Асинхронное
12	OFPT_FLOW_REMOVED	Асинхронное
13	OFPT_PORT_STATUS	Асинхронное
Сообщения-команды контроллера		
14	OFPT_PACKET_OUT	Контроллер ↔ Коммутатор
15	OFPT_FLOW_MOD	Контроллер ↔ Коммутатор
16	OFPT_GROUP_MOD	Контроллер ↔ Коммутатор
17	OFPT_PORT_MOD	Контроллер ↔ Коммутатор
18	OFPT_TABLE_MOD	Контроллер ↔ Коммутатор
Составные сообщения		
19	OFPT_MULTIPART_REQUEST	Контроллер ↔ Коммутатор
20	OFPT_MULTIPART_REPLY	Контроллер ↔ Коммутатор
Barrier сообщения		
21	OFPT_BARRIER_REQUEST	Контроллер ↔ Коммутатор
22	OFPT_BARRIER_REPLY	Контроллер ↔ Коммутатор
Сообщения конфигурации очередей		
23	OFPT_QUEUE_GET_CONFIG_REQUEST	Контроллер ↔ Коммутатор

24	OFPT_QUEUE_GET_CONFIG_REPLY	Контроллер ↔ Коммутатор
Асинхронные сообщения конфигурации		
25	OFPT_GET_ASYNC_REQUEST	Контроллер ↔ Коммутатор
26	OFPT_GET_ASYNC_REPLY	Контроллер ↔ Коммутатор
27	OFPT_SET_ASYNC	Контроллер ↔ Коммутатор
Сообщения конфигурации таблицы метрик		
28	OFPT_METER_MOD	Контроллер ↔ Коммутатор

1.7 Алгоритм работы ПКС-сети

Далее подробно рассмотри этапы работы программно-конфигурируемой сети. Предварительно рассмотрим этап подключения OpenFlow-коммутатора к контроллеру:

1. При подключении OpenFlow-коммутатора к контроллеру, стороны обмениваются сообщением OFPT_HELLO, состоящим из заголовка и набора полей, содержащих версию поддерживаемого протокола и данные информирующие о начале сеанса связи;

2. После установления сессии контроллер посылает сообщение OFPT_FEATURES_REQUEST для идентификации OpenFlow-коммутатора и считывания его возможностей;

3. Коммутатор отвечает сообщением OFPT_FEATURES_REPLY в котором содержатся поля: однозначная идентификация маршрута (datapath_id), максимальное число буферизованных пакетов (n_buffers) в сообщении (packet-in) при отправке контроллеру, число поддерживаемых таблиц потоков (n_tables), тип соединения (основное или вспомогательное) (auxiliary_id) и комбинация флагов;

4. OpenFlow-коммутатор информирует контроллер о изменении состояния портов (напр. добавлены, удалены, конфигурированы) сообщением OFPT_PORT_STATUS;

5. Контроллер инициирует запрос OFPMP_PORT_DESC в виде составного сообщения OFPT_MULTIPART_REQUEST на получение описания всех портов сети, которые поддерживают протокол OpenFlow;

6. OpenFlow-коммутатор реагирует ответом в виде одного или нескольких сообщений OFPT_MULTIPART_REPLY с параметрами описания портов OFPMP_PORT_DESC;

7. Контроллер конфигурирует OpenFlow-коммутатор командой OFPT_SET_CONFIG или OFPT_GET_CONFIG_REQUEST (очереди, фрагментация, потеря пакетов и т.д.);

8. Контроллер посылает запрос OFPT_BARRIER_REQUEST на получение информации о состоянии всех выполняемых операциях и настройках на OpenFlow-коммутаторе в данный момент времени;

9. OpenFlow-коммутатор завершает обработку всех запросов, полученных до OFPT_BARRIER_REQUEST и отправляет ответ OFPT_BARRIER_RE-PLY («срез» о текущем состоянии OpenFlow-коммутатора);

10. OpenFlow-коммутатор реагирует на запрос конфигурации сообщением OFPT_GET_CONFIG_REPLY;

11. Контроллер инициирует запрос OFPMP_DESC в виде составного сообщения на получение описания производителя коммутатора, версии аппаратного обеспечения, серийного номера и др., далее получает ответ;

12. Контроллер инициирует запрос OFPMP_TABLE_FEATURES в виде составного сообщения на получение информации об используемых потоковых таблицах (тип, размер, порядок использования) и получает ответ;

13. Контроллер инициирует запрос OFPT_ROLE_REQUEST на изменение правил, далее получает ответ;

14. Контроллер инициирует запрос OFPT_FLOW_MOD на конфигурацию таблиц потоков;

15. Контроллер инициирует запрос OFPT_GROUP_MOD на конфигурацию таблицы групп потоков;

16. Контроллер посылает пакет OFPT_PACKET_OUT с широковещательным ARP запросом для получения MAC-адресов всех подключенных устройств;

17. Контроллер получает пакет OFPT_PACKET_IN с ARP ответом;

18. Для проверки соединения контроллер периодически посылает на OpenFlow-коммутатор запросы OFPT_ECHO_REQUEST (содержат только заголовок), а в ответ получает OFPT_ECHO_REPLY, если запрос соответствует ответу, то соединение находится в рабочем состоянии.

Описанные выше этапы подключения OpenFlow-коммутатора к контроллеру изобразим в виде MSC (Message Sequence Chart) диаграммы (Рис. 8).

После завершения этапа подключения всех OpenFlow-коммутаторов к контроллеру, сеть находится в режиме ожидания и готова к работе. Рассмотрим обработку пакета, поступившего с хоста, подключенного к OpenFlow-коммутатору:

1. Пакет поступает на вход одного из буферов накопителей OpenFlow-коммутатора;

2. Дождавшись своей очереди пакет начинает обрабатываться: очищается список действий, инициализируются данные о работе конвейера, из пакета извлекаются данные заголовка;

3. Конвейерная обработка начинается с нулевой таблицы потоков. Ведется поиск приоритетной записи – анализ на совпадение полей заголовка с классификаторами в таблице потоков (используется алгоритм TCAM), если найдено несколько соответствующих записей, то выбирается запись с наибольшим приоритетом и выполняются указанные инструкции обновляются показатели счетчиков и таймеров;

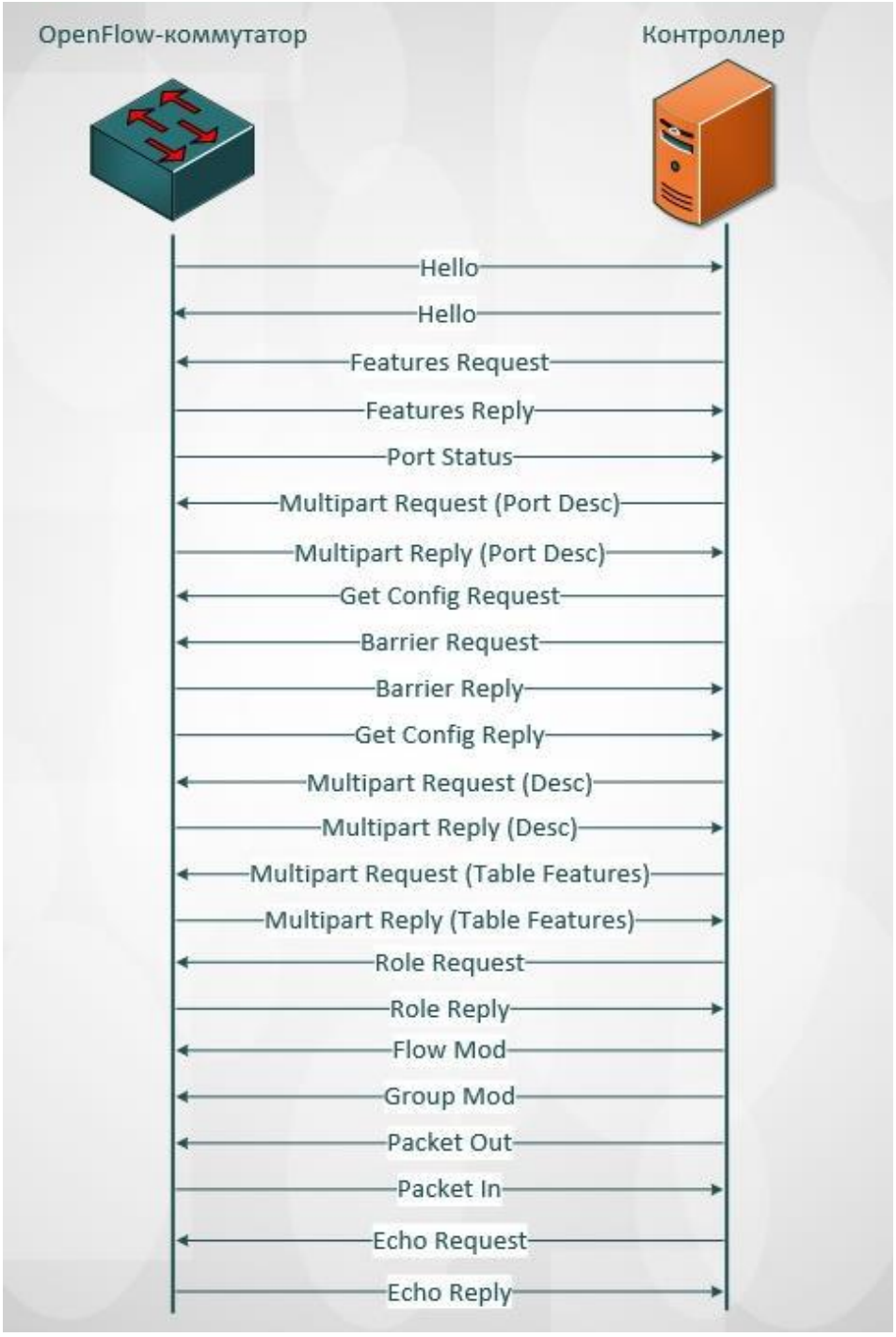


Рисунок 8. MSC диаграмма обмена сообщениями при подключении OpenFlow-коммутатора

4. Инструкции в записи могут явно направить пакет в другую таблицу потоков, где такой же процесс обработки пакета повторяется снова;

5. Если при просмотре записи не происходит перенаправления пакета в другую таблицу, то конвейерная обработка для данного пакета завершается и пакет обрабатывается в соответствии с накопившимся списком действий в процессе обработки конвейером;

6. Если OpenFlow-коммутатор получил пакет, который не соответствует ни одной записи в таблице потоков, в инструкции указан непосредственный вывод на контроллер или неверно указано значение поля TTL, то копия инкапсулируется и по защищенному каналу посылается на контроллер;

7. Контроллер обрабатывает пакет и указывает из какого порта OpenFlow-коммутатора его необходимо отправить или просто отбросить;

8. После применения всех действий пакет отправляется из указанного порта.

Этапы обработки пакета, отправленного с хоста, подключенного к OpenFlow-коммутатору, в виде цифр изображены на схеме рисунка 9.

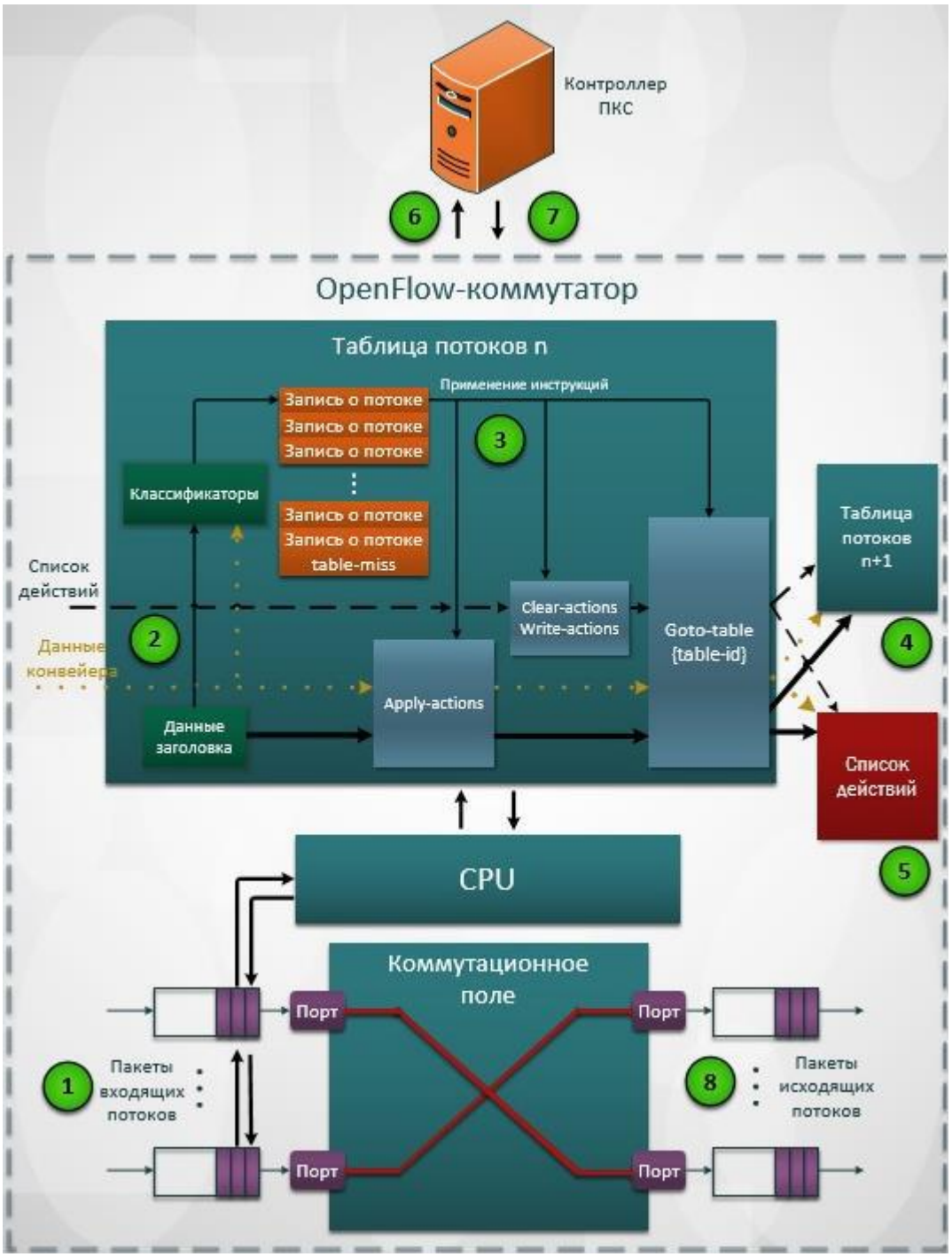


Рисунок 9. Алгоритм работы OpenFlow-коммутатора

1.8 Классификаторы протокола OpenFlow

Ранее, в пункте «Таблицы OpenFlow» частично были описаны поля и структура классификаторов как одной из нескольких составляющих таблиц потоков. С выходом новых версий протокола OpenFlow строение классификаторов постепенно усложнялось, начиная с версии протокола 1.3

каждый классификатор может состоять из заголовка и от нуля до некоторого количества полей соответствия классификаторов, закодированных при помощи формата OXM TLV. Чтобы визуализировать из каких типов и классов состоят классификаторы протокола OpenFlow, изобразим следующую схему (Рис. 10):

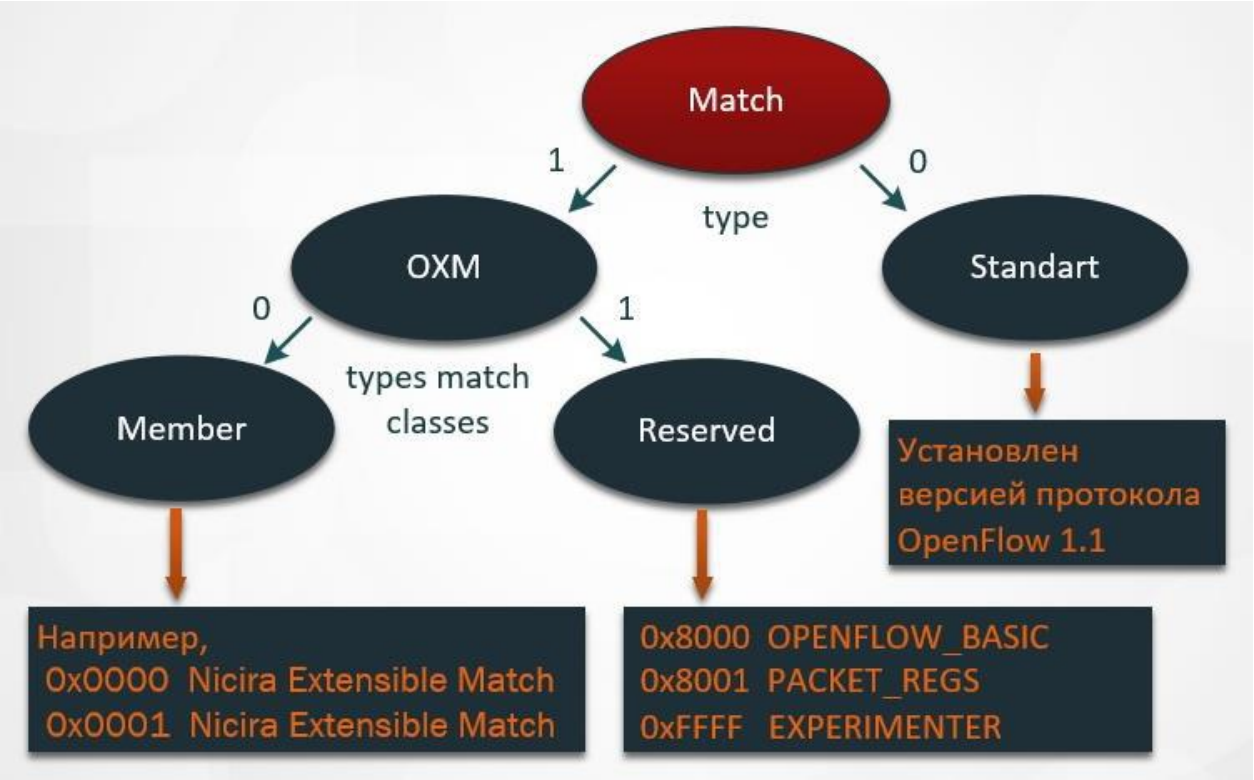


Рисунок 10. Типы классификаторов протокола OpenFlow

1.8.1 Заголовок классификаторов

Поля заголовка:

› Type – определены два типа классификаторов. Первый тип – стандартный (значение поля «тип» равняется нулю), определены версией протокола OpenFlow 1.1 и считаются устаревшими, но базовыми для всех OpenFlow-коммутаторов. Второй тип – расширяемые классификаторы OpenFlow Extensible Match (OXM);

› Length – динамически может изменяться в указанных пределах, зависит от количества полей классификаторов;

› OXM fields – поля классификаторов;

› Pad – нулевой байт.

1.8.2 OXM fields

Все поля соответствия классификатора описываются, используя динамически расширяемый формат OXM, которые в совокупности (вместе) образуют компактный type-length-value (TLV) формат. Каждый OXM TLV классификатор может быть от 5 до 259 (включительно) байт длиной. Первые 4 байта каждого OXM TLV классификатора – заголовок (Рис. 11).

Рисунок 11. Заголовок OXM TLV классификатора

1. OXM class – спецификация OpenFlow разделяет два типа OXM классов:

› ONF reserved – классы, описываемые спецификацией и зарезервированные для последующих стандартизаций;

› ONF member – классы, используемые членами консорциума ONF. Позволяют членам консорциума использовать свои классификаторы и уникально идентифицироваться, благодаря им (напр. Nicira).

2. OXM field – поле используется для определения конкретного поля соответствия в каждом типе класса (см. табл. 2 и 3);

3. OXM hasmask – поле может принимать значение 0 или 1. Определяет наличие в поле данных OXM битовую маску;

4. OXM length – длина полезных данных OXM TLV.

1.8.3 OXM classes – reserved

Тип OXM классов – Member не описывается спецификациями и не требует разбора в рамках данной работы, поскольку поиск параметров для контроля, установленных 83 приказом Министерства связи и массовых коммуникаций будет проводиться в одном из типов OXM классов – Reserved. Рассмотрим классы Reserved на примере протокола OpenFlow версии 1.5.1. В спецификации протокола определены три класса в данном типе: basic, register и experimenter.

› experimenter – класс используется для расширения стандартного набора полей соответствия в OpenFlow-коммутаторе;

› register – класс используется для хранения временных значений и информации, связанной с пакетом в процессе обработки конвейером;

› OpenFlow basic – основной класс классификаторов, который содержит базовый набор полей соответствия, однако не все из этого набора полей должны обязательно поддерживаться OpenFlow-коммутатором, но каждое обязательное match-поле должно поддерживаться не менее чем одной таблицей потоков. Кроме базовых полей соответствия основного класса, в необязательном порядке, таблицы потоков могут содержать match-поля классов experimenter и register. Поля соответствия класса OpenFlow basic разделяют на два типа. К первому типу относятся поля, которые сравниваются с данными, извлеченными из заголовков пакетов (header match fields). Все они имеют разный размер, обязательность/необязательность к реализации, наличие маски и описание (табл. 2).

Таблица 2. Поля соответствия класса OpenFlow basic (header match fields)

№	**Math-поле**	**Размер (бит)**	**Маска**	**Описание**
1	OFPXMT_OFB_ETH_DST	48	Да	MAC-адрес получателя
2	OFPXMT_OFB_ETH_SRC	48	Да	MAC-адрес отправителя
3	OFPXMT_OFB_TYPE	16	Нет	Ethernet type

4	OFPXMT_OFB_VLAN_VID	12+1	Да	VLAN-ID по стандарту 802.1Q
5	OFPXMT_OFB_VLAN_PCP	3	Нет	VLAN-PCP по стандарту 802.1Q
6	OFPXMT_OFB_IP_DSCP	6	Нет	Diff Serv Code Point (DSCP)
7	OFPXMT_OFB_IP_ECN	2	Нет	ECN бит в заголовке IP
8	OFPXMT_OFB_IP_PROTO	8	Нет	IPv4 или IPv6 номер протокола
9	OFPXMT_OFB_IPV4_SRC	32	Да	IPv4 адрес отправителя
10	OFPXMT_OFB_IPV4_DST	32	Да	IPv4 адрес получателя
11	OFPXMT_OFB_TCP_SRC	16	Нет	TCP порт отправителя
12	OFPXMT_OFB_TCP_DST	16	Нет	TCP порт получателя
13	OFPXMT_OFB_TCP_FLAGS	12	Нет	TCP флаги
14	OFPXMT_OFB_UDP_SRC	16	Нет	UDP порт отправителя
15	OFPXMT_OFB_UDP_DST	16	Нет	UDP порт получателя
16	OFPXMT_OFB_SCTP_SRC	16	Нет	SCTP порт отправителя
17	OFPXMT_OFB_SCTP_DST	16	Нет	SCTP порт получателя
18	OFPXMT_OFB_ICMPV4_TYPE	8	Нет	ICMP тип
19	OFPXMT_OFB_ICMPV4_CODE	8	Нет	ICMP код
20	OFPXMT_OFB_ARP_OP	16	Нет	ARP код отправителя/получателя
21	OFPXMT_OFB_ARP_SPA	32	Да	IPv4 адрес отправителя в payload протокола ARP
22	OFPXMT_OFB_ARP_TPA	32	Да	IPv4 адрес получателя в payload протокола

				ARP
23	OFPXMT_OFB_ARP_SHA	48	Да	MAC-адрес отправителя в payload протокола ARP
24	OFPXMT_OFB_ARP_THA	48	Да	MAC-адрес получателя в payload протокола ARP
25	OFPXMT_OFB_IPV6_SRC	128	Да	IPv6 адрес отправителя
26	OFPXMT_OFB_IPV6_DST	128	Да	IPv6 адрес получателя
27	OFPXMT_OFB_IPV6_FLABEL	20	Да	IPv6 метка потока
28	OFPXMT_OFB_ICMPV6_TYPE	8	Нет	ICMPv6 тип
29	OFPXMT_OFB_ICMPV6_CODE	8	Нет	ICMPv6 код
30	OFPXMT_OFB_IPV6_ND_TAR-GET	128	Нет	IPv6, сообщение NDM, адрес получателя
31	OFPXMT_OFB_IPV6_ND_SLL	48	Нет	IPv6, сообщение NDM, source link-layer address option
32	OFPXMT_OFB_IPV6_ND_TLL	48	Нет	IPv6, сообщение NDM, target link-layer address option
33	OFPXMT_OFB_MPLS_LABEL	20	Нет	MPLS метка
34	OFPXMT_OFB_MPLS_TC	3	Нет	MPLS класс трафика
35	OFPXMT_OFB_MPLS_BOS	1	Нет	MPLS флаг «дно стека»
36	OFPXMT_OFB_PBB_ISID	24	Да	Технология PBB, идентификатор сервиса I-SID
37	OFPXMT_OFB_IPV6_EXTHDR	9	Да	IPv6 Extension Header pseudo-field
38	OFPXMT_OFB _PBB_UCA	1	Нет	Технология PBB, поле UCA

Второй тип полей соответствия содержит поля, которые используются в процессе обработки конвейером (pipeline match fields) и никаким образом не взаимодействуют с заголовками пакета (табл. 3).

Таблица 3. Поля соответствия класса OpenFlow basic (pipeline match fields)

№	Math-поле	Размер (бит)	Маска	Описание
1	OFPXMT_OFB_IN_PORT	32	Нет	Входящий порт. Номер логиче-ского или физического входя-щего порта, начиная с единицы.
2	OFPXMT_OFB_IN_PHY_PORT	32	Нет	Физический порт. Соответ-ствие физического и логиче-ского порта, когда сообщение приходит на логический порт.
3	OFPXMT_OFB_METADATA	64	Да	Используется для передачи ин-формации между таблицами.
4	OFPXMT_OFB_TUNNEL_ID	64	Да	Метаданные, ассоциированные с логическим портом.
5	OFPXMT_OFB_ACTSET_OUT-PUT	32	Нет	Output port from action set Metadata.
6	OFPXMT_OFB_PACKET_TYPE	16+16	Нет	Packet type - canonical header type of outermost header.

Однако, как и каждая технология SDN имеет свои недостатки: Во-первых, каждому коммутатору, несмотря на использование OpenFlow, приходится обрабатывать большие заголовки пакетов, ища соответствия их полей в записях своих flow-таблиц. Разумеется, плюсом здесь является то, что стандарт OpenFlow рассматривает потоки пакетов, а не отдельные пакеты, соответственно обрабатывается только заголовок первого пакета потока.

Во-вторых, коммутаторам необходимо хранить большие объѐмы данных в flow-таблицах, что приводит к необходимости поддержания

достаточно большого объема памяти, и вынуждает использовать более мощные и, соответственно, достаточно дорогие платы. В-третьих, реализация всей функциональности OpenFlow на одном устройстве может подвергнуть сеть сбоям в результате перегрузки контроллера. Например, даже легальные запросы могут перегрузить контроллер и устроить сбой сети, если коммутатор настроен так, чтобы при получении TCP-пакета с флагом SYN каждый раз отправлять контроллеру.

Контрольные вопросы:

1. На каком интерфейсе (южном или северном) используется протокол OpenFlow?
2. Для чего в SDN сетях необходим контроллер?
3. Какой протокол транспортного уровня используется для OpenFlow?
4. Какие основные функции выполняет коммутатор SDN?
5. Описать основные подходы к формированию архитектуры SDN.
6. Какbt международные организации участвуют в стандартизации SDN?

2. ЛАБОРАТОРНЫЕ РАБОТЫ

2.1 Описание интерактивного обучающего курса для лабораторных работ

Вход в программу

Для запуска программы необходимо выбрать на рабочем столе ярлык интерактивного обучающего курса «Учебный курс», после чего на экране монитора ПК появится окно «Вход в программу» (рис. 2.1).

В программе предусмотрен учет пользователей, который обеспечивает сохранение статистических данных о пользователях (например, данных о полученных допусках к моделированию и выполненных работах по моделированию). Поэтому учащемуся, заходящему в программу, необходимо пройти процедуру авторизации. Обычно для этого необходимо указать в качестве логина цифры номера зачетной книжки, а в качестве пароля свою «Фамилию», с учетом регистра букв. Далее нажать кнопку «Войти» (рис. 2.1).

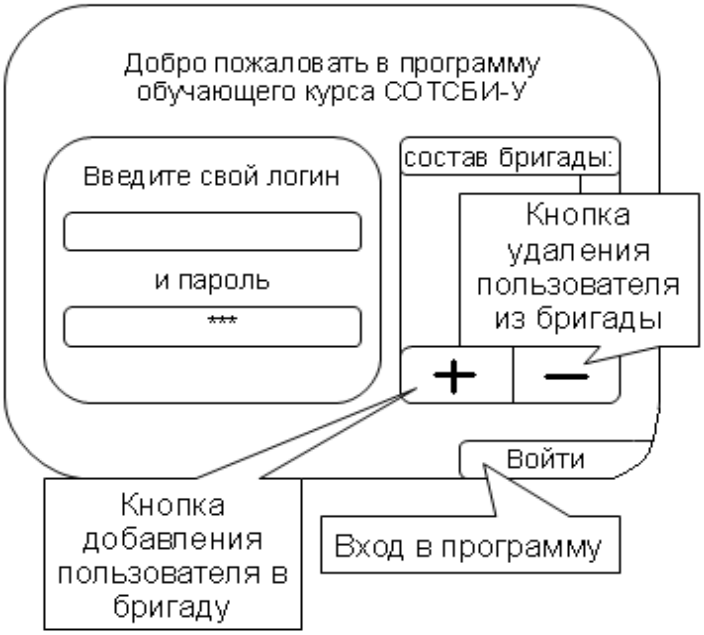


Рис.2.1. Окно авторизации пользователей

При работе нескольких пользователей на одном рабочем месте учащегося, они могут быть объединены в бригаду нажатием кнопок «+» и «-». Сохраняются только данные по проделанной работе отдельно для каждого пользователя (даже если он работал в составе бригады). После добавления всех пользователей необходимо нажать кнопку «Войти».

Главное меню

После входа в программу учащийся попадает в главное меню (рис. 2.2): раскрывающееся меню (кнопки теория, тестирование, моделирование), отчет,

сервисные кнопки (руководство пользователя, блокнот, глоссарий), системные кнопки (развернуть и выход).

После нажатия кнопок меню «Теория», «Тестирование» или «Моделирование», выбирается курс обучения (ТфОП, СПД, ИБ, СОРМ и прочее).

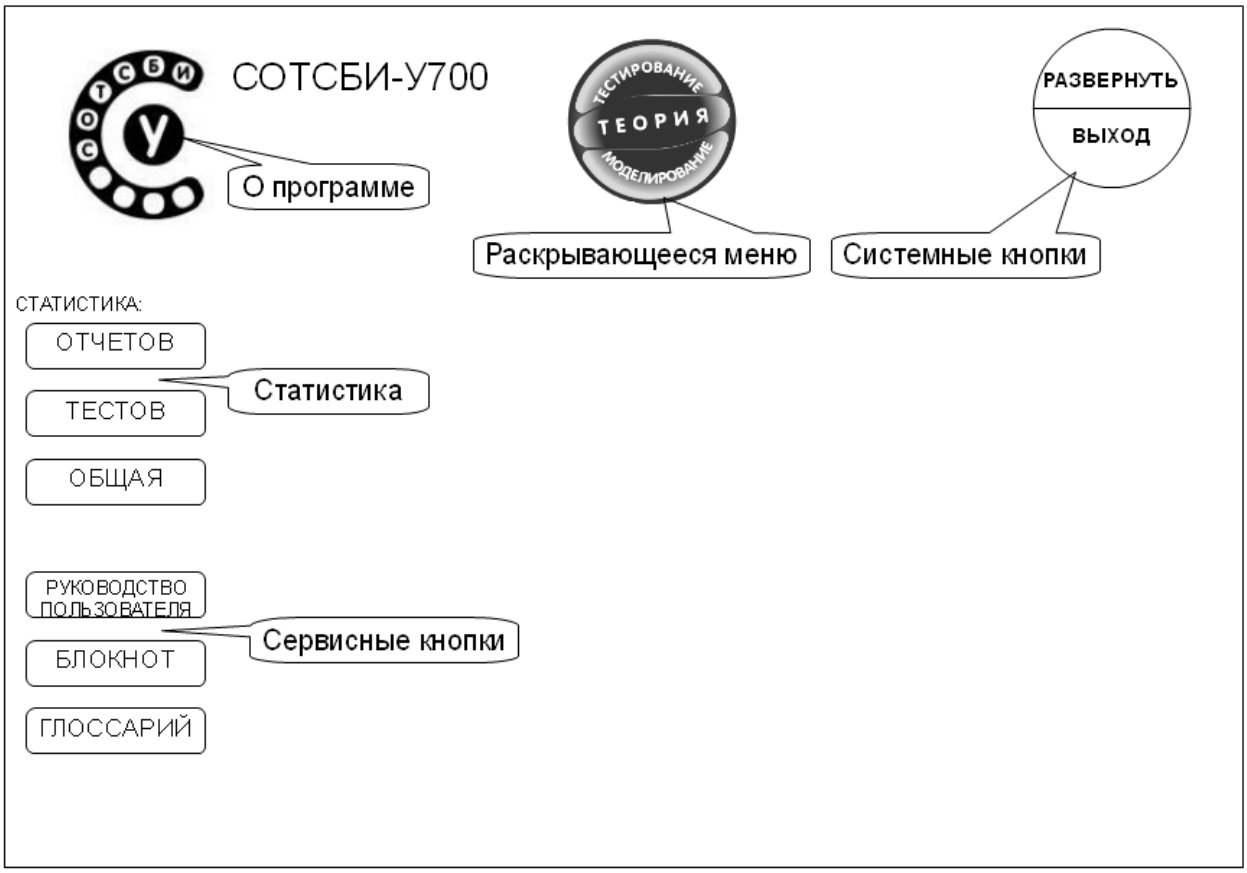


Рис. 2.2. Интерфейс главного меню

Далее выбирается конкретная глава теории, либо конкретный тест, либо доступную после прохождения теста, интерактивную работу в разделе «Моделирование». При входе в блокнот, статистику, руководство пользователя, глоссарий, о программе или при открытии теоретического материала, тестирования, моделирования в окне просмотра появляется уменьшенная версия выбранной части обучающего курса. Для перехода в нормальный режим просмотра достаточно нажать на окно просмотра.

Тестирование

Уровень изучения теоретического материала, изложенного в разд. 1, проверяется с помощью тестирования.

Интерфейс тестирования (рис. 2.3) состоит из заголовка. блока задания, блока ответов, блока статистики, кнопки «Проверить»/«Следующий вопрос» и кнопки «Возврат в главное меню». Блок задания представляет собой

вопрос, ответ на который необходимо выбрать из предложенных вариантов блока ответов.

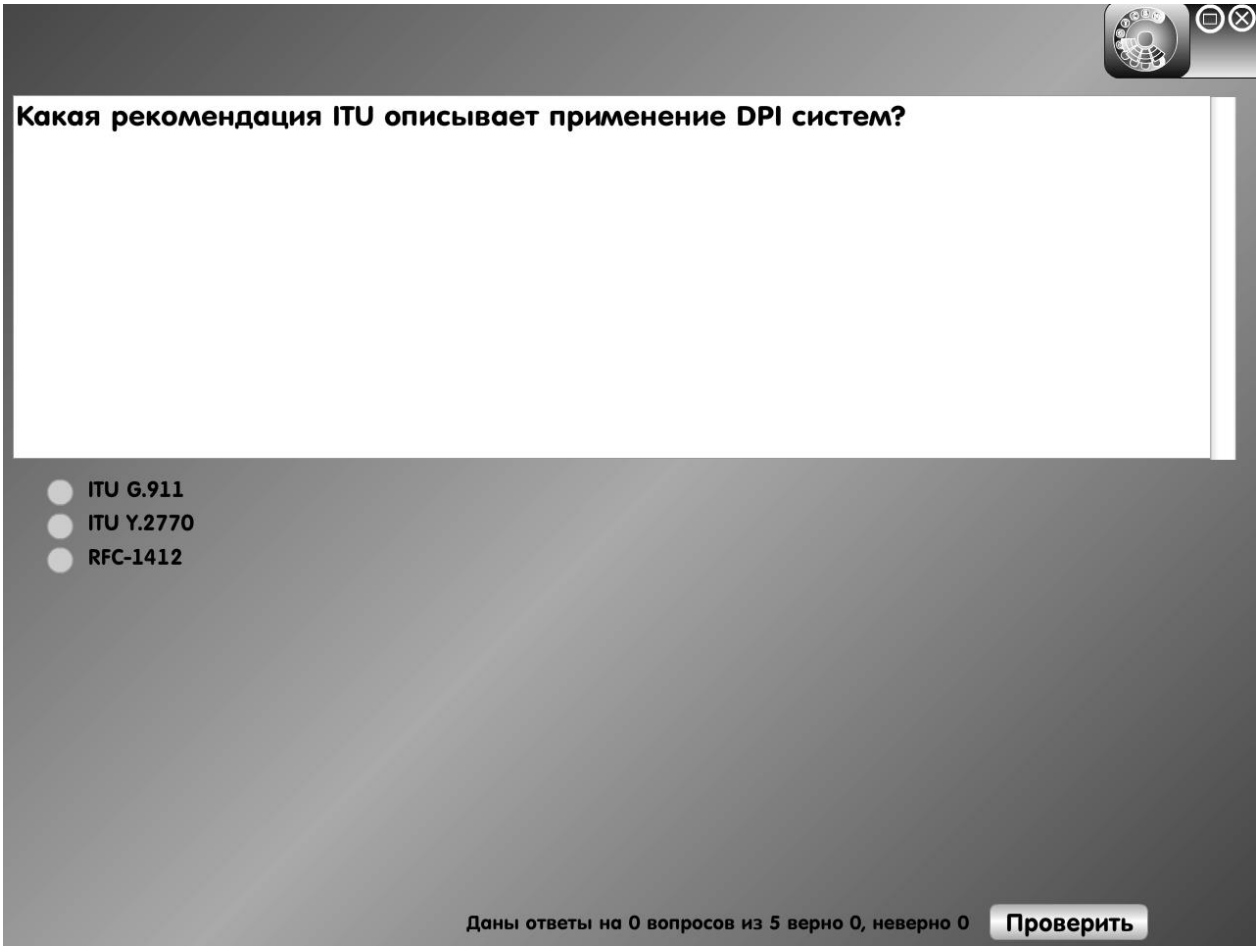


Рис. 2.3. Пример формы тестирования

После выбора, какого-либо варианта в блоке ответов, учащийся должен нажать на кнопку «Проверить». После ответа, выбранный вариант подсвечивается (зеленый цвет – верный ответ, красный цвет – неверный ответ). Кнопка «Проверить» исчезает, и появляется кнопка «Следующий вопрос», позволяющая перейти к следующему вопросу теста. Блок статистики, отображает общее количество вопросов, а также правильных и неправильных ответов. Обычно для получения допуска необходимо ответить правильно на 75% всех вопросов.

В случае если допуск не получен, программа проанализирует ошибочные ответы и сгенерирует выборку слайдов теории, которые предлагается изучить или повторить учащемуся. Эти ссылки расположены в специальном блоке «Рекомендуется изучить», из которого происходит переход к соответствующему слайду теоретического материала.

1.2. Перечень лабораторных работ

Теоретические сведения

- n-DPI

Ntopng является сетевым зондом, позволяющим наблюдать за использованием сети. Информация представляется с помощью web-интерфейса, основанного на HTML5, включающим в себя возможности конфигурации и администрирования. Ntopng выводит информацию о сети с возможностью ее сортировки по задействованным приложениям, протоколам, хостам или портам. Также имеется возможность просмотра информации о сети в виде активных соединений. Система ntopng может записывать трафик и собранную статистику.

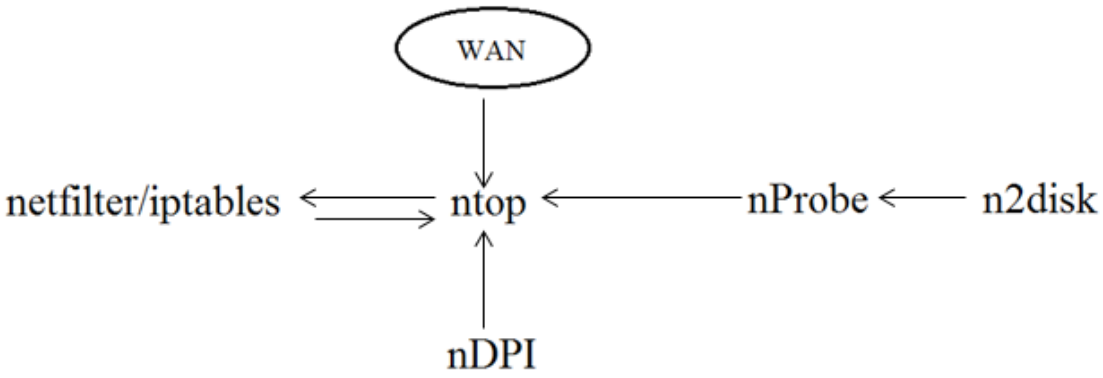


Рис.2.4. Архитектура внедрения nTOP на сети

Для распознавания приложений и протоколов ntopng использует фреймворк nDPI с открытым исходным кодом. nDPI способен идентифицировать прикладные протоколы независимо от того какие порты были использованы на транспортном уровне, благодаря заданным в нем сигнатурам. На данный момент nDPI включает в себя правила для идентификации около 200 протоколов.

Веб-интерфейс n-DPI показывает выявленные потоки трафика (рис.2.5, 2.6), время обнаружения, число обнаруженных пакетов в потоке, и число потерянных пакетов (рис.2.7).

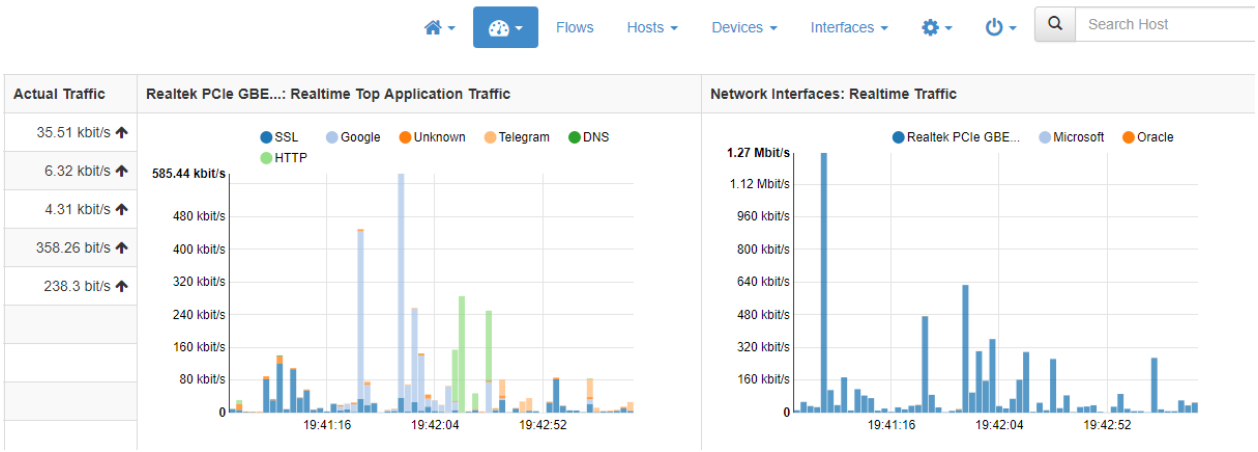
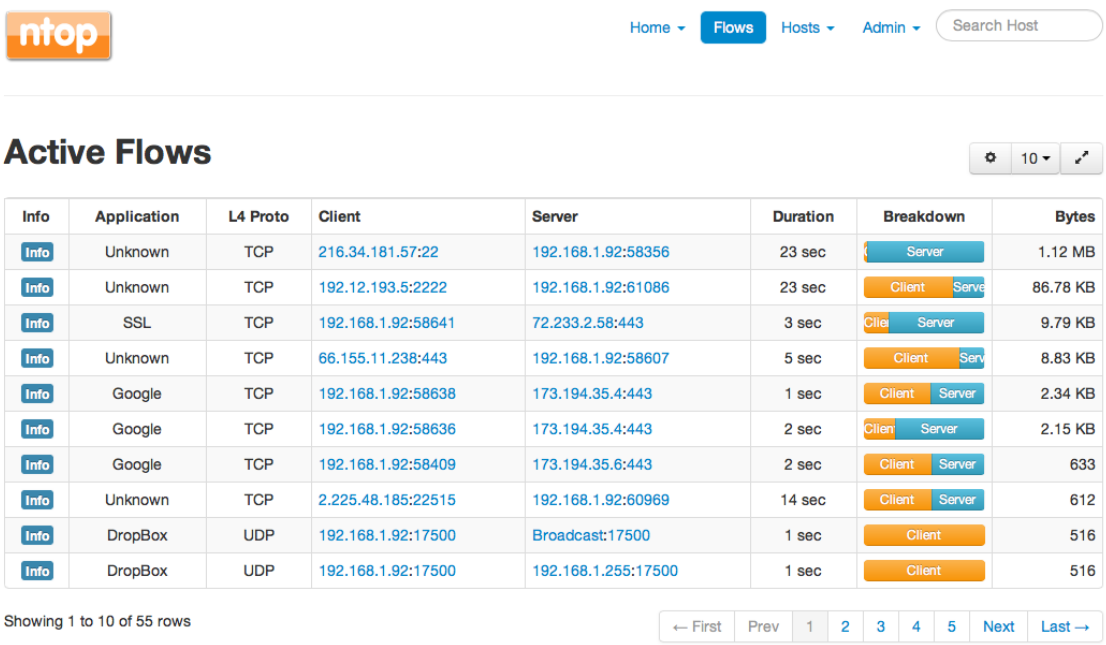


Рис.2.5. Визуализация сетевой активности в ntopng



Info	Application	L4 Proto	Client	Server	Duration	Breakdown	Bytes
Info	Unknown	TCP	216.34.181.57:22	192.168.1.92:58356	23 sec	Server	1.12 MB
Info	Unknown	TCP	192.12.193.5:2222	192.168.1.92:61086	23 sec	Client Serve	86.78 KB
Info	SSL	TCP	192.168.1.92:58641	72.233.2.58:443	3 sec	Clie Server	9.79 KB
Info	Unknown	TCP	66.155.11.238:443	192.168.1.92:58607	5 sec	Client Serv	8.83 KB
Info	Google	TCP	192.168.1.92:58638	173.194.35.4:443	1 sec	Client Server	2.34 KB
Info	Google	TCP	192.168.1.92:58636	173.194.35.4:443	2 sec	Clien Server	2.15 KB
Info	Google	TCP	192.168.1.92:58409	173.194.35.6:443	2 sec	Client Server	633
Info	Unknown	TCP	2.225.48.185:22515	192.168.1.92:60969	14 sec	Client Server	612
Info	DropBox	UDP	192.168.1.92:17500	Broadcast:17500	1 sec	Client	516
Info	DropBox	UDP	192.168.1.92:17500	192.168.1.255:17500	1 sec	Client	516

Рис.2.6. Обнаруженные nDPI потоки

На рис.2.6 отображены IP-адреса, количество потоков по каждому IP (Flows), возможные тревоги (Alerts), связанные с нарушением работы, время, в течение которого ведется наблюдение за указанным IP (Seen Since), уникальный номер автономной станции (или ASN) присваиваемый для использования в BGP маршрутизации, скорость передачи данных (Throughput) и объем трафика по данному IP (Traffic).

ASN	SPB TV Telecom LLC [ASN 197888]	Whois Lookup
Name	91.228.177.207	
First / Last Seen	15/06/2017 16:21:50 [2 min, 14 sec ago]	15/06/2017 16:24:03 [1 sec ago]
Sent vs Received Traffic Breakdown	Sent	Rcvd
Traffic Sent / Received	10,083 Pkts / 13.16 MB	3,815 Pkts / 327.11 KB
Active Flows / Total Active / Low Goodput	'As Client'	'As Server'
	0 / 0 / 0	7 / 13 / 0
TCP Packets Sent Analysis	Retransmissions	3 Pkts
	Out of Order	293 Pkts
	Lost	155 Pkts

Рис.2.7. Число обнаруженных и потерянных пакетов видео SPB TV в nDPI

- Wireshark

Система nDPI может быть использована для анализа трафика, а для проверки правильности и целостности, сигнатуры используется сниффер «Wireshark».

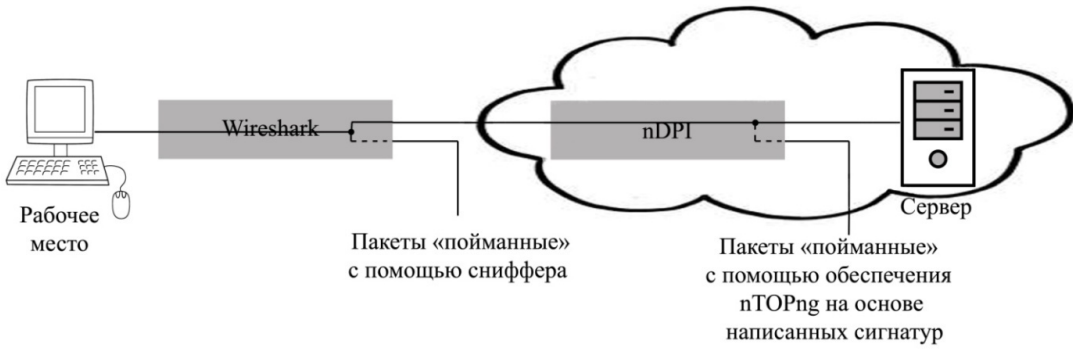


Рис.2.8. Схема лабораторной установки

Одновременно с nDPI может быть запущена программа Wireshark. Отфильтровав весь трафик и выделив в нем необходимые пакеты, полученные только с заданного IP-адреса (например, 91.228.131.209), программа-анализатор трафика указывает число перехваченных пакетов. В данном случае 26 065 пакетов (рис.2.9).

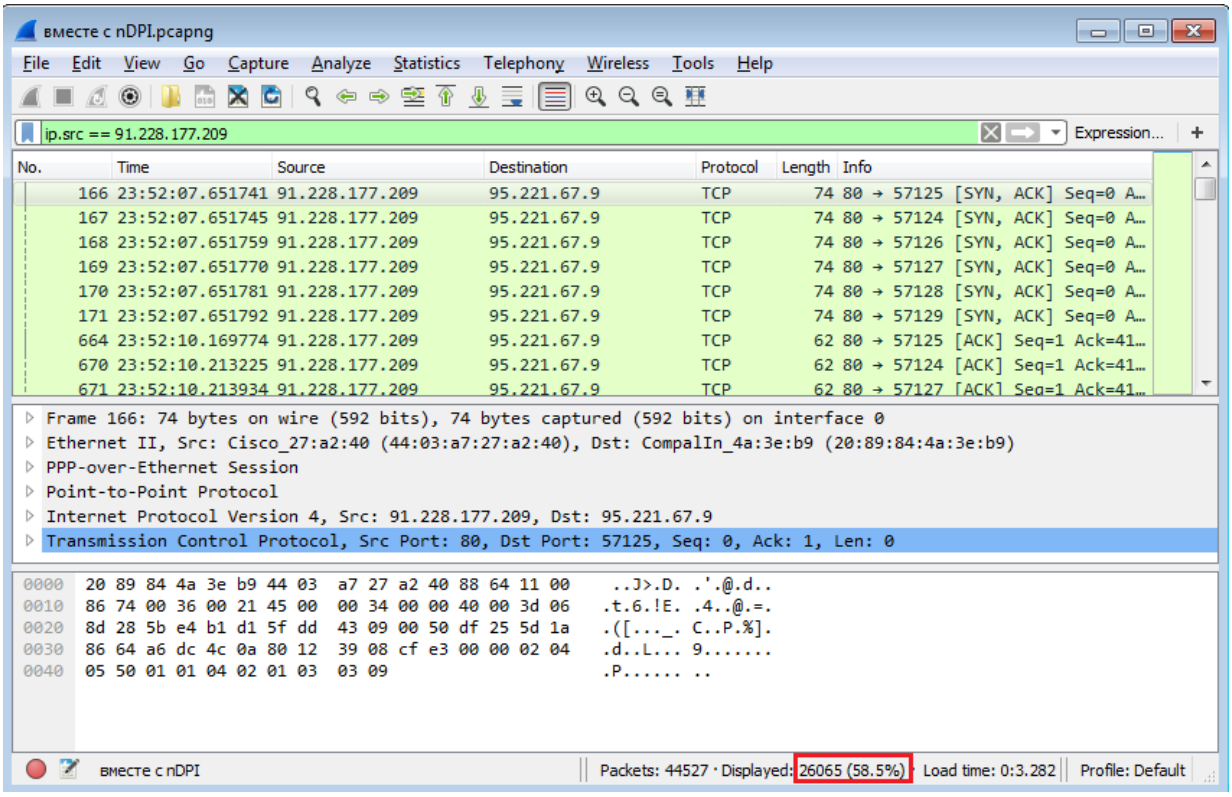


Рис.2.9 Количество пакетов, обнаруженных Wireshark с IP SPB TV

Курс содержит две лабораторные работы («Создание и внедрение сигнатуры», «Получение задержки определения потока пакетов приложения»).

Лабораторная работа (ЛР) 1. «Создание и внедрение сигнатуры»

Постановка задачи

На вход системы DPI поступает неизвестный поток пакетов, который распознается программой расшифровки трафика Wireshark. Требуется проанализировать поток пакетов, выявить сигнатуру, сформулировать сигнатуру, описать ее на языке C++, внедрить с систему n-DPI и проверить успешное определение потока пакетов.

П.1 Формирование сигнатуры

Для выявления сигнатуры неизвестного потока пакетов, следует снять трассировку данного потока пакетов с помощью программы Wireshark. Выписать фильтр для выбранного потока пакетов, состоящий из IP-адресов отправителя и получателя, транспортных портов отправителя и получателя, а так же типа протокола. Проанализировать критерии определения, на которые опирается Wireshark, и выписать соответствующие поля служебных заголовков. Найти характеристики (сигнатуры), точно определяющий приложение породившее данный поток пакетов.

Пример обнаружения DPI системой трафика BitTorrent

BitTorrent – это пиринговый (P2P) сетевой протокол для обмена файлами через Интернет. Файлы передаются по частям, каждый torrent-клиент, получая эти части, в то же время отдаёт их другим клиентам, что снижает нагрузку и зависимость от каждого клиента-источника и обеспечивает избыточность данных. Связь между клиентами пиринговой сети BitTorrent начинается с рукопожатия представленного на рис.2.10.

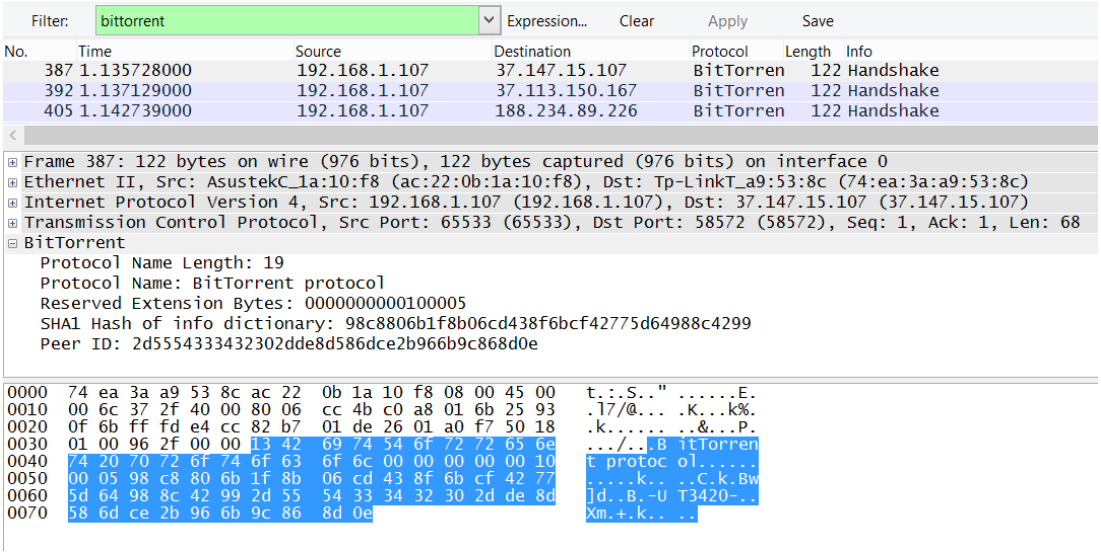


Рис.2.10 Пакет BitTorrent, полученный с помощью Wireshark

Заголовок BitTorrent при рукопожатии имеет следующий формат: <acharacter (1 byte)><astring (19 byte)>. Первый байт всегда фиксирован и имеет значение «19», а значение строки «BitTorrent protocol». Исходя из анализа поля данных, происходит идентификация BitTorrent-трафика:

1) Первый байт в полезной нагрузке TCP - 19 (0x13).

2) Следующие 19 байт соответствуют строке «BitTorrent protocol».

Таким образом, формат правила политики DPI системы для обнаружения и присвоения низшего приоритета будет таким.

Таблица 2.1

Формат правила политики для трафика протокола BitTorrent

Правило политики для трафика BitTorrent	
Условие	Действие
Если: У1: «L4+Тип протокола = TCP» И У2: «Первый байт в поле данных TCP = 0x13» И У3: «Следующие 19 байт в поле данных TCP = «BitTorrent protocol» »	Тогда: Д1: «Присвоить низший приоритет»

Пример обнаружения DPI системой трафика HTTP

HTTP (HyperTextTransferProtocol) — это протокол прикладного уровня, созданный для передачи данных в виде гипертекстовых документов в формате HTML. В настоящий момент протокол используется для передачи произвольных данных. В основе протокола HTTP лежит технология «клиент-сервер», то есть предполагается существование клиентов, которые инициируют соединение и посылают запрос, и серверов, которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

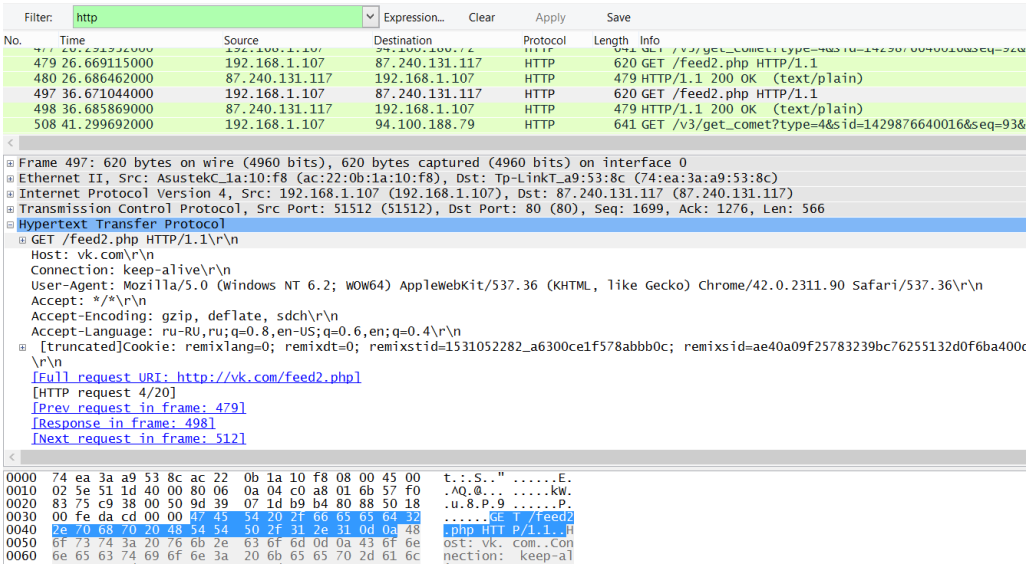


Рис.2.11. Пакет HTTP, полученный с помощью программы Wireshark

Для обнаружения протокола HTTP достаточно проверить, что на транспортном уровне использовался протокол TCP, и поле данных пакета TCP начинается с одной из следующих команд: «GET» (рис.2.11), «POST», «HEAD». Также, после одной из этих команд должен стоять пробел, а через некоторый промежуток встречается текст «HTTP/». Если всё это выполняется, то этот пакет несёт в себе HTTP запрос.

Таким образом, формат правила политики DPI системы для обнаружения и присвоения приоритета будет таким.

Таблица 2.2

Формат правила политики для трафика протокола HTTP

Правило политики для трафика HTTP	
Условие	Действие
Если: У1: «L4+Тип протокола = TCP» И У2: «Поле данных TCP = «GET» ИЛИ «POST» ИЛИ «HEAD» + «HTTP/» »	Тогда: Д1: «Присвоить приоритет»

DPI система не только обнаруживает пакеты протокола HTTP, но и узнает их тогда, когда HTTP используется для передачи других данных. Например, когда видео или P2Pтрафик передаётся поверх либо вместо HTTP.

После выделения ключевых заголовков или байт протокола, в свободной форме описать одну или несколько сигнатур определения приложения и его потока пакетов. Нарисовать алгоритм работы сигнатуры.

Пример алгоритма работы сигнатуры обнаружения OTT-видео Sopcast

А) Принимается пакет исследуемого потока и анализируется тип протокола транспортного уровня. Если протокол не относится к UDP или TCP, то пакет не является пакетом потока приложения Sopcast. Если на транспортном уровне применяется протокол TCP, но длина полезных данных не 54 байта, то тоже не Sopcast. Если TCP с длиной полезных данных 54 байта, то проверяется множество условий (рис.2.12). Условия для TCP можно разделить на 5 групп проверок соответствия взаимных значений среди поля передаваемых данных. Например, в первой группе проверяется, чтобы второй байт полезной нагрузки не был равен значению третьего байт полезной нагрузки +/- 4, а так же, чтобы второй байт полезной нагрузки не был равен значению четвертого байта полезной нагрузки +/- 1. И другие условия. Если пакет проваливает проверку одной из групп, то считается что он не относится к Sopcast. Если же пакет проходит все 5ть групп проверок, то он относится к

Sopcast. В случае передачи поверх UDP проводится анализ поля данных по одной из 7 групп проверок – для 8 различных длин пакетов UDP: 52, 80 (28, 96), 60, 42, 28, 286, 76 байт. Если длина пакета UDP не равна 52, то проводится проверка условий для длины 80, 28 или 96. Если длина пакета в этот раз тоже не соответствует, то для 60 байт и так далее. Например, в случае UDP пакета длиной 52 байта нулевой, первый и девятый байты полезной нагрузки должны иметь значение 0xFF, второй байт – значение 0x01, восьмой – 0x02, десятый, двенадцатый, тринадцатый и четырнадцатый – 0x00, одиннадцатый – 0x2c. Если все эти условия выполнены, то пакет относится к Sopcast.

П.2 Для внедрения сигнатуры в систему n-DPI, требуется описать ее на языке C++.

Пример сигнатуры Sopcast описанной на языке C++ для внедрения в n-DPI

```
#include "ndpi_protocols.h"
#ifdef NDPI_PROTOCOL_SOPCAST static void ndpi_int_sopcast_add_connection (struct
ndpi_detection_module_struct *ndpi_struct, struct ndpi_flow_struct *flow) { ndpi_set_detected_protocol
(ndpi_struct, flow, NDPI_PROTOCOL_SOPCAST, NDPI_PROTOCOL_UNKNOWN); }
```

Проверка в случае применения TCP:

```
#if !defined(WIN32) static inline else __forceinline static endif u_int8_t ndpi_int_is_sopcast_tcp (const u_int8_t *
payload, const u_int16_t payload_len){
```

Проверка в случае применения TCP для длины полезных данных равной 54 байта:

```
#if (payload_len != 54) return 0;
```

Второй байт полезной нагрузки не должен быть равен значению третьего байта полезной нагрузки +/- 4.

```
#if (payload[2] != payload[3] - 4 && payload[2] != payload[3] + 4)  return 0;
```

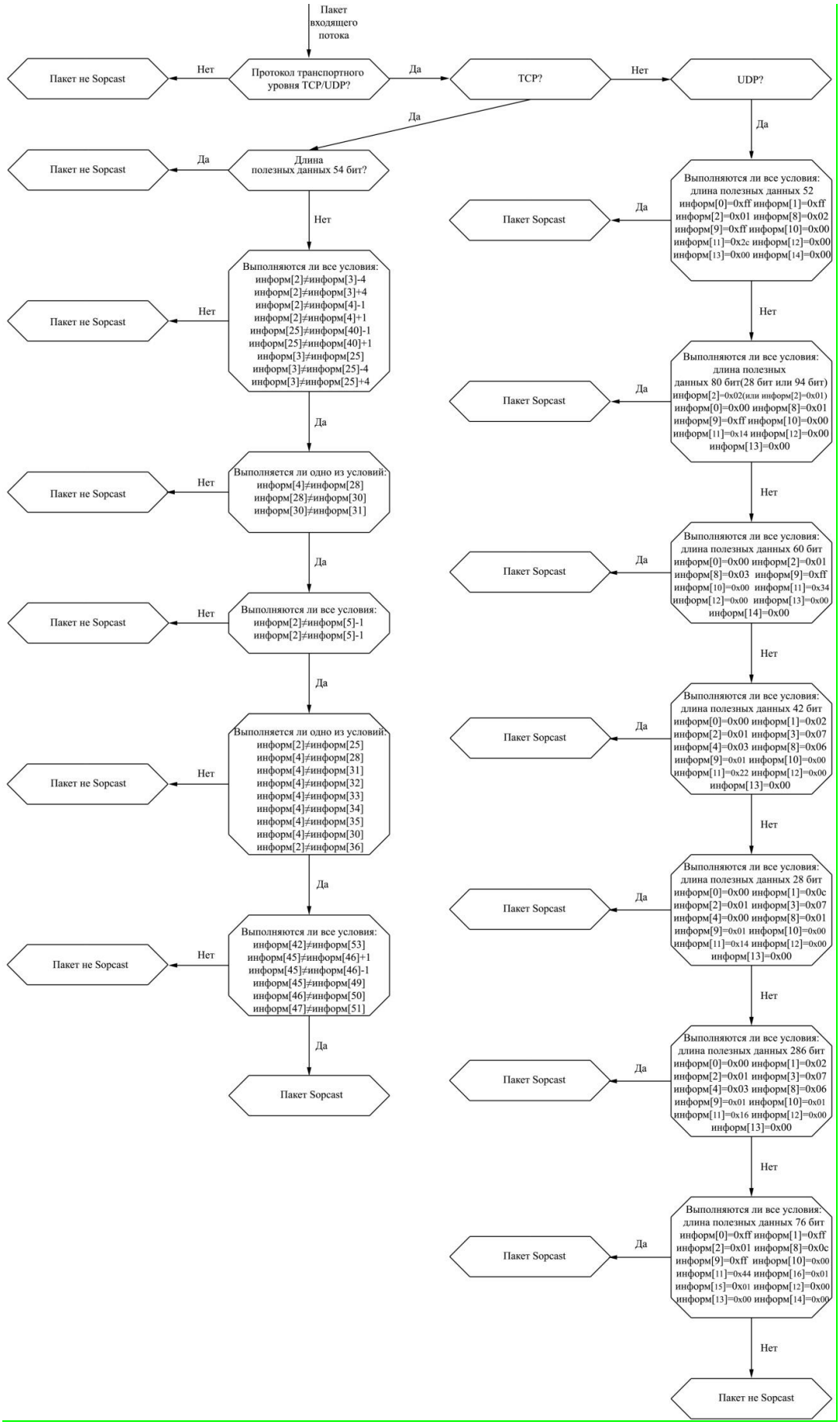


Рис.2.12 алгоритм работы сигнатуры обнаружения OTT-видео Sopcast

Второй байт полезной нагрузки не должен быть равен значению четвертого байта полезной нагрузки +/- 1.

```
#if (payload[2] != payload[4] - 1 && payload[2] != payload[4] + 1) return 0;
#if (payload[25] != payload[25 + 16 - 1] + 1 && payload[25] != payload[25 + 16 - 1] - 1) { if (payload[3] !=
payload[25] && payload[3] != payload[25] - 4 && payload[3] != payload[25] + 4 && payload[3] != payload[25] -
21){ return 0;} }
#if (payload[4] != payload[28] || payload[28] != payload[30] ||payload[30] != payload[31] || get_u_int16_t(payload,
30) != get_u_int16_t(payload, 32) ||get_u_int16_t(payload, 32) != get_u_int16_t(payload, 34)){ if ((payload[2] !=
payload[5] - 1 && payload[2] != payload[5] + 1) || payload[2] != payload[25] || payload[4] != payload[28] ||
payload[4] != payload[31] || payload[4] != payload[32] || payload[4] != payload[33] || payload[4] != payload[34] ||
payload[4] != payload[35] || payload[4] != payload[30] || payload[2] != payload[36]) { return 0; } }
#if (payload[42] != payload[53]) return 0;
#if (payload[45] != payload[46] + 1 && payload[45] != payload[46] - 1) return 0;
#if (payload[45] != payload[49] || payload[46] != payload[50] || payload[47] != payload[51]) return 0;
#return 1; }
#static void ndpi_search_sopcast_tcp (struct ndpi_detection_module_struct *ndpi_struct, struct ndpi_flow_struct
*flow){ struct ndpi_packet_struct *packet = &flow->packet; if (flow->packet_counter == 1 && packet-
>payload_packet_len == 54 && get_u_int16_t(packet->payload, 0) == ntohs(0x0036)){ if (ndpi_int_is_sopcast_tcp
(packet->payload, packet->payload_packet_len)){
```

Запись в систему журналирования о обнаружении TCP Sopcast пакета.

```
NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found sopcast TCP \n");
ndpi_int_sopcast_add_connection (ndpi_struct, flow);  return; } }
#NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "exclude sopcast TCP. \n");
#NDPI_ADD_PROTOCOL_TO_BITMASK (flow->excluded_protocol_bitmask, NDPI_PROTOCOL_SOPCAST);
}
```

Проверка в случае применения UDP для длины полезных данных равной 52 байта:

```
#static void ndpi_search_sopcast_udp (struct ndpi_detection_module_struct *ndpi_struct, struct ndpi_flow_struct
*flow){ struct ndpi_packet_struct *packet = &flow->packet;
#NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "search sopcast. \n");
#if (packet->payload_packet_len == 52
```

Нулевой, первый и девятый байты полезной нагрузки должны иметь значение 0xFF, второй байт – значение 0x01, восьмой – 0x02, десятый, двенадцатый, тринадцатый и четырнадцатый – 0x00, одиннадцатый – 0x2c. Если все эти условия выполнены, то пакет относится к Sopcast. Если пакет имеет другую длину, то производятся проверки из следующих 6 групп проверок для UDP.

```
&& packet->payload[0] == 0xff && packet->payload[1] == 0xff && packet->payload[2] == 0x01 && packet-
>payload[8] == 0x02 && packet->payload[9] == 0xff && packet->payload[10] == 0x00 && packet->payload[11]
== 0x2c && packet->payload[12] == 0x00 && packet->payload[13] == 0x00 && packet->payload[14] == 0x00) {
NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found sopcast with if I. \n");
ndpi_int_sopcast_add_connection(ndpi_struct, flow); return; }
```

Проверка в случае применения UDP для длины полезных данных равной 80 или 28 или 94 байта:

```
#if ((packet->payload_packet_len == 80 || packet->payload_packet_len == 28 || packet->payload_packet_len == 94)
&& packet->payload[0] == 0x00 && ( packet->payload[2] == 0x02 || packet->payload[2] == 0x01) && packet-
```

```
>payload[8] == 0x01 && packet->payload[9] == 0xff && packet->payload[10] == 0x00 && packet->payload[11]
== 0x14 && packet->payload[12] == 0x00 && packet->payload[13] == 0x00) {
NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found sopcast with if II. \n");
ndpi_int_sopcast_add_connection(ndpi_struct, flow); return; }
```

Проверка в случае применения UDP для длины полезных данных равной 60 байт:

```
#if (packet->payload_packet_len == 60 && packet->payload[0] == 0x00 &&packet->payload[2] == 0x01 &&
packet->payload[8] == 0x03 && packet->payload[9] == 0xff && packet->payload[10] == 0x00 && packet-
>payload[11] == 0x34 && packet->payload[12] == 0x00 && packet->payload[13] == 0x00 && packet-
>payload[14] == 0x00){ NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found
sopcast with if III. \n"); ndpi_int_sopcast_add_connection(ndpi_struct, flow);  return; }
```

Проверка в случае применения UDP для длины полезных данных равной 42 байта:

```
#if (packet->payload_packet_len == 42 && packet->payload[0] == 0x00 && packet->payload[1] == 0x02 &&
packet->payload[2] == 0x01 && packet->payload[3] == 0x07 && packet->payload[4] == 0x03 && packet-
>payload[8] == 0x06 && packet->payload[9] == 0x01 && packet->payload[10] == 0x00 && packet->payload[11]
== 0x22 && packet->payload[12] == 0x00 && packet->payload[13] == 0x00) { NDPI_LOG
(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found sopcast with if IV. \n");
ndpi_int_sopcast_add_connection(ndpi_struct, flow);  return; }
```

Проверка в случае применения UDP для длины полезных данных равной 28 байт:

```
#if (packet->payload_packet_len == 28 && packet->payload[0] == 0x00 && packet->payload[1] == 0x0c &&
packet->payload[2] == 0x01 && packet->payload[3] == 0x07 && packet->payload[4] == 0x00 && packet-
>payload[8] == 0x01 && packet->payload[9] == 0x01 && packet->payload[10] == 0x00 && packet->payload[11]
== 0x14 && packet->payload[12] == 0x00 && packet->payload[13] == 0x00) { NDPI_LOG
(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found sopcast with if V. \n");
ndpi_int_sopcast_add_connection(ndpi_struct, flow);  return; }
```

Проверка в случае применения UDP для длины полезных данных равной 286 байт:

```
#if (packet->payload_packet_len == 286 && packet->payload[0] == 0x00 && packet->payload[1] == 0x02 &&
packet->payload[2] == 0x01 && packet->payload[3] == 0x07 && packet->payload[4] == 0x03 && packet-
>payload[8] == 0x06 && packet->payload[9] == 0x01 && packet->payload[10] == 0x01 && packet->payload[11]
== 0x16 && packet->payload[12] == 0x00 && packet->payload[13] == 0x00) { NDPI_LOG
(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found sopcast with if VI. \n");
ndpi_int_sopcast_add_connection(ndpi_struct, flow);  return; }
```

Проверка в случае применения UDP для длины полезных данных равной 76 байт:

```
#if (packet->payload_packet_len == 76 && packet->payload[0] == 0xff && packet->payload[1] == 0xff &&
packet->payload[2] == 0x01 && packet->payload[8] == 0x0c && packet->payload[9] == 0xff &&  packet-
>payload[10] == 0x00 && packet->payload[11] == 0x44 && packet->payload[16] == 0x01 && packet-
>payload[15] == 0x01 && packet->payload[12] == 0x00 && packet->payload[13] == 0x00 && packet-
>payload[14] == 0x00){ NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "found
sopcast with if VII. \n"); ndpi_int_sopcast_add_connection(ndpi_struct, flow); return; }
#NDPI_LOG(NDPI_PROTOCOL_SOPCAST, ndpi_struct, NDPI_LOG_DEBUG, "exclude sopcast. \n");
#NDPI_ADD_PROTOCOL_TO_BITMASK(flow->excluded_protocol_bitmask, NDPI_PROTOCOL_SOPCAST);
}
```

```
#void ndpi_search_sopcast(struct ndpi_detection_module_struct *ndpi_struct, struct ndpi_flow_struct *flow) {
struct ndpi_packet_struct *packet = &flow->packet; if (packet->udp != NULL) ndpi_search_sopcast_udp
(ndpi_struct, flow); if (packet->tcp != NULL)  ndpi_search_sopcast_tcp(ndpi_struct, flow); }
#void init_sopcast_dissector(struct ndpi_detection_module_struct *ndpi_struct, u_int32_t *id,
NDPI_PROTOCOL_BITMASK *detection_bitmask) { ndpi_set_bitmask_protocol_detection("Sopcast",
ndpi_struct, detection_bitmask, *id, NDPI_PROTOCOL_SOPCAST, ndpi_search_sopcast,
NDPI_SELECTION_BITMASK_PROTOCOL_V4_V6_TCP_OR_UDP_WITH_PAYLOAD,
SAVE_DETECTION_BITMASK_AS_UNKNOWN, ADD_TO_DETECTION_BITMASK); *id += 1; }
#endif
```

Пример сигнатуры SSH описанной на языке C++ для внедрения в n-DPI

Когда приходит первый пакет, проверяется длина полезной нагрузки. Если она составляет от 7 до 100 байт и указана 4 версия SSH (рис.2.13), то в лог nDPI записывается обнаружение протокола SSH и указание на прием первого пакета. Когда приходят последующие пакеты, проводится аналогичная проверка длины поля нагрузки и версии протокола, после чего записывается обнаружение еще одного пакета SSH и указание на обнаруженный поток пакетов.

```
if (flow->l4.tcp.ssh_stage == 0) {
        if (packet->payload_packet_len > 7 && packet->payload_packet_len < 100
                && memcmp(packet->payload, "SSH-", 4) == 0) {
                NDPI_LOG(NDPI_PROTOCOL_SSH, ndpi_struct, NDPI_LOG_DEBUG, "ssh stage 0 passed\n");
                flow->l4.tcp.ssh_stage = 1 + packet->packet_direction;
                return;
        }
} else if (flow->l4.tcp.ssh_stage == (2 - packet->packet_direction)) {
        if (packet->payload_packet_len > 7 && packet->payload_packet_len < 100
                && memcmp(packet->payload, "SSH-", 4) == 0) {
                NDPI_LOG(NDPI_PROTOCOL_SSH, ndpi_struct, NDPI_LOG_DEBUG, "found ssh\n");
                ndpi_int_ssh_add_connection(ndpi_struct, flow);
                return;
```

Рис.2.13 Программный код C++ сигнатуры SSH

Построение схемы эксперимента

n-DPI – СПО выполняющие функцию глубокого анализа потока пакетов.

Для проверки работоспособности сигнатуры, требуется применить сервер с установленной системой DPI. Для этого можно воспользоваться лабораторным сервером, или установить систему на собственный компьютер (сервер).

П.3 Установить n-DPI

Установка и настройка утилит

А. Установка системы под ОС Windows.

Первый шаг к работе с nDPI это установка ntopng. Для этого с главной страницы сайта http://www.ntop.org через кнопку «download» выполнить переход к странице загрузки. Далее после нажатия кнопки «available here» (рис.2.14) выбрать операционную систему, на которой будет запускаться nDPI. И после перехода на вкладку нужной ОС, скачать архив, из которого установить ntopng (рис.2.15).

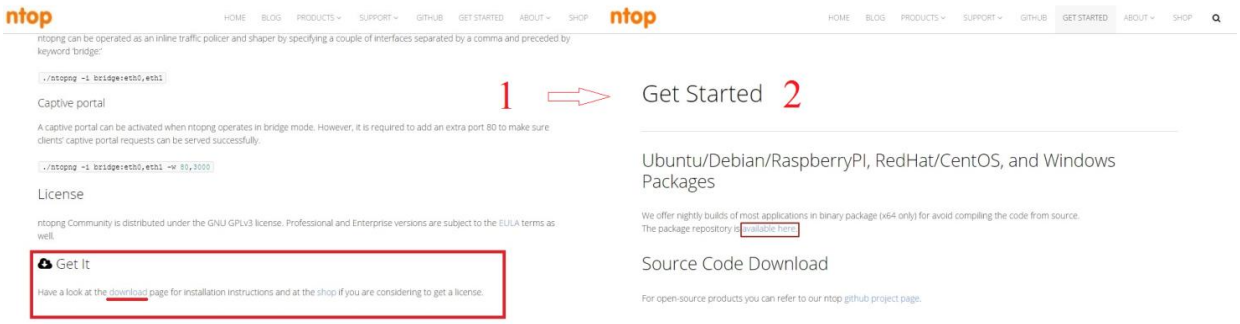


Рис.2.14 Ссылка на установку nTOP

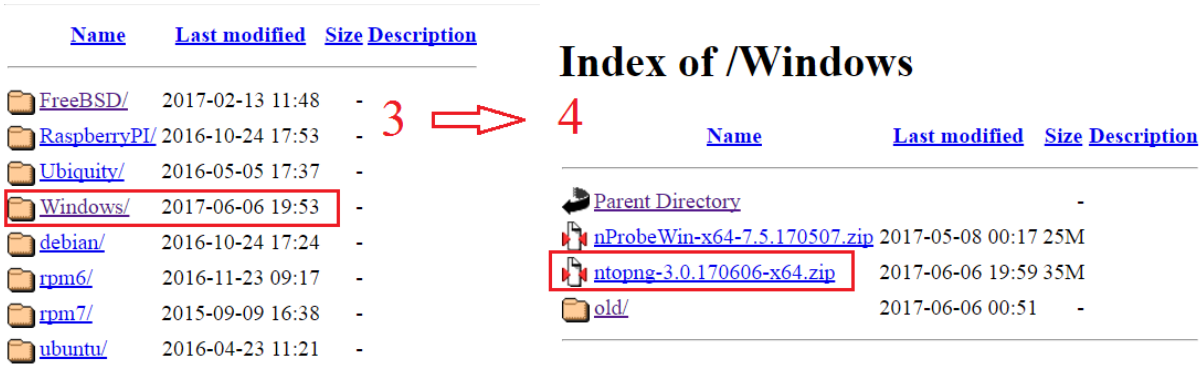


Рис.2.15 Выбор ОС Windows и архива nTOP

Для того, чтобы в ntop заработал непосредственно сам nDPI, на сайте http://www.ntop.org надо осуществить переход в раздел «github» (https://github.com/ntop).

На странице github следует нажать nDPI и загрузить страницу https://github.com/ntop/nDPI , откуда скачать архив содержащий нормативные документы, инструкции по работе с nDPI и официально разработанные и внедренные в nDPI сигнатуры. Для скачивания архива следует нажать кнопку «Clone or download», а затем «Download ZIP» На рис.2.16 показан порядок перехода к загрузке архива, а так же его содержимое.

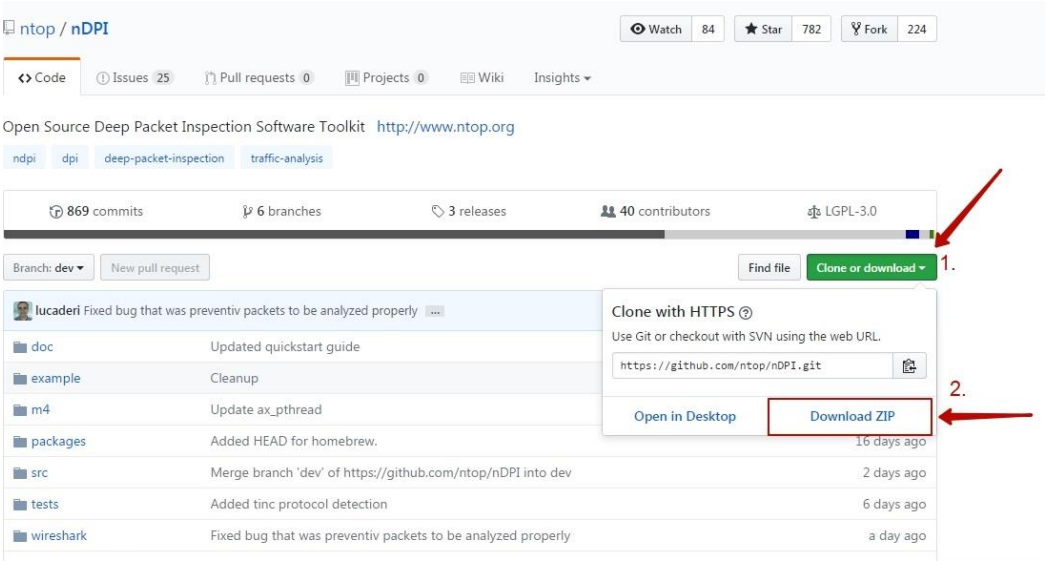


Рис.2.16 Архив установки nDPI

Для запуска nTOP понадобится redis server. Его можно скачать с главного сайта разработчика https://redis.io/ в разделе «download». Откуда

открывается репозиторий redis для ОС Windows https://github.com/MicrosoftArchive/redis и перейти по ссылке для готовых релизов: https://github.com/MicrosoftArchive/redis/releases . Где скачать архив или установщик для ОС Windows.

Если не удается найти собранный установщик nDPI под ОС Windows можно использовать эмулятор консоли Cygwin (https://cygwin.com/install.html) и кроссплатформенный компилятор MinGW 32 (http://www.mingw.org/wiki/LinuxCrossMinGW).

Алгоритм запуска nTOP:

1. Запустить redis-server;
2. Зайти в службы windows (рис.2.17), запустить там ntopng;

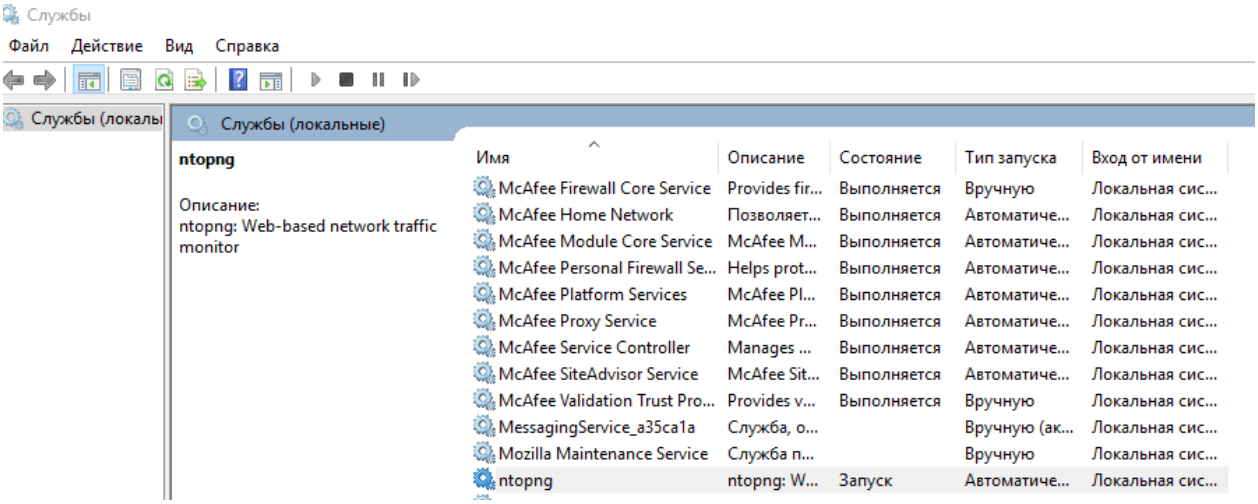


Рис.2.17 Службы ОС Windows и запуск ntopng

3. Веб-интерфейс ntopng доступен по адресу компьютера в сети или localhost адресу, с указанием порта 3000. Таким образом необходимо в адресной строке браузера прописать 127.0.0.1:3000/lua (логин admin, пароль по умолчанию admin – рис.2.18).

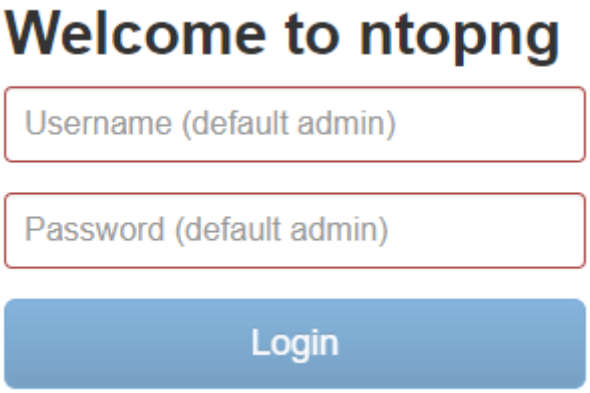


Рис.2.18 Авторизация ntopng

После правильно выполненных действий, приведенных выше, получаем доступ к графическому интерфейсу ntop.

В. Быстрая установка системы под ОС Linux.
Инструкции по установке указаны на сайте разработчиков по адресу http://www.ntop.org/get-started/download/ .
При необходимости установить PF-Ring:

```
#git clone https://github.com/ntop/PF_RING.git /home/administrator/PF_RING/
#apt-get install build-essential bison flex linux-headers-$(administrator -r) libnuma-dev
#apt-get install git autoconf automake autogen libpcap-dev libtool
#cd /home/administrator/PF_RING
#make
#cd /home/administrator/PF_RING/kernel
#make install
#sudo insmod ./pf_ring.ko
#cd /home/administrator/PF_RING/userland/lib/
#make install
```

Далее воспользоваться установкой netfilter и/или готовой сборкой ntopng и nDPI.

А.1 Скачать архив nDPI основанный на репозитарии GIT commit – ветка репозитария netfilter.

Либо на сайте: github.com/vel21ripn/nDPI и ветка netfilter

Либо путем выполнения команды в программе репозитария git:

```
#git clone -b netfilter https://github.com/vel21ripn/nDPI.git /home/administrator/nDPI/
```

Далее следует перейти в скаченный и распакованный каталог nDPI и запустить в нем скрипт распаковки autogen.sh:

```
#cd nDPI
#./autogen.sh
#cd ndpi-netfilter
```

Если версия ядра ОС меньше 3.10 и ядро еще не пропатчено, то следует его пропатчить – для этого:

Указать NF_CONNTRACK_CUSTOM=2,

Пересобрать ядро ОС и перезагрузиться:

```
#W=`pwd`
#cd /usr/src/linux
#patch -p1 <$W/kernel-patch/XXXX.diff
#make menuconfig
#make
```

Далее следует собрать расширение iptables (iptables-extension) из папки ndpi-netfilter:

```
#make
#make install
#make modules_install
#conntrack -F && rmmod xt_ndpi
```

А.2 Другой вариант- непосредственно использовать сборку ntopng и nDPI:

```
#git clone https://github.com/ntop/ntopng.git /home/administrator/ntopng/
#cd ntopng
#./autogen.sh
#./configure
#make
#make install

#git clone https://github.com/ntop/nDPI.git /home/administrator/nDPI/
#apt-get install build-essential
#apt-get install git autoconf automake autogen libpcap-dev libtool
#cd nDPI
#./autogen.sh
#./configure –with-pic
#make
```

В. Подробные пошаговые инструкции настройки системы.

Установка необходимых пакетов библиотек для операционной системы Linux: libncurses5-dev, libnfnetlink-dev, linuxdoc-tools, kernel-package, dh-autoreconf, dkms, iptables-dev, bzr.

```
#cd /usr/src/ && sudo apt-get install libncurses5-dev  libnfnetlink-dev linuxdoc-tools kernel-package dh-autoreconf
dkms iptables-dev bzr
#sudo apt-get build-dep iptables
```

Применение IMQ патча:

```
#sudo wget http://devel.aanet.ru/ndpi/linux-imqmq-3.17.3.patch.xz
```

```
#sudo ln -s /usr/src/linux-3.17.8 /usr/src/linux
#cd /usr/src/linux
#sudo xz -dc ../linux-imqmq-3.17.3.patch.xz | sudo patch -p1
```

Установка iptables:

```
#sudo apt-get install iptables
```

Установка xtables-addons, необходимого для nDPI:

```
#sudo bzr branch lp:ubuntu/xtables-addons
#cd xtables-addons
#sudo dpkg-buildpackage -b -us -uc
#sudo dpkg -i ../xtables-addons*.deb
```

Установка nDPI:

```
#sudo wget http://devel.aanet.ru/ndpi/nDPI-1.5.1.20150513.tar.gz
#sudo tar -xvzf nDPI-1.5.1.20150513.tar.gz && cd ./nDPI-1.5.1.20150513 #sudo ./autogen.sh
#cd ./ndpi-netfilter
#sudo make
#sudo make install
#sudo make modules_install
#sudo modprobe xt_ndpi
```

Проверки корректности:

```
#lsmod | grep xt_ndpi
```

После завершения установки, требуется настройка совместной работы утилит. Настройка Iptables:

```
#iptables -A INPUT -t mangle -m ndpi --proto bittorrent -j MARK --set-mark 10
#iptables -A INPUT -t mangle -m ndpi --proto skype -j MARK --set-mark 20
#iptables -A OUTPUT -t mangle -m ndpi --proto skype -j MARK --set-mark 20
#iptables -A OUTPUT -t mangle -m ndpi --proto http -j MARK --set-mark 30
```

Создание древа очередей:

```
#tc qdisc add dev eth0 root handle 1: htb default 30
#tc class add dev eth0 parent 1: classid 1:1 htb rate 5mbit ceil 5mbit burst 15000
#tc class add dev eth0 parent 1:1 classid 1:10 htb rate 4mbit ceil 5mbit burst 15000
#tc class add dev eth0 parent 1:1 classid 1:20 htb rate 1kbit ceil 10kbit burst 15000
#tc class add dev eth0 parent 1:1 classid 1:30 htb rate 100kbit ceil 100kbit burst 15000
```

Установка фильтров:

```
#tc filter add dev eth0 parent 1:1 handle 10 fw classid 1:10
#tc filter add dev eth0 parent 1:1 handle 20 fw classid 1:20
#tc filter add dev eth0 parent 1:1 handle 30 fw classid 1:30
```

П.4 Далее следует направить поток данных на сервер n-DPI. Для этого следует настроить отправку данных на компьютере источнике потока пакетов через сервер DPI. Укажите маршрут к серверу DPI.

П.5 Для проверки работоспособности DPI с компьютера источника данных следует открыть один из сайтов, и убедиться, что DPI зафиксировал появление нового потока пакетов от браузера.

П.6 После этого инициировать неизвестный поток пакетов и убедиться, что DPI его не распознал.

П.7 Внедрить формализованную на языке C++ сигнатуру в n-DPI.

Повторить П.6 и убедиться, что DPI успешно распознал сигнатуру.

Задание

З.1 Проанализируйте процесс поиска сигнатуры и примеры представления сигнатуры на языке C++.

З.2. Выберете новый еще неизвестный для n-DPI поток пакетов, и сформируйте сигнатуру для его обнаружения.

З.3. Доработайте существующий метод определения потока пакетов путем обнаружения дополнительной сигнатуры, чтобы повысить точность анализа.

З.4 Внедрите новую сигнатуру в n-DPI, проанализируйте работу системы.

З.5 Составьте отчет о выполнении лабораторной работы [6].

Лабораторная работа 2. «Получение задержки определения потока пакетов приложения»

Постановка задачи

На вход системы DPI поступает поток пакетов. Требуется зафиксировать задержку определения приложения, породившего поток пакетов, а так же

проанализировать ресурсоемкость работы системы DPI и сформулировать предложения по совершенствованию метода получения задержки и скорости проведения анализа системой n-DPI.

Построение схемы эксперимента

n-DPI – СПО выполняющие функцию глубокого анализа потока пакетов.

Для получения задержки определения потока пакетов приложения, требуется применить сервер с установленной системой DPI. Для этого можно воспользоваться лабораторным сервером, или установить систему на собственный компьютер (сервер). Установка сервера n-DPI выполняется согласно ЛР1 П.3-5.

Вычисление задержки определения потока

В момент выполнения ЛР1 П.5 следует произвести запись трассировки с помощью программ tcpdamp, tshark или wireshark на сервере n-DPI. Зафиксировать время обнаружения потока пакетов системой n-DPI, используя пользовательский веб-интерфейс и систему журналирования. Проанализировать полученный .pcap файл с точки зрения времени поступления пакетов, в том числе изучаемого потока пакетов. На основе данных времени поступления пакетов изучаемого потока и данных времени обнаружения приложения, порождающего поток, получить задержку определения потока пакетов.

Задание

З.1.Проанализируйте процесс вычисления задержки определения потока.

З.2.Выберете не изученный с точки зрения задержки определения в системе n-DPI поток пакетов, и получите задержку для него.

3.3.Проанализируйте работу n-DPI и сформулируйте предложения по совершенствованию метода вычисления задержки или повышению скорости анализа системы DPI.

3.4.Определите наиболее ресурсоемкую работу системы DPI, и зафиксируйте потребление аппаратных ресурсов в момент определения выбранного потока пакетов.

3.5.Составьте отчет о выполнении лабораторной работы.

ЛИТЕРАТУРА

Основная

1. Гольдштейн Б.С., Соколов Н.А., Яновский Г.Г. Сети связи: учебник для вузов. - СПб.: БХВ-Петербург, 2011.
2. OpenFlow Switch [Электронный ресурс] // Open Networking Foundation: specification. 26 March, 2015. Version 1.5.1. URL: https://www.opennetwork-ing.org/images/stories/downloads/sdn-resources/onf-specifications/open-flow/openflow-switch-v1.5.1.pdf. (Дата обращения: 27.03.16).
3. Network Configuration Protocol (NETCONF) [Электронный ресурс] // IETF: RFC 6241. June, 2011.
4. Елагин В.С., Зобнин А.А. Аспекты реализации системы законного перехвата трафика в сетях SDN // Журнал "Вестник связи" - М, 2016. - №12. С.6-9
5. Елагин В.С. Подходы к моделированию систем законного перехвата трафика в SDN // V международная научно-техническая и научно-методическая конференция «Актуальные проблемы инфотелекоммуникаций в науке и образовании»: сб. науч. ст. /под. Ред. С.В. Бачевского. СПб. : СПБГУТ. 2016. С. 353- 358.
6. Елагин В.С., Сорокин В.А. Исследование технологических возможностей внедрения СОРМ в SDN // Журнал «Труды учебных заведений связи» - СПб. : СПБГУТ. 2017. С. 28-35

Дополнительная

7. Намиот Д. Е. Интерфейсы прикладного уровня в SDN [Электронный ре-сурс] // Современные информационные технологии и ИТ-образование: научн. статья. 2015. № 11. URL: http://istina.msu.ru/publications/article/11664576/. (Дата обращения: 13.03.16).
8. Технологии реализации программно-конфигурируемых сетей. [Электрон-ный ресурс] // Журнал сетевых решений LAN: научн. журн. 2014. № 4. URL: http://www.osp.ru/lan/2014/04/13040709. (Дата обращения: 21.02.16).
9. А.Б. Гольдштейн, Б.С. Гольдштейн. Softswitch – СПб: БХВ – Санкт-Петербург, 2006.
10. SDN architecture for cable access network [Электронный ресурс] //

Cable-Labs: technical report. 26 June, 2015. URL: http://www.cablelabs.com/specifi-cation/sdn-architecture-technical-report/. (Дата обращения: 03.04.16).

11. 14. The role of DPI in an SDN world [Электронный ресурс] // QOSMOS: white paper. December, 2012; URL: https://www.sdxcentral.com/wp-content/up-loads/2012/12/Qosmos_DPI-SDN-WP_Dec-2012.pdf. (Дата обращения: 15.05.16).

Статьи

12. Ефимушкин В. А. Международная стандартизация программно-конфигурируемых сетей // Электросвязь. 2014. №8.
13. Крюков Ю. С. Законный перехват IP-трафика. Международный опыт // Защита информации. INSIDE: информационно-метод. журн. 2005. №8.
14. Шалимов А. В. Каким должен быть контроллер SDN // Вестник связи: научн. журн. 2014. № 4.
15. Гольдштейн Б.С., Елагин В.С., Крюков Ю.С., Семенов Ю.Н. Новая парадигма законного перехвата сообщений в NGN/IMS // Вестник связи, № 4, 2010. С. 38-46.

Сайты

16. http://www.skri.sut.ru
17. http://www.protei.ru
18. http://www.niits.ru
19. http://www.sotsbi.spb.ru
20. http://www.seventest.ru
21. http://www.ietf.org

Гольдштейн Борис Соломонович
Елагин Василий Сергеевич
Селиванов Андрей Евгеньевич

ПРОГРАММНО-КОНФИГУРИРУЕМЫЕ СЕТИ (SDN). ПРОТОКОЛ OPENFLOW

Учебное пособие

Редактор ...

Компьютерная верстка ...

План издания 2016 г., п. …

Подписано к печати 12.07.2017
Объем … усл.-печ. л. Тираж … экз. Заказ …

Редакционно-издательский отдел СПбГУТ
191186 СПб., наб. р. Мойки, 61

Отпечатано в СПбГУТ