Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра геометрии и компьютерных наук

НАСТРОЙКА И ИЗМЕРЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

Методические указания

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по программам высшего образования по направлению подготовки 02.04.02 Фундаментальная информатика и информационные технологии

Оренбург
2017

УДК 004.72 (076.5)
ББК 32.973-018я7
Н32

Рецензент – кандидат технических наук Д. В. Горбачев
Авторы: Ю. А. Ушаков, М. В. Ушакова, П. Н. Полежаев, А. Л. Коннов

Н32 Настройка и измерение производительности программно-конфигурируемых сетей : методические указания / Ю. А. Ушаков, М. В. Ушакова, П. Н. Полежаев, А.Л. Коннов; Оренбургский гос. ун-т. – Оренбург : ОГУ, 2017.

Методические указания содержат описание теоретических основ функционирования программно-конфигурируемых сетей, принципов их создания, а также методов измерения их производительности с помощью программных инструментов.

Методические указания предназначены для выполнения лабораторных работ по дисциплине «Компьютерные коммуникации и сети» для магистров направления подготовки 02.04.02 Фундаментальная информатика и информационные технологии.

УДК 004.72 (076.5)
ББК 32.973-018я7

Содержание

Введение

В настоящее время услугами вычислительных центров обработки данных (ЦОД) широко пользуются российские и иностранные компании с целью размещения своих научных и бизнес-приложений, что позволяет им избежать расходов на создание и поддержание функционирования собственных ЦОД. С другой стороны, владельцы ЦОД путем консолидации вычислительных ресурсов и систем хранения данных (СХД) способны снизить совокупную стоимость владения ИТ-инфраструктурой за счет обслуживания значительного количества клиентов, а также использования эффективных технических средств балансировки и распределения нагрузки, управления пересылкой данных в сети, интеграции нескольких территориально разрозненных ЦОД.

Существующие сетевые протоколы вычислительных ЦОД Fiber Channel, Infiniband, а также традиционный Ethernet, имеют ограниченные возможности по управлению трафиком и QoS. Усовершенствованные варианты протокола Ethernet – Converged Enhanced Ethernet и Cisco Data Center Ethernet включают расширения по управлению потоками на основе приоритетов, разделению пропускной способности, управлению перегрузками и логическим состоянием полос передачи данных, по обеспечению передачи без потерь, а также по одновременному использованию нескольких параллельных путей передачи данных между узлами. Основные недостатки данных решений – сложная децентрализованная схема управления потоками данных, основанная на множестве закрытых протоколов, значительная стоимость сетевого оборудования, сложность его расширения. Отдельно следует заметить, что данные решения используют реактивную схему управления потоками, когда решения по коммутации принимаются во время передачи потока.

Весьма перспективным направлением технологий вычислительных ЦОД является использование программно-конфигурируемых сетей (ПКС – Software Defined Network). Принципы ПКС впервые возникли в исследовательских лабораториях Стэнфорда и Беркли, и в настоящее время развиваются в рамках

консорциума Open Network Foundation и европейского проекта OFELIA. Известен положительный опыт компаний Google и Amazon по внедрению ПКС в ЦОД.

В основе подхода ПКС лежит возможность динамически управлять пересылкой данных в сети с помощью открытого протокола OpenFlow. Все активные сетевые устройства объединяются под управлением сетевой операционной системы (СОС), которая обеспечивает приложениям доступ к управлению сетью. Сетевые контроллеры могут быть централизованными, использовать общие абстракции для пересылки пакетов.

За счет управления пересылкой данных в сети использование ПКС в ЦОД позволяет реализовать схемы одновременной многопутевой передачи данных, управление потоками на основе приоритетов, виртуализацию сети, обеспечить QoS, эффективно распределить нагрузку на сеть. Открытость стандарта OpenFlow и вынесение логики управления в отдельный контроллер упрощает программное и аппаратное обеспечение активного сетевого оборудования, что позволит в будущем производителям снизить его стоимость, а, следовательно, уменьшатся затраты на создание ЦОД. Централизация и открытость средств управления ПКС позволяет гибко и эффективно адаптировать работу ЦОД под возникающие потребности бизнеса, что ускоряет внедрение инноваций и обеспечивает конкурентоспособность компаний.

В настоящее время отсутствуют решения на основе ПКС для вычислительных ЦОД, направленных на управление потоками данных вычислительных задач с целью их топологической локализации и снижения сетевой конкуренции между ними, что позволит повысить эффективность работы распределенного вычислительного ЦОД.

Наличие потребности компаний в создании эффективных распределенных вычислительных ЦОД, включающих СХД, на основе ПКС, перспективный характер технологии ПКС, а также отсутствие готовых решений в этой области являются существенными факторами, определяющими актуальность изучения ПКС и получения практических навыков работы с ними.

1 Теоретические основы программно-конфигурируемых сетей

1.1 Общие сведения об управляемых сетях

Большинство экспертов уверены в том, что технология Ethernet станет основой для построения единой, конвергентной сетевой инфраструктуры ЦОД. Институт IEEE разрабатывает стандарты, которые должны обеспечить гарантированную полосу пропускания и контроль потока для трафика с учетом его приоритета, а также более усовершенствованные механизмы управления потоками. С этими нововведениями Ethernet сможет заменить в ЦОД существующие сети Fibre Channel [1] и Infiniband [2], а также существенно повысить производительность работы новых Storage Area Network (SAN) систем на основе iSCSI.

Поддержка мобильности виртуальных машин, технология Fibre Channel over Ethernet (FCoE) [3] и другие новые приложения требуют построения крупномасштабных сетей, функционирующих на втором уровне модели OSI.

Постоянно совершенствуясь, Ethernet превратилась сегодня в наиболее масштабируемую и экономически эффективную технологию для построения сетей TCP/IP.

Трафик локальных вычислительных сетей (ЛВС) менее всего зависит от характеристик сети, но активно использует свойственные Ethernet-технологии широковещательной/групповой рассылки, виртуальных ЛВС (Virtual Local Area Network, VLAN). Для эффективной работы SAN необходима малая задержка и передача пакетов без потерь. Для реализации высокопроизводительных вычислительных систем (High-Performance Computing Cluster) требуется еще меньшая задержка и широкая полоса пропускания. Последние достижения в области технологии iSCSI привели к тому, что все больше заказчиков рассматривают ее как надежное решение для сетей SAN.

Благодаря своей гибкости, Ethernet сегодня доминирует в большинстве сетевых сред. Современные технологии обеспечивают скорость 10 Гбит/с (на один канал), в ближайшем будущем скорость возрастет до 100-Гбит/с, а в более отдаленной перспективе возможно появление терабитного варианта Ethernet. В

результате в настоящее время Ethernet рассматривается в качестве основного кандидата для конвергентной инфраструктуры ЦОД.

Традиционные Ethernet сети обладают архитектурными недостатками, связанными с принципом доступа к несущей среде: непредсказуемым временем задержки, увеличением нагрузки на центральные процессоры серверов, снижением производительности существующей инфраструктуры в результате использования части пропускной способности сети для передачи трафика сети хранения; проблемами с обеспечением безопасности передаваемого трафика. Традиционная технология Ethernet не пригодна для передачи критичного к потерям трафика, в частности, трафика сетей хранения, не может играть роль надежного транспорта канального уровня.

Над вопросом усовершенствования традиционного протокола Ethernet на протяжении ряда лет работали несколько организаций. Итогом их труда стал набор усовершенствований, получивших название Converged Enhanced Ethernet (CEE), Data Center Bridging (DCB) [4] и Cisco Data Center Ethernet (DCE) [5]. В их основе лежат одни и те же базовые спецификации. Вместе с тем необходимо учитывать, что DCE содержит расширенный набор возможностей по сравнению с CEE и DCB.

Data Center Ethernet представляет собой архитектуру на базе набора открытых стандартов расширений Ethernet, которая предназначена для улучшения и расширения функционала классического Ethernet в соответствии с требованиями, предъявляемыми к Ethernet как к конвергентному транспорту современного ЦОД.

DCE включает в себя два основных компонента: набор расширений Ethernet и аппаратные средства, обеспечивающие внутреннюю передачу трафика без потерь (Lossless Ethernet Switch Fabric).

Термин Lossless Ethernet определяет тип коммутаторов Ethernet, которые имеют ряд дополнительных функциональных возможностей, основной из которых является возможность передачи данных без потерь, вследствие перегрузок сети (lossless). В lossless сетях, достаточно важно блокировать операции I/O, поскольку в отличие от TCP/IP потеря одного пакета, обычно ведет к прерыванию передачи последовательности данных и требует повторной передачи всей последовательности

протоколом верхнего уровнявместо повторной передачи только нескольких потерянных блоков.

Cisco Data Center Ethernet (Cisco DCE) включает в себя два основных компонента: набор расширений Ethernet и аппаратные средства, обеспечивающие внутреннюю передачу трафика без потерь. Набор расширений Cisco DCE содержит как обязательный, так и опциональный функционал.

К обязательному функционалу относятся следующие расширения:

а) механизм управления потоком на основе приоритетов (Priority-based Flow Control, PFC) [6]. PFC расширяет функционал стандартного механизма PAUSE (описанного в стандарте IEEE 802.3x). Если механизм PAUSE вызывает прекращение передачи всего трафика по каналу Ethernet, то механизм PFC разделяет его на восемь виртуальных полос (virtual line) и позволяет управлять передачей трафика на основе приоритетов раздельно для каждой линии. Таким образом, можно создать линию без потерь (losslesslane) для чувствительного к потерям трафика (такого как Fibre Channel) и использовать остальные линии в стандартном режиме сброса пакетов для обычного трафика IP. Механизм Priority-based Flow Control описан в стандарте IEEE 802.1Qbb;

б) Enhanced Transmission Selection (ETS) обеспечивает управление разделением пропускной способности консолидированного канала для разных типов линий, решая задачу совместной передачи разных типов трафика без ущерба для качества. Этот инструмент описан в стандарте IEEE 802.1Qaz [7];

в) Data Center Bridging eXchange (DCBX) отвечает за обнаружение и автоматическое согласование ряда параметров, включая управление полосой и потоком по классам, а также управление перегрузками и логическим состоянием полос. Кроме того, с помощью механизма DCBX взаимодействующие устройства определяют совместимость соседнего устройства с DCE, т.е. очерчивают логическую границу домена DCE в сети ЦОД. Механизм Data Center Bridginge Xchange описан в стандарте IEEE 802.1Qaz;

г) Layer 2 Multi-Pathing (L2MP). В отличие от классического протокола SpanningTree, блокирующего множественные соединения между узлами, механизм

L2MP обеспечивает возможность одновременного использования нескольких параллельных путей, благодаря чему пропускная способность расходуется более эффективно. Механизм Layer 2 Multi-Pathing описан в стандарте IEEE 802.1Qau [8].

Вторая составляющей архитектуры DCE — «коммутационная фабрика без потерь» (Lossless Ethernet Switch Fabric) — является не менее важной, чем набор расширений Ethernet. Для того чтобы обеспечить реальную передачу трафика Ethernet без потерь, необходимо реализовать два обязательных требования: механизм приостановки передачи трафика по каналу в соответствии с классом трафика, такой как PFC, и метод приостановки трафика от входящего к исходящему порту через внутреннюю коммутационную фабрику.

Передача трафика без потерь внутри коммутационной фабрики достигается за счет объединения механизма PFC для приостановки трафика на входном порту коммутатора и механизма управления очередями на выходном порту коммутатора для предотвращения передачи пакетов внутри фабрики в случае недоступности выходного порта (механизм Virtual Output Queues, VOQ [9]). Таким образом, при соблюдении двух упомянутых выше требований реализуется полноценный сквозной Ethernet без потерь.

Технология Brocade Virtual Cluster Switching (VCS) [10] позволяет компаниям уменьшить сложность и снизить стоимость их сетевой инфраструктуры за счет использования новой архитектуры Ethernet-фабрики для ЦОД. VCS увеличивает масштабируемость и эффективность использования сети, кардинально упрощает архитектуру и увеличивает доступность приложений, что необходимо для современных центров обработки данных использующих виртуализацию. VCS включает набор динамических сервисов расширяющих функционал и обеспечивающих защиту инвестиций заказчиков, что делает эту технологию основой для построения сетей виртуальных ЦОД (см. рисунок 1).

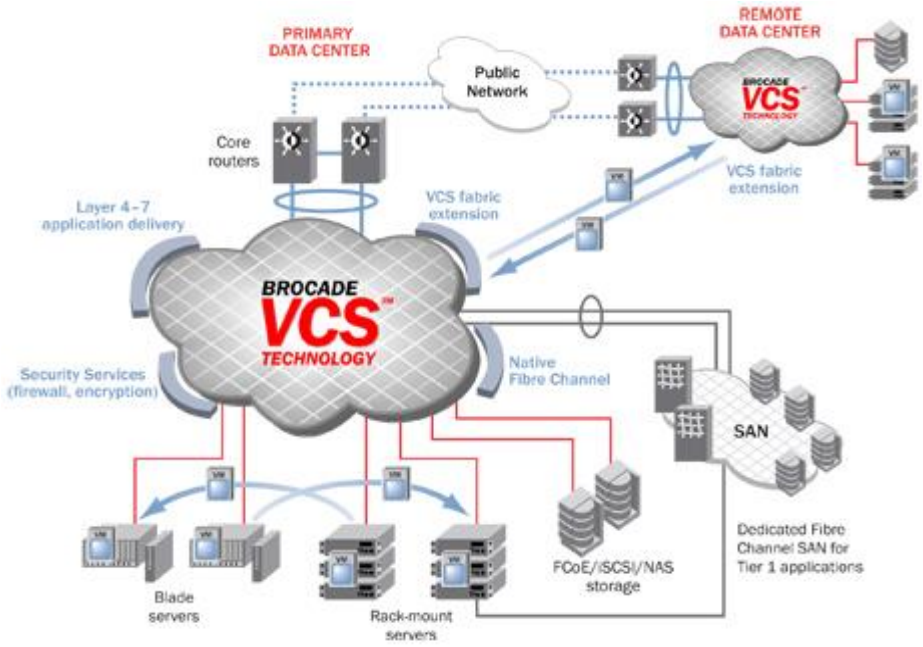

Рисунок 1 – Ethernet-фабрика VCS [10]

Ethernet-фабрика позволяет обойти такие ограничения традиционной многоуровневой архитектуры ЦОД как:

а) образование петель трафика при неправильной настройке протокола STP;

б) блокировку части портов и невозможность их использования для передачи трафика;

в) задержки и потери пакетов при изменениях топологии сети;

г) сложность в настройке большого количества устройств.

VCS позволяет отойти от идеологии настройки и функционирования коммутаторов как самостоятельных устройств, имеющих собственные уровни управления и передачи данных. Теперь коммутаторы работают в составе фабрики, имеющей единый интерфейс управления и передачи данных вне зависимости от количества подключенных к фабрике устройств, количества и типа соединений между ними. Новые коммутаторы, подключаемые к фабрике, присоединяются и настраиваются автоматически без вмешательства администратора. Снаружи фабрика выглядит как логически единый коммутатор с множеством внешних портов.

Одним из прогрессивных методов канальной коммутации в Интернет является технология многопротокольной коммутации на основе меток (Multiprotocol Label Switching — MPLS). MPLS - масштабируемый независимый механизмом передачи данных. В сети MPLS, пакетам данных присваиваются особые метки. Решение о передаче пакета данных другому узлу сети (коммутация) осуществляется только на основании метки без необходимости изучения пакета данных. Такая технология сделала возможным создание виртуального канала связи независимо от среды передачи и использующего любой протокол канального и сетевого уровней.

Классификация трафика по нескольким параметрам позволяет маршрутизировать потоки трафика каждого класса по специально оптимизированному пути администратором сети.

При точном планировании маршрутов и правил технология MPLS обеспечивает высокий уровень контроля над трафиком. Это является предпосылкой к более производительной работе сетей, позволяет гарантировать качество услуг, позволяющее адаптировать сеть к потребностям пользователей. Критерии, применимые в MPLS сетях для классификации пакетов различаются в зависимости от задач.

Компьютерные сети в ЦОД, как правило, строятся на базе протокола Ethernet. В основе работы коммутатора Ethernet лежит принцип программного управления на основе таблицы коммутации. Управление таблицами коммутации дает возможность произвольным образом управлять поведением и скоростными характеристиками и отдельного коммутатора, и параметрами передаваемых потоков данных в масштабах всей сети Ethernet.

При обработке пакетов данных коммутатор Ethernet обращается к таблице коммутации. На основании полученной информации, в том числе адреса порта получателя, параметров качества обслуживания, коммутационная матрица осуществляет дальнейшую обработку и передачу данных на целевой выходной порт (см. рисунок 2).

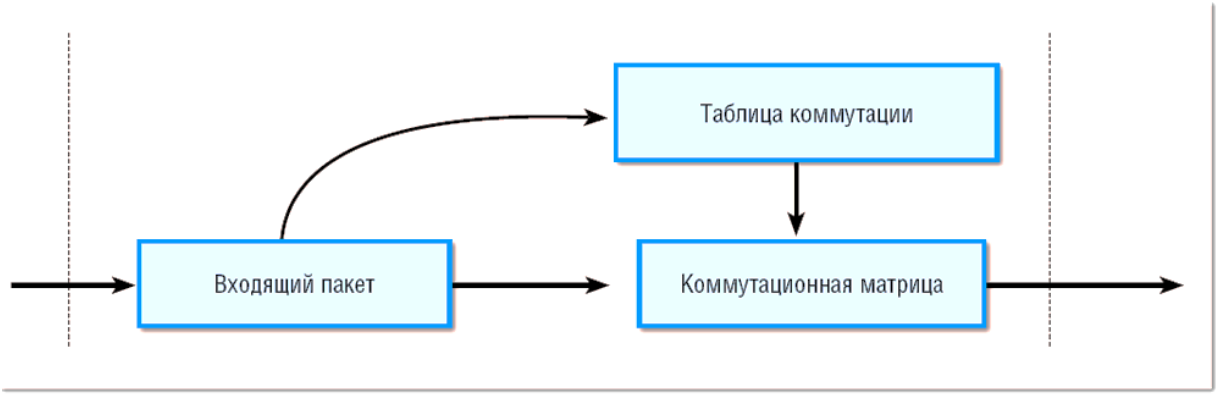

Рисунок 2 – Обработка и продвижение данных в коммутаторе [11]

Основная идея ПКС состоит в том, чтобы не изменяя существующего сетевого оборудования, отделить (перехватить) управление этим оборудованием (маршрутизаторами и коммутаторами) за счет создания специального программного обеспечения, которое может работать на обычном отдельном компьютере и которое находится под контролем администратора Сети.

Для реализации этой идеи специалистами из Стенфорда и Беркли был разработан открытый протокол OpenFlow [12] для управления маршрутизацией и коммутацией в Сети, не ориентированный на продукты какого-то отдельного производителя. С помощью этого протокола специалисты сами могут определять и контролировать: кто с кем, при каких условиях и с каким качеством может взаимодействовать в Сети. Все маршрутизаторы и коммутаторы объединяются под управлением Сетевой Операционной Системы (СОС), которая обеспечивает приложениям доступ к управлению сетью, и которая постоянно отслеживает конфигурацию средств Сети.

В коммутаторе OpenFlow реализован только уровень передачи данных. Вместо процессора обработки используется более простой контроллер, в задачи которого входит принятие поступающего кадра, извлечения из него MAC-адреса и немедленной передаче кадра коммутационной матрице если адрес есть в таблице. Если адрес отсутствует, коммутатор отправляет запрос на центральный контроллер сети OpenFlow и на основании ответа добавляет необходимые записи в таблицу коммутации. После этого коммутатор осуществляет обработку кадра.

Концепция гибридного коммутатора OpenFlow предполагает наличие в коммутаторе и традиционного уровня управления, и контроллера OpenFlow. Это позволяет реализовать функциональность OpenFlow в действующих сетях Ethernet и, в частности, осуществлять разработку нового программного обеспечения и протоколов, не мешая работе остальных пользователей сети.

1.2 Сетевые операционные системы в программно-конфигурируемых сетях

В основе ПКС лежит представление о компьютерной сети, как сети, имеющей «плоскость данных», которая отвечает за пересылку пакетов на основе состояния в каждом коммутаторе, и «плоскости управления», которая отвечает за вычисление, «планирование» и управление пересылкой. В сетях новой архитектуры все маршрутизаторы и коммутаторы объединяются под управлением единой сетевой операционной системы (СОС), которая обеспечивает приложениям доступ к управлению сетью (см. рисунок 3) и постоянно отслеживает конфигурацию средств сети.

Рисунок 3 – Схема ПКС

В отличие от традиционного толкования термина СОС как операционной системы, интегрированной со стеком сетевых протоколов, в данном случае под СОС понимается программная система, обеспечивающая мониторинг, доступ, управление, ресурсами всей сети, а не конкретного узла, и СОС (или контроллер) является основным, звеном ПКС-сети. Операционная система не управляет сетью, она предоставляет программный интерфейс (API) для управления. Таким образом, фактически решение задач управления сетью выносится на уровень приложений, реализованных на основе API сетевой операционной системы.

СОС формирует данные о состоянии всех ресурсов сети и обеспечивает доступ к ним для приложений управления сетью. Эти приложения управляют разными аспектами функционирования сети, включая построение топологии, принятие маршрутизирующих решений, балансировку нагрузки.

Для реализации этой идеи был разработан открытый протокол OpenFlow для управления сетевым оборудованием, не ориентированный на продукты какого-то отдельного поставщика. С помощью этого протокола специалисты сами могут определять и контролировать: какие узлы, при каких условиях и с каким качеством могут взаимодействовать в сети.

OpenFlow (англ. открытый поток) — открытый протокол и технология управления процессом обработки данных, которые передаются по компьютерной сети коммутаторами и маршрутизаторами. Схема работы протокола OpenFlow представлена на рисунке 4.

Рисунок 4 – Схема работы протокола OpenFlow

Протокол применяется для управления работой сетевых коммутаторов (маршрутизаторов) с центрального устройства – контроллера сети (например, с выделенного сервера или даже с обычного персонального компьютера). Это управление полностью заменяет или дополняет собой работающую на коммутаторе (маршрутизаторе) прошивку (специальная программа, встроенная производителем, которая осуществляет построение маршрута, и создает карты коммутации). Контроллер применяется для управления таблицами потоков коммутаторов (маршрутизаторов). На основании таблиц потоков принимается решение о передаче принятого пакета на конкретный порт коммутатора, или его игнорирование. В результате, в сети устанавливаются прямые сетевые соединения с минимальными задержками и требуемыми параметрами передачи данных.

Протокол OpenFlow решает также проблему зависимости от сетевого оборудования какого-либо конкретного поставщика, поскольку ПКС использует абстракции для пересылки пакетов, которые сетевая операционная система использует для управления сетевыми коммутаторами.

В настоящее время разработан целый ряд сетевых операционных систем, реализующих протокол OpenFlow.

а) NOX является платформой управления сетью, обеспечивающей высокоуровневый программируемый интерфейс, с помощью которого можно управлять сетью и/или создавать приложения для управления сетью. Так как NOX является контроллером OpenFlow, то управление сетью производится на уровне потоков и подразумевается, что контроллер определяет, как каждый поток маршрутизируется в сети [13].

В отличие от стандартных средств построения сети (таких как создание маршрутизаторов на основе Linux), NOX позволяет централизованно программировать модель для всей сети. NOX предназначен для поддержки как крупных корпоративных сетей с сотнями коммутаторов (с поддержкой многих тысяч узлов), так и небольших сетей из нескольких узлов.

Ядро NOX предоставляет приложениям абстрактное представление сетевых ресурсов, включая топологию сети и расположение всех обнаруженных узлов.

Основные цели NOX:

1) обеспечить платформу, которая позволяет разработчикам и исследователям вводить новшества в домашних или корпоративных сетях, используя реальные сетевые аппаратные средства. Разработчики на NOX могут управлять всеми подключениями в сети, включая передачу, маршрутизацию, контроль узлов и пользователей, которым разрешается связь. Кроме того, NOX может влиять на любой поток.

2) предоставить нужное сетевое программное обеспечение операторам. На текущий момент можно централизованно управлять всеми коммутаторами сети, разрешением на уровне пользователей и на уровне узлов с помощью механизма политик.

б) Maestro – СОС для организации приложений для управления сетью [14]. Maestro обеспечивает интерфейсы для реализации модульных приложений управления сетью, для доступа и изменения состояния сети, а также координации их взаимодействия.

Платформа Maestro предоставляет интерфейсы для:

1) внедрения новых пользовательских функций управления путем добавления построенных из модулей компонентов управления;

2) поддержания состояния сети от имени управляющих компонентов;

3) компоновки компонентов управления, определяя последовательность исполнения и общее состояние компонентов сети.

Кроме того, Maestro пытается использовать параллелизм на единственной машине, чтобы улучшить производительность системы. Основной особенностью сети является то, что OpenFlow контроллер отвечает за начальную установку каждого потока, связываясь с подчиненными коммутаторами. Таким образом, производительность контроллера может быть узким местом. В разработке Maestro пытается уменьшить усилия программистов по управлению распараллеливанием, выполняет большую часть утомительного и сложного управления распределением рабочей нагрузки и планирования рабочих потоков. Проект Maestro переносим и

масштабируем, поскольку разработан с использованием платформы Java и поддерживает многопоточность.

в) Beacon – быстрый кроссплатформенный модульный, основанный на Java, контроллер OpenFlow, который поддерживает и работу на основе событий и на основе потоков [15].

Главные особенности:

1) устойчивость – Beacon был в разработке в течение полутора лет и использовался в нескольких научно-исследовательских проектах, сетевых классах и испытательном стенде. Beacon в настоящее время функционирует на 100 виртуальных и 20 физических коммутаторах в экспериментальном ЦОД и работал в течение многих месяцев без сбоев;

2) кроссплатформенность – Beacon написан на Java и работает на многих платформах от высококачественных многоядерных серверов Linux до телефонов на Android;

3) динамичность – пакеты Beacon могут быть запущены / остановлены / обновлены / установлены во время выполнения, не прерывая другие независимые пакеты;

г) Trema – платформа контроллера OpenFlow, которая включает все необходимое для создания контроллеров OpenFlow на Ruby и/или C [16].

Платформа включает весь исходный код Trema, необходимый для самостоятельной разработки контроллеров OpenFlow. Дерево исходных кодов включает все основные библиотеки и функциональные модули, которые работают интерфейсами коммутаторов OpenFlow. Также включено несколько примеров контроллеров OpenFlow, разработанных с помощью Trema, а для отладки имеется плагин для wireshark.

д) SNAC – контроллер OpenFlow, который использует веб-интерфейс менеджера сетевой политики, чтобы управлять сетью [17]. Он включает гибкий язык определения политики и удобный для пользователя интерфейс для конфигурирования устройств и отслеживания событий. Он построен как модуль NOX, однако требует определенную версию базового контроллера NOX.

е) BigSwitch – контроллер OpenFlow с закрытым исходным кодом, основанный на контроллере Beacon [18]. Контроллер BigSwitch использует удобную в использовании консоль централизованного управления сетью, что позволяет использовать контроллер в условиях развитой инфраструктуры крупного предприятия;

ж) Floodlight [19] является контроллером OpenFlow с открытым исходным кодом. Floodlight Open SDN Controller относится к контроллерам корпоративного класса, под лицензией Apache, на основе Java OpenFlow контроллера. Проект поддерживается сообществом разработчика, в том числе рядом инженеров из Big Switch Network.

Floodlight предназначен для работы с растущим числом коммутаторов, маршрутизаторов, виртуальных коммутаторов и точек доступа, поддерживающих протокол OpenFlow.

Основные функциональные возможности Floodlight:

а) наличие системы загрузки модулей, которая позволяют модифицировать и расширить функционал контроллера;

б) простая установка с минимальными зависимостями;

в) поддержка широкого спектра виртуальных и физических OpenFlow коммутаторов;

г) возможность обрабатывать смешанные сети, OpenFlow и традиционные.

1.3 Описание таблиц протокола OpenFlow

OpenFlow-совместимые коммутаторы делятся на два типа: OpenFlow-only (только OpenFlow) и OpenFlow-hybrid (OpenFlow-гибрид). Коммутаторы типа OpenFlow-only поддерживают только операции OpenFlow, в этих коммутаторах все пакеты обрабатываются обработчиком OpenFlow и не могут быть обработаны чем-либо еще.

Коммутаторы типа OpenFlow-hybrid поддерживают все операции OpenFlow и обычные операции коммутации Ethernet, т.е. традиционную коммутацию второго уровня, VLAN, маршрутизацию третьего уровня, ACL и QoS.

Эти коммутаторы должны обеспечивать механизм разделения вне OpenFlow, который направляет трафик либо через обработчика пакетов OpenFlow, либо через обычный обработчик пакетов. Например, коммутатор может использовать теги VLAN или входящий порт пакета, чтобы определить, следует ли обрабатывать пакет с помощью одного обработчика или другого, или он может направить все пакеты обработчику OpenFlow.

OpenFlow-обработчик каждого OpenFlow-коммутатора содержит несколько таблиц потоков, а каждая таблица потоков содержит несколько записей. OpenFlow-обработчик определяет, как пакет взаимодействует с этими таблицами потоков (см. рисунок 5).

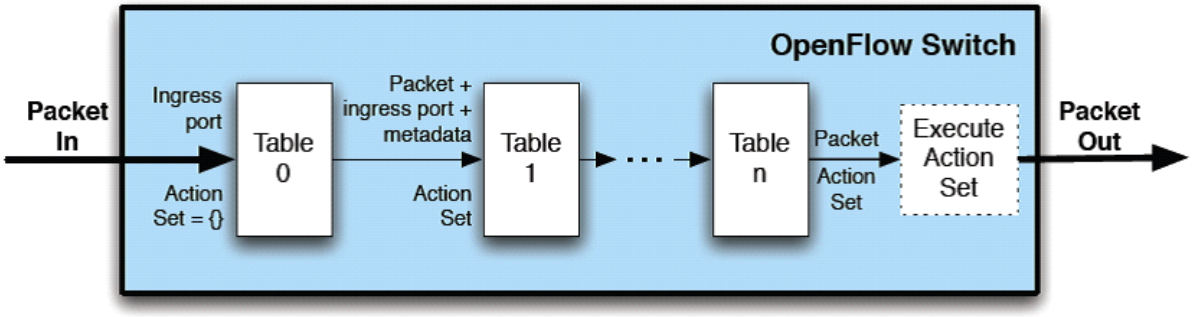

Рисунок 5 – Прохождение пакета через конвейер обработки [20]

OpenFlow коммутатор может иметь только одну таблицу потоков, в этом случае конвейер обработки значительно упрощается. Таблицы потоков в OpenFlow-коммутаторах последовательно нумеруются, начиная с нуля. Конвейерная обработка всегда начинается с первой таблицы потоков, пакет сначала сравнивается с записями таблицы потоков с номером ноль. Другие таблицы потоков могут быть использованы в зависимости от результата сравнения с первой таблицей.

Если пакет совпадает с записью в таблице потоков, то выполняется соответствующий этой записи набор инструкций. Инструкции в записи потока

могут явно перенаправлять пакет в другую таблицу потоков, где тот же процесс повторяется снова. Записи потока могут перенаправлять пакет только в ту таблицу, номер которой больше собственного номера таблицы, другими словами, конвейерный процесс может идти только вперед, а не назад. Очевидно, что записи последней таблицы конвейера не могут содержать инструкции перехода на другую запись (Goto). Если соответствующая запись потока не направляет пакеты в другую таблицу потоков, конвейер обработки останавливается в этой таблице. Когда конвейерная обработка прекращается, пакет обработан соответствующим набором действий и обычно отправлен.

Если пакет не соответствует ни одной записи в таблице потоков, то эта таблица пропускается. Поведение в случае пропуска таблицы зависит от настроек этой таблицы, по умолчанию – это отправка пакетов контроллеру через канал управления, другой вариант – удаление пакета. Также возможен случай, что при пропуске таблицы обработку пакетов следует продолжать, в этом случае пакет обработается следующей по порядку таблицей.

В таблице 1 показаны поля для сравнения входящих пакетов. Каждая запись может содержать либо конкретное значение, либо значение ANY (любой пакет).

Таблица 1 – Поля пакетов для участия в процессе коммутации

№	Название поля	Расшифровка
1	2	3
1	IngressPort	Входящий интерфейс пакета
2	Metadata	Метаданные
3	Ethersrc	Ethernet адрес источника
4	Etherdst	Ethernet адрес получателя
5	Ethertype	Версия протокола Ethernet
6	VLAN id	Идентификатор сети VLAN

Продолжение таблицы 1

1	2	3
7	VLAN priority	Приоритет сети VLAN
8	MPLS label	Метка протокола MPLS
9	MPLS traficclass	Класс трафика протокола MPLS
10	IPv4 src	IP адрес источника
11	IPv4 dst	IP адрес получателя
12	IPv4 ToSbits	Приоритет пакета
14	TCP/ UDP / SCTP src port	Номер порта протокола TCP/UDP/SCTP источника пакета
15	ICMP Type	Тип протокола ICMP
16	TCP/ UDP / SCTP dst port	Номер порта протокола TCP/UDP/SCTP получателя пакета
17	ICMP Code	Код протокола ICMP

Если коммутатор поддерживает технологию битовых масок в полях «Ethersrc», «Etherdst», «IPv4 src», «IPv4 dst», такие маски позволяют более точно указать совпадения. В дополнение к заголовку пакета, совпадения могут также быть выполнены в отношении входящего интерфейса и поля метаданных. Метаданные могут быть использованы для передачи информации между таблицами в коммутаторе.

1.4 Обеспечение качества обслуживания в программно-конфигурируемых сетях

Обеспечение качества обслуживания (Quality of Service, QoS) в существующих гибридных сетях основывается на контроле входа и выхода пакета из устройства. При этом могут использоваться как потоки, так и пакеты данных, реализация QoS сводится к установке приоритетов конкретных пакетов. Контроль над прохождением пакетов через сеть доступен только в пределах центра обработки

данных (ЦОД), за пределами ЦОД вся ответственность ложится на провайдеров телекоммуникационных услуг.

Кроме того, реализация QoS позволяет решить проблемы, связанные с работой сети, такие как перегрузка каналов и потери пакетов, которые в конечном итоге приводят к снижению производительности приложений. Перегрузка каналов и потери пакетов могут произойти по ряду причин, например превышение максимальной пропускной способности, высокий коэффициент загрузки (это делает устройства неспособными к эффективной обработке входящих пакетов), а также при неправильной конфигурации сетевых устройств.

Некоторые типы QoS – например, ограничение полосы пропускания, могут помочь в решении сетевых проблем. Управление полосой пропускания может уменьшить проблемы, связанные с ограниченной пропускной способностью внешних каналов связи между ЦОД и Интернет. Если пропускная способность ограничена на другом конце канала, этот метод не улучшит производительность, так как контроллер OpenFlow не имеет сведений об этих ограничениях. OpenFlow контроллер будет применять правила вслепую по отношению к трафику за пределами центра обработки данных.

Вне контекста приложений QoS редко дает заметное улучшение в производительности с точки зрения конечного пользователя. Контекст является необходимым для того, чтобы применять правильные методы и политику в нужное время, чтобы обеспечить оптимальную производительность для пользователей конкретного приложения.

Большинство коммутаторов и маршрутизаторов сегодня поставляются со встроенной поддержкой QoS. Некоторые продукты поддерживают только тип DiffServ (RFC-2474 и RFC-2475, небольшое количество очередей и политик), другие поддерживают тысячи очередей и большое количество политик. На практике, при реализации QoS в сетях с коммутацией потоков, поток отображается на класс трафика QoS, то есть поток получает идентификатор уже имеющегося класса.

В настоящее время существует два основных типа QoS услуг: ограничение скорости на входе и гарантии минимальной пропускной способности на выходе.

Первая услуга, как правило, реализуется с помощью измерения скорости входного порта или потока, после превышения определенной скорости пакеты отбрасываются в соответствии с некоторым алгоритмом. Ограничение минимальной гарантированной полосы пропускания означает, что каждому потоку будет доступна определенная часть от имеющейся пропускной способности.

Спецификация OpenFlow 1.1.0 предусматривает поддержку ограниченного функционала в области QoS. На каждый порт можно подключить одну или несколько очередей. Каждый поток ассоциируется с конкретной очередью. Обработка в очереди происходит в соответствии с настройкой конкретной очереди. По умолчанию все очереди имеют настройку на предоставление минимальной гарантированной пропускной способности, другие типы очередей не поддерживаются текущей версией OpenFlow.

Для обеспечения QoS в свойствах конкретного потока существует поле IP ToS, в нем содержится 6-битный DSCP идентификатор, Input VLAN Priority для приоритезации одного VLAN над другим. Для обработки потока существуют следующие действия:

а) OFPAT_SET_VLAN_PCP устанавливает приоритет VLAN 802.1q;

б) OFPAT_SET_NW_TOS устанавливает новое значение DSCP потоку;

в) OFPAT_SET_QUEUE устанавливает новый идентификатор очереди для данного потока, другими словами ассоциирует конкретный поток с очередью вне зависимости от DSCP и VLAN Priority.

Для настройки самих очередей QoS, необходимы особые команды, не относящиеся к OpenFlow. Очереди настраиваются средствами операционной системы коммутатора. OpenFlow-контроллер может только запросить информацию об очередях на конкретном порту: количество очередей, их тип, количество пакетов в очереди. Также можно запросить статистику по всем очередям коммутатора: переданные пакеты, переданные байты, количество переполнений очереди. Каждая очередь имеет определенный размер в байтах и тип – простая FIFO очередь или с ограничением по гарантированной полосе пропускания. Для второго типа очереди настраивается гарантированная полоса пропускания в процентах (одна единица

измерения равна 0.1 процента) от максимально доступной полосы пропускания порта коммутатора.

Таким образом, существует два действия для обеспечения QoS в OpenFlow – настройка очередей на коммутаторах с привязкой к портам или без нее и связывание конкретных потоков

Существует альтернативный метод обеспечения QoS – передача обязанностей по обработке пакетов конкретного потока операционной системе коммутатора. Это возможно только на коммутаторах, поддерживающих как традиционную обработку пакетов, так и обработку с помощью OpenFlow (см. рисунок 6).

Рисунок 6 – Схема обеспечения QoS в OpenFlow коммутаторе

Однако, базовая концепция QoS предполагает, что все пакеты всегда идут по одному направлению в течение довольно длительного времени, топология сети статична, путь между двумя точками детерминирован. Сети OpenFlow не позволяют применять сложные механизмы QoS именно из-за недетерминированности топологии и путей следования пакетов. К тому же, как правило, при глубокой интеграции виртуализированных ресурсов в ЦОД требования к обеспечению требуемых показателей работы сети меняются в процессе работы кардинально для разных виртуальных сетей.

В связи с описанными ограничениями реализации QoS на практике лучше использовать первый подход к реализации – настройка очередей на коммутаторах и сопоставление битов QoS и конкретных потоков на контроллере. Затем всю обработку QoS наиболее рационально перенести на операционную систему коммутатора в соответствии с битами QoS, что позволит максимально использовать все преимущества дополнительных возможностей очередей коммутаторов.

2 Лабораторная работа №1 – Создание программно-конфигурируемой сети

2.1 Теоретические сведения

2.1.1 Работа с эмулятором mininet

Виртуальное окружение mininet является симулятором топологии на виртуальных хостах. Это виртуальная машина для системы Oracle VirtualBox, которая заранее готова к работе. Исходный материал с полной документацией можно найти на сайте Openflow.org, а полный учебник по работе с mininet и образ для виртуальной машины – по адресу: http://www.openflow.org/wk/index.php/OpenFlow_Tutorial.

Mininet — это образ Ubuntu с уже установленными контроллером, пакетным перехватчиком и эмулятором mininet. Установим образ на Virtualbox.

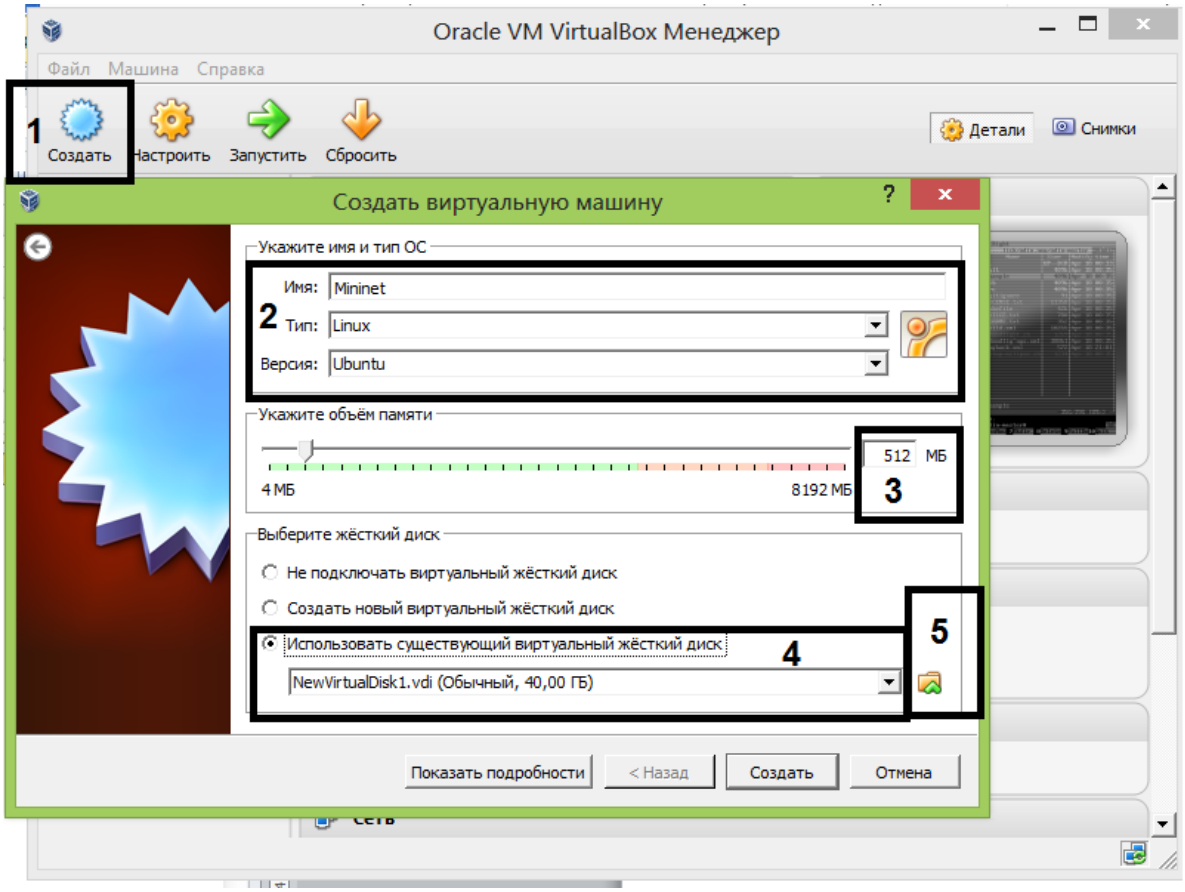

Рисунок 7 – Создание виртуальной машины

После запуска VirtualBox необходимо нажать кнопку создать (поле 1 на рисунке 7), написать название виртуальной машины и выбрать тип Linux-Ubuntu, как показано в поле 2 рисунка 7. Объем памяти установить не менее 512Мб (поле 3), для жетского диска установить значение «Использовать существующий жесткий диск» (поле 4), нажать кнопку выбора файла (поле 5). Необходимо выбрать скачаный образ Mininet и нажать кнопки «Открыть», потом «Создать».

После создания необходимо настроить на виртуальной машине интерфейсы, через которые можно открывать ssh сессии.

Для этого в VirtualBox выделяем виртуальную машину с mininet, нажимаем кнопку «Настроить» (рисунок 8), далее на вкладке «Сеть» (поле 1 рисунка 8) выбираем вкладку «Адаптер 2» (поле 2) и ставим галку напротив «Включить сетевой адаптер» (поле 3). Затем из выпадающего списка (поле 4) выбираем «Внутренняя сеть».

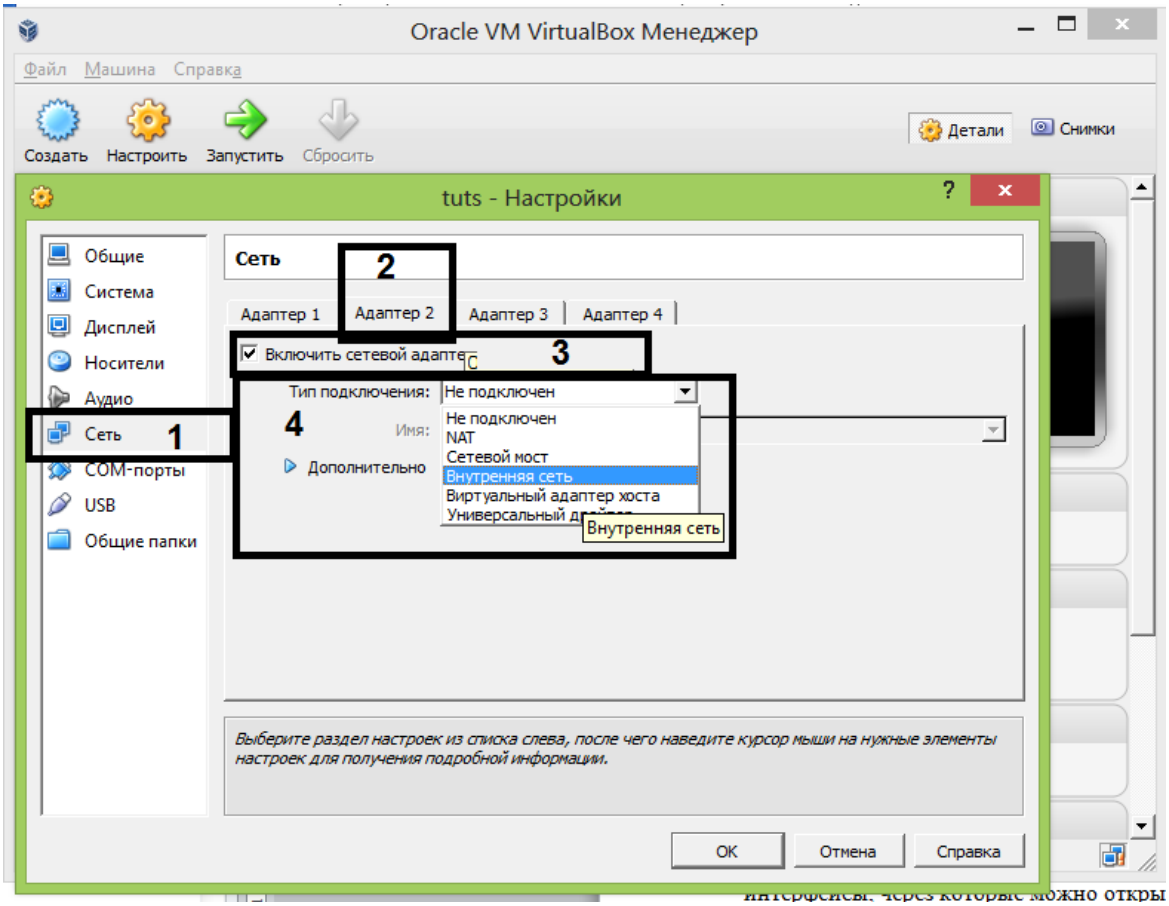

Рисунок 8 – Настройка адаптера 2 виртуальной машины

После этого, для возможности доступа к Интернет и внутренней сети необходимо настроить адаптер 1 (рисунок 9). В вкладке сверху (поле 1) выбрать «Адаптер 1», затем тип адаптера – «Сетевой мост» (поле 2), в выпадающем списке физических адаптеров (поле 3) выбрать нужный (Ethernet или WiFi адаптер). После настройки необходимо применить параметры нажатием кнопки «Ok».

Теперь необходимо запустить виртуальную машину двойным щелчком мыши на названии виртуальной машины. После запуска (рисунок 10) будет доступна консоль Linux.

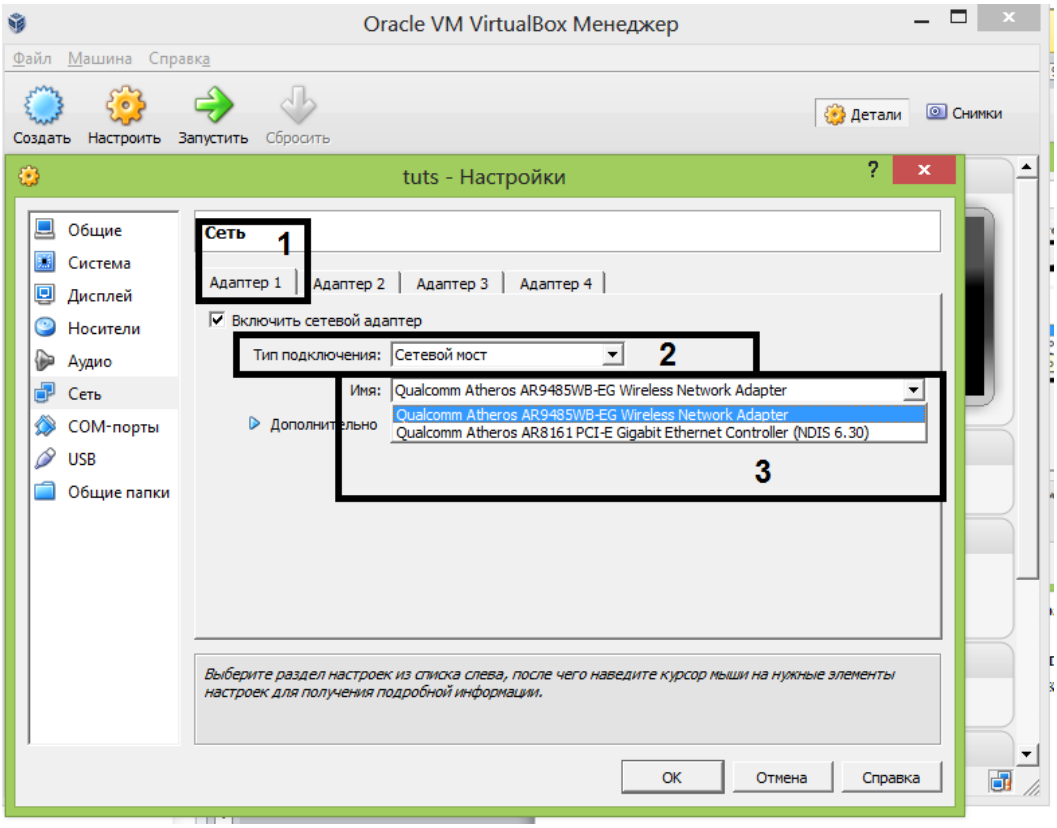

Рисунок 9 – Настройка адаптера 2 виртуальной машины

Необходимо ввести логин и пароль по умолчанию, для 32-битного образа пользователь openflow пароль openflow, для 64-битного образа это соответственно (mininet, mininet). Обратите внимание, что при вводе пароля символы не отображаются.

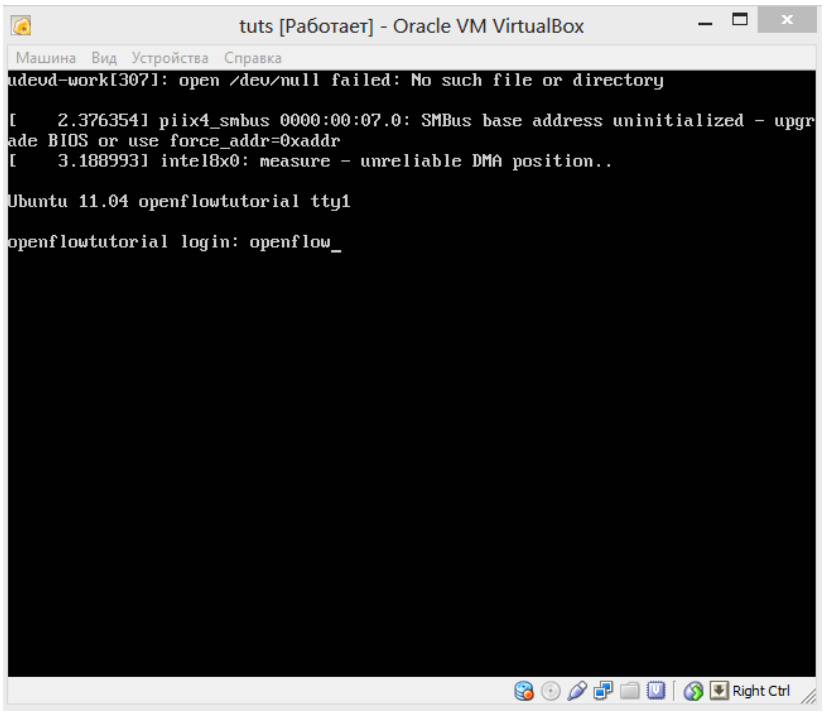

Рисунок 10 – Внешний вид консоли виртуальной машины

Для проверки сетевых интерфейсов необходимо использовать команду «ifconfig -a» (рисунок 11).

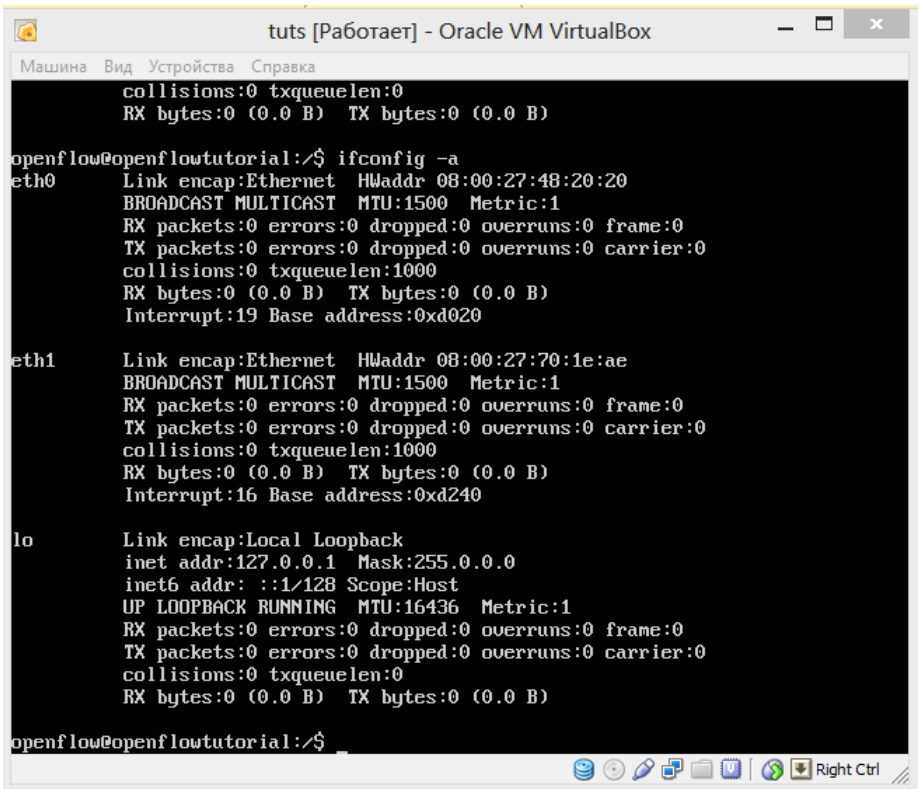

Рисунок 11 – Проверка сетевых параметров виртуальной машины

Если адресов нет, как на рисунке 11, необходимо выполнить команду на получение адресов через DHCP «sudo dhclient eth0», «sudo dhclient eth1» в соответствии с именами интерфейсов. После выполнения необходимо проверить факт получения адресов (рисунок 12 поле 1).

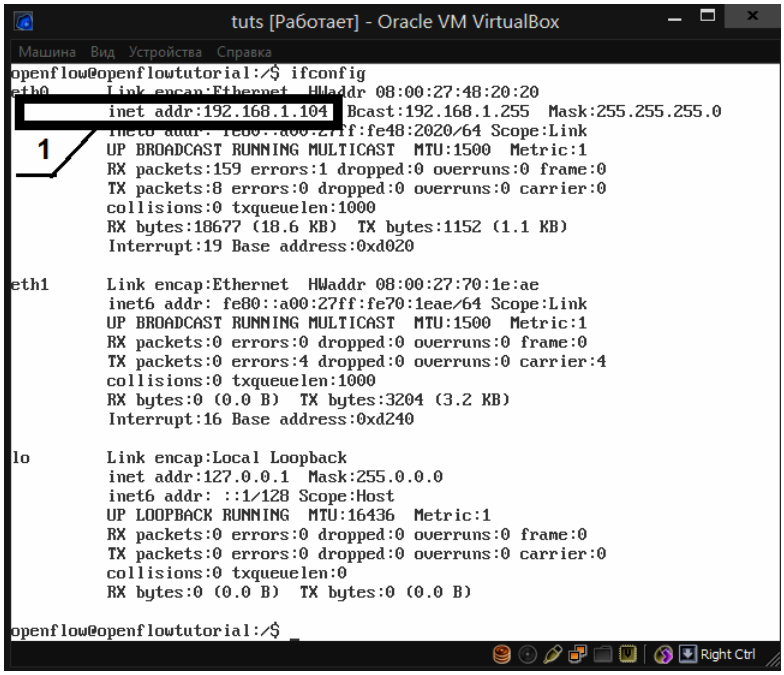

Рисунок 12 – Местонахождение адреса в выводе команды ifconfig

После этого можно подключиться любым SSH клиентом, например PUTTY (рисунок 13).

Адрес в поле 1 должен соответствовать адресу в выводе команды «ifconfig», как на рисунке 6 поле 1. Тип протокола – SSH (поле 2). После нажатия кнопки «Open» откроется окно (рисунок 14) с предложением ввести логин и пароль.

Для полноценного использования необходимо установить графическую среду. Это происходит только при подключении к Интернет. Если в сети не используется прокси-сервер, то можно просто приступать к выполнению команд установки, иначе необходимо прописать прокси-сервер, перед этим войти в режим root командой «sudo su», ввести пароль и выполнить команды:

«export http_proxy=«http://адрес_сервера:порт_сервера» и «export ftp_proxy=«http://адрес_сервера:порт_сервера».

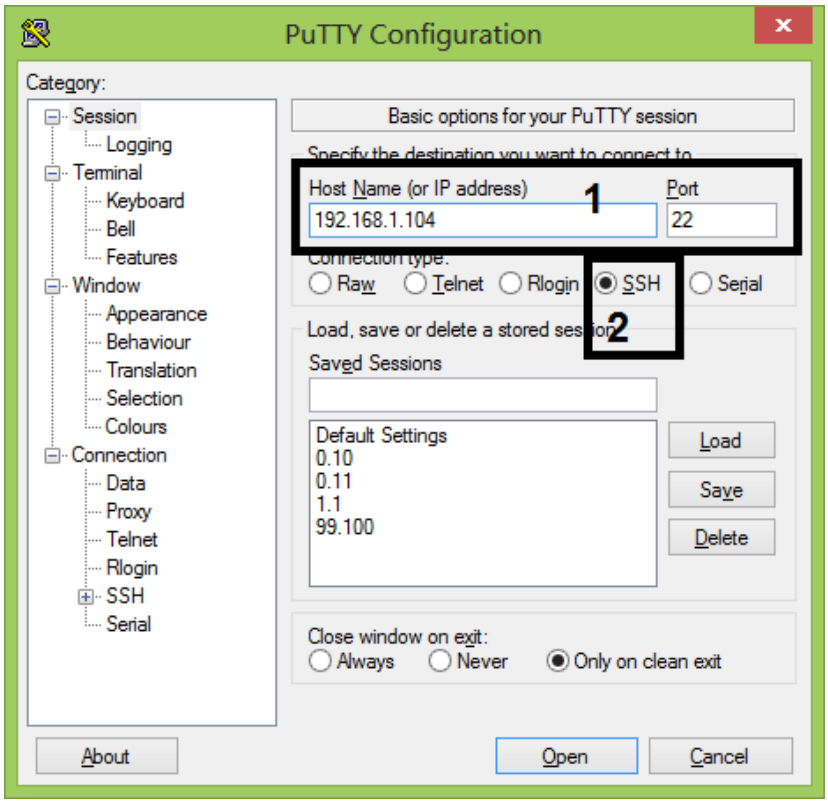

Рисунок 13 – Настройка SSH клиента

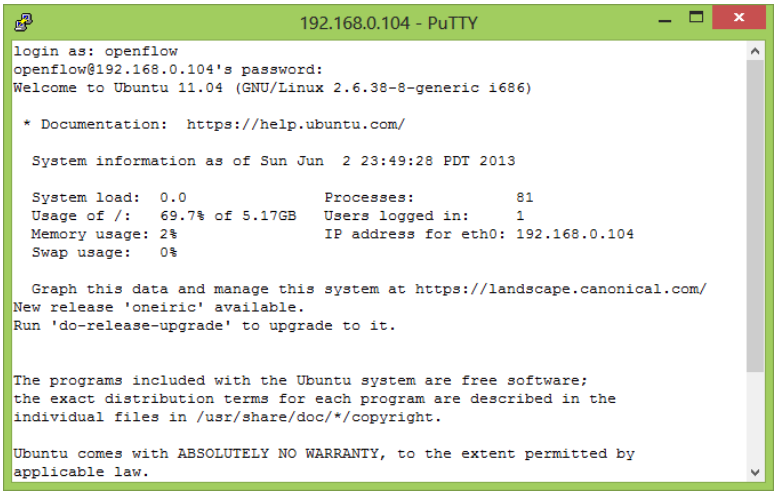

Рисунок 14 – Подключение по SSH

Если прокси сервер требует авторизацию, то команда будет выглядеть как «export http_proxy=«http://логин:пароль@адрес_сервера:порт_сервера» и «export ftp_proxy=«http://логин:пароль@адрес_сервера:порт_сервера».

После настройки можно запустить установку графического интерфейса командой «sudo apt-get update» и «sudo apt-get install xinit flwm»

Графический интерфейс можно запустить только из самой виртуальной машины, но не из SSH клиента.

Для работы с mininet можно использовать следующие ключи запуска (см. таблицу 2).

Таблица 2 – Ключи запуска mininet

Команда	Действие
1	2
-h, --help	Вывод справки
--switch=	Выбор типа программно-управляемого коммутатора. Варианты: [kernel user ovsk]
--host=	Задает адрес запуска mininet
--controller=	Задает тип контрооллера. Варианты: [nox_dump none ref remote nox_pysw]
--topo=	Задает параметры топологии. Варианты: [tree reversed single linear minimal] через запятую могут указываться аргументы ,arg1,arg2,...argN
-c, --clean	Закрывает mininet и очищает все виртуальные машины
--mac	Включает формирование разных MAC адресов для хостов
-x, --xterms	Запустить консоли в графическом режиме для каждого хоста
--arp	Заранее установить arp записи в хосты в статичном режиме
--ip=	Адрес контроллера
--port=	Порт контроллера
--prefixlen=	Длина маски для автоматического распределения адресов

Например, для создания топологии, показанной на рисунке 15, необходимо выполнить следующую команду:

sudo mn --topo single,3 --mac --switch ovsk --controller remote

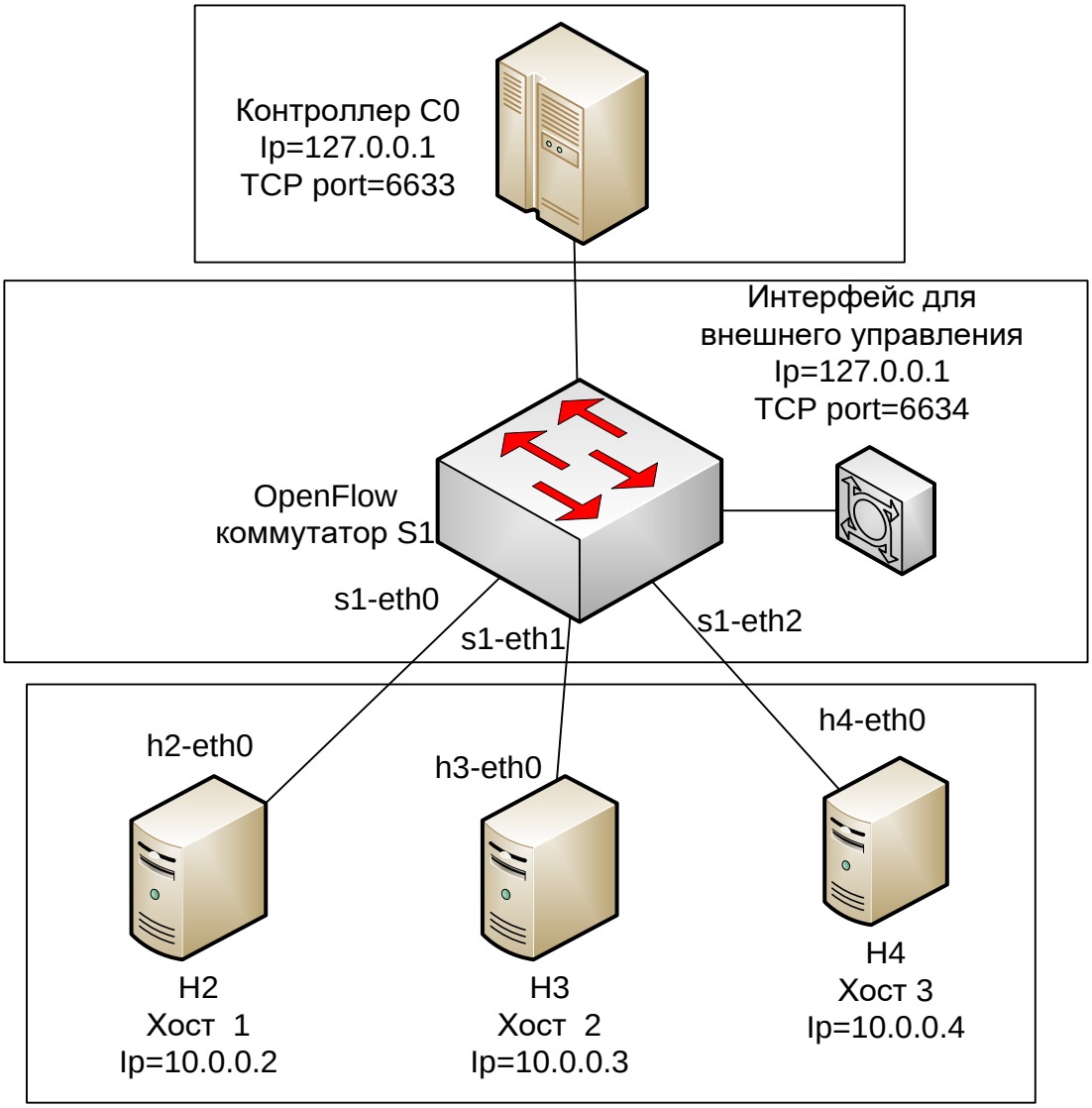

Рисунок 15 – Топология с 3 узлами и 1 коммутатором

При входе в mininet отображается консоль управления, начинающаяся со строки «mininet>»

Список команд представлен в таблице 3.

Таблица 3 – Список команд управления mininet

Команда	Описание
1	2
dump	Отображает все IP адреса устройств, их интерфейсы
intfs	Отображает список интерфейсов устройств
iperf	Измеряет производительность обмена по TCP между первым и последним хостом
iperfudp	Измеряет производительность обмена по UDP между первым и последним хостом
net	Показывает схему соединений устройств
link	Позволяет создать новую связь. Синтаксис: «link устройство1 устройство2 [up|down]»
nodes	Показывает список устройств
noecho	Выполняет команду, но не отображает информацию о результате
pingpair	Посылает ICMP пакеты между первыми двумя хостами
pingall	Посылает несколько ICMP пакетов между всеми хостами по очереди и показывает результат
py	Выполняет любую команду на языке Python
quit, exit	Закрывает mininet
source	Читает список команд из указанного после команды файла
xterm, gterm	Открывает в графическом режиме отдельное окно с консолью соответствующего хоста. Синтаксис команды: «xterm узел1 узел2 …». Команды работают только в графическом режиме виртуальной машине (см. рисунок 17)
sh	Выполняет произвольную команду в операционной системе
dpctl	Выполняет внешнее управление коммутаторами, синтаксис будет рассмотрен ниже.

Продолжение таблицы 3

ifconfig	Настраивает сетевые параметры указанного перед командой хоста. Синтаксис команды: «хост1 ifconfig параметры»
ping	Выполняет команду ping на указанном хосте. Синтаксис команды «хост1 ping хост2»

Для получения помощи по командам можно воспользоваться знаком «?» (рисунок 16), для подробной помощи можно набрать «help команда».

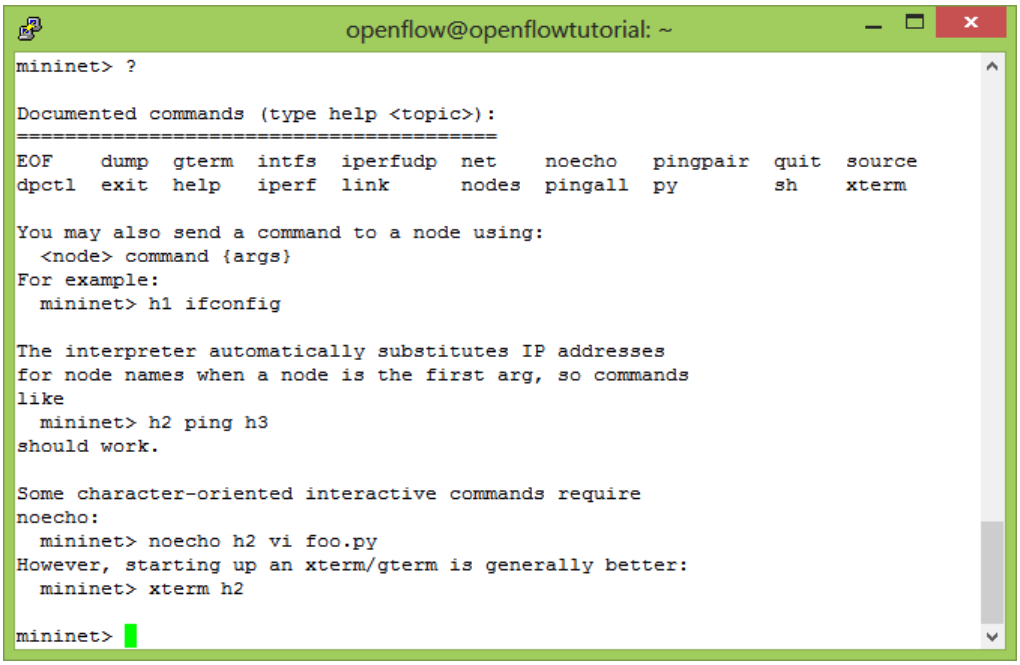

Рисунок 16 – Список и синтаксис команд mininet

При необходимости использовать графический интерфейс (для команд xterm, gterm) необходимо запустить его командой «startx» (рисунок 17), после запуска нажать правой кнопкой мыши на пустом месте рабочего стола и выбрать «Open Terminal Here» для запуска консоли. В консоли графического режима можно работать так же, как и в текстовом режиме.

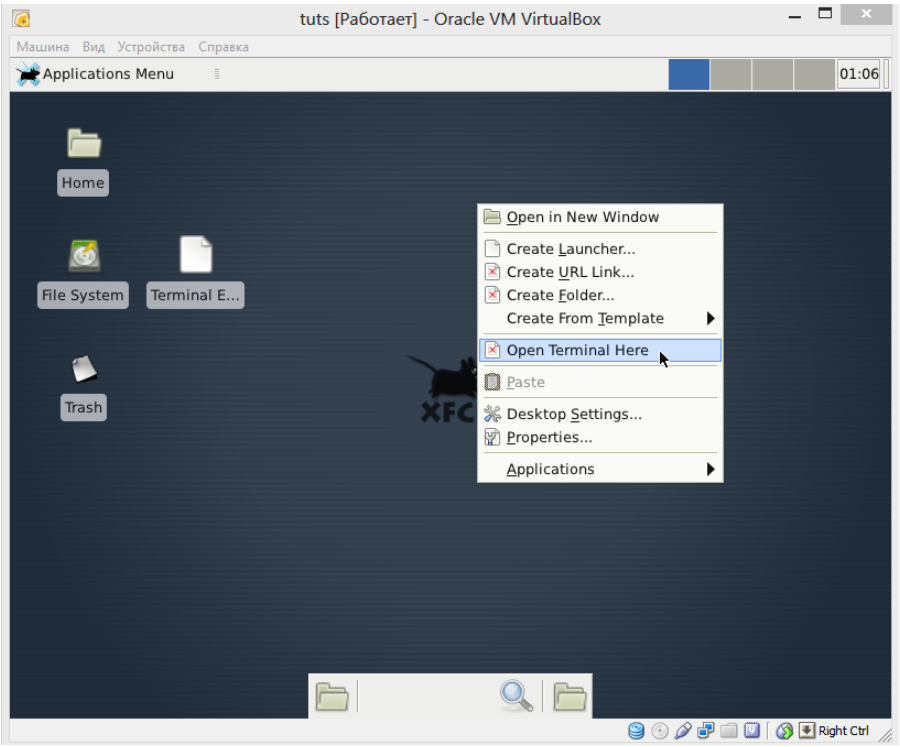

Рисунок 17 – Графический режим

Для запуска консоли виртуального хоста необходимо использовать команду xterm. Она запускает отдельное окно с консолью, в которой находится полноценный Linux на виртуальной машине (рисунок 18).

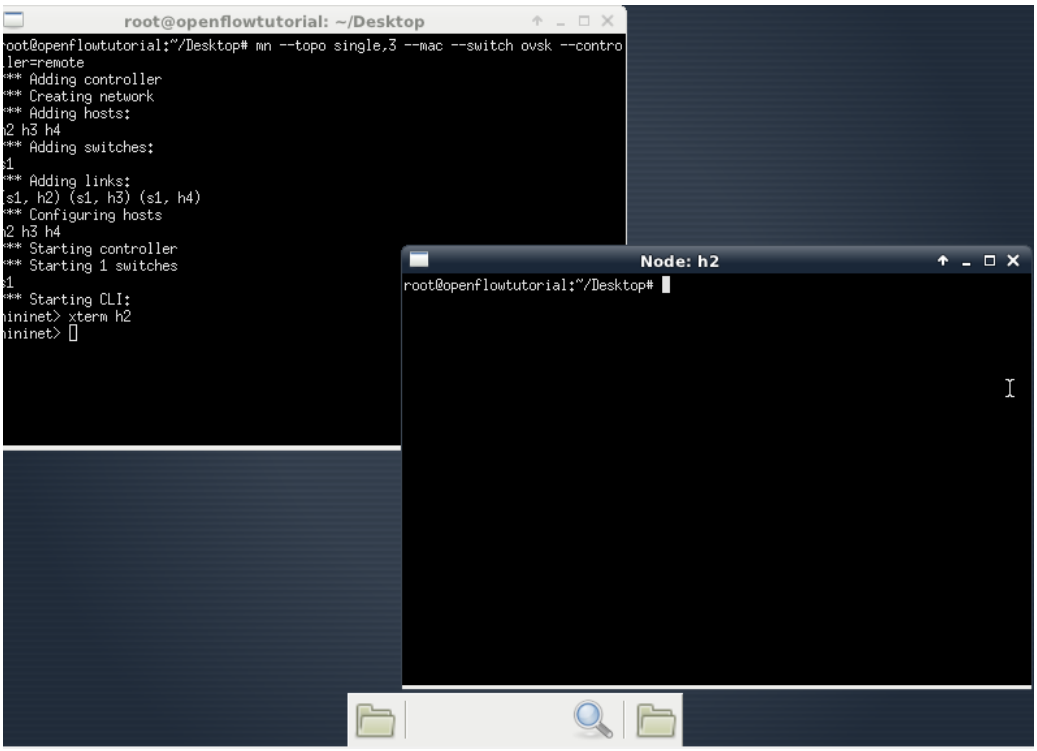

Рисунок 18 – Консоль xterm виртуального хоста

Весь набор команд Linux работает в этой консоли, включая ifconfig, ping, iperf, vi, sh и т.д.

По умолчанию, если не используется внешний контроллер (нет параметра «--controller=remote»), коммутатором можно управлять через команду «dpctl», рассмотренной в следующем разделе.

2.1.2 Внешнее управление таблицами OpenFlow коммутаторов

Программа «dpctl» используется для внесения изменений, просмотра и мониторинга таблиц OpenFlow. Основные команды показаны в таблице 4.

Таблица 4 – Команды dpctl

Команда	Описание
1	2
show	Показывает информацию о коммутаторе. Синтаксис команды «dpctl show tcp:адрес:порт», где «адрес» – это адрес коммутатора (127.0.0.1 в mininet), а «порт» - управляющий TCP порт коммутатора (6634 в mininet)
status	Показывает статистику по таблицам, портам и коммутатору в общем. Синтаксис аналогичен команде «show», но после команды можно использовать название статистики для фильтрации вывода
dump-desc	Показывает суммарную информацию о аппаратном обеспечении и прошивке коммутатора. Синтаксис аналогичен команде «show»
dump-flows	Показывает все записи в таблицах потоков на коммутаторе. Синтаксис аналогичен команде «show», после команды можно ставить условия фильтрации по номеру потока, например «(X=11,12,13)»

Продолжение таблицы 4

1	2
dump-ports	Показывает все интерфейсы на коммутаторе. Синтаксис аналогичен команде «show»
dump-queue	Показывает все программные очереди пакетов на коммутаторе. Синтаксис аналогичен команде «show»
dump-tables	Показывает статистику по таблицам OpenFlow. Синтаксис аналогичен команде «show»
add-flow	Добавляет запись в таблицу потоков. Синтаксис команды: «dpctl add-flow tcp:адрес:порт in_port=порт1,actions=output:порт2», где in_port=порт1 – это условие применения записи к потоку, поле output:порт2 – это реакция записи на поток, в данном случае перенаправить поток на другой порт коммутатора (порт2) если поток пришел из указанного интерфейса (порт1)
add-flows	Добавляет запись в таблицу потоков. Синтаксис команды: «dpctl add-flow tcp:адрес:порт файл», где «файл» – это файл с записями потоков
add-queue	Добавляет программную очередь коммутатора. Синтаксис команды: «dpctl add-queue tcp:адрес:порт интерфейс номер_очереди скорость», где «интерфейс» – это порт коммутатора, «номер_очереди» – уникальный идентификатор очереди, «скорость» – минимальная гарантированная пропускная способность в Кб/с

Продолжение таблицы 4

1	2
mod-flows	Изменяет реакцию на совпадение с полем сравнения потока. Синтаксис команды «dpctl mod-flows tcp:адрес:порт список_условий новое_действие», где «список условий» – полное перечисление поле сравнения потока как было указано при создании записи командой «add-flow», «новое_действие» – любое стандартное действие, которое будет заменено в таблице потоков. Если список условий указан в общем, то будут изменены все потоки, для которых это поле будет хотя бы частично совпадать с полем сравнения
mod-queue	Позволяет изменять минимальную гарантированную пропускную способность конкретной очереди. Синтаксис команды «dpctl mod-queue tcp:адрес:порт интерфейс номер_очереди скорость», где параметры аналогичны команде «add-queue»
mod-port	Синтаксис команды «dpctl mod-ports tcp:адрес:порт номер_порта команда», где номер порта – число, команда – одно из значений «up», «down», «flood», «noflood».
del-flows	Удаляет поток или группу потоков. Синтаксис команды «dpctl del-flows tcp:адрес:порт маска», где «маска» – условие или часть условия поля сравнения потока, при совпадении с которой поток удаляется
del-queue	Удаляет очередь. Синтаксис команды «dpctl del-queue tcp:адрес:порт интерфейс номер_очереди», где параметры аналогичны команде «add-queue»
monitor	Выводит на экран пакеты, пришедшие на коммутатор

Продолжение таблицы 4

1	2
execute	Выполняет произвольную команду в ОС коммутатора. Синтаксис «dpctl execute tcp:адрес:порт команда», где «команда» – произвольная строка в кавычках
ping	Показывает время отклика коммутатора. Синтаксис аналогичен команде «show»
benchmark	Измеряет производительность коммутатора или контроллера. Синтаксис «dpctl benchmark tcp:адрес:порт размер_пакета количество», где «размер_пакета» – размер пакета, посылаемого для проверки производительности, а «количество» – это количество таких пакетов. Результат возвращается в виде суммарного времени выполнения посылки всех команд и пакетов, а также в вид средней производительности в пакетах/с

Список полей для создания списка условий применения конкретной записи о потоке к пакету при добавлении и изменении записей (действия «add-flow», «add-flows», «mod-flows», «del-flow») приведен в таблице 5.

Таблица 5 – Поля для фильтрации записей OpenFlow

Поле проверки	Описание
1	2
in_port	Входящий порт коммутатора
Meta	Произвольные метаданные
eth_dst	MAC-адрес получателя
eth_type	Тип фрейма
eth_src	MAC-адрес отправителя

Продолжение таблицы 5

vlan_vid	Номер VLAN сети
vlan_pcp	Приоритет VLAN сети
ip_dscp	Поле DSCP для QoS
ip_ecn	Поле IP ECN для сообщениях о перегрузке сети
ip_proto	Тип подпротокола IP
ip_src	IP адрес источника
ip_dst	IP адрес получателя
tcp_src	TCP порт источника
tcp_dst	TCP порт получателя
udp_src	UDP порт источника
udp_dst	UDP порт получателя
sctp_src	SCTP порт источника
sctp_dst	SCTP порт получателя
icmp_code	Тип ICMP пакета
icmp_type	Код ICMP пакета
arp_op	Тип ARP пакета
arp_spa	Адрес IP источника ARP пакета
arp_tpa	Адрес IP получателя ARP пакета
arp_sha	Адрес MAC источника ARP пакета
arp_tha	Адрес MAC получателя ARP пакета
ipv6_src	IPv6 адрес источника
ipv6_dst	IPv6 адрес получателя
ipv6_flabel	IPv6 метка потока
icmpv6_code	Тип пакета ICMPv6
icmpv6_type	Код пакета ICMPv6

Список сравнений не ограничивается только приведенными полями, каждый производитель вправе добавлять собственные поля для сравнения. Однако список действий над потоками стандартен и представлен в таблице 6, хотя производители коммутаторов OpenFlow вправе дополнять его.

Таблица 6 – Список действий над потоком.

Действие	Описание
1	2
output	Перенаправить поток в другой порт коммутатора. Требует указания номера порта (например «output:10»)
mpls_ttl	Установить значение MPLS TTL
mpls_dec	Уменьшить на единицу значение MPLS TTL
push_vlan	Инкапсулировать пакет в VLAN с указанным номером
pop_vlan	Декапсулировать пакет из VLAN
push_mpls	Инкапсулировать пакет в MPLS пакет с указанным тегом
pop_mpls	Декапсулировать пакет из MPLS
queue	Ассоциировать пакет с указанной очередью
group	Группировать запись в группу записей
nw_ttl	Установить указанное значение поля IP TTL
nw_dec	Уменьшить на единицу значение поля IP TTL
set_field	Установить произвольное значение поля пакета

Все действия могут выполняться последовательно, если их указано несколько, например действие «queue:1,output:10» означает ассоциацию с первой очередью и перенаправление в десятый порт.

2.1.3 Установка и настройка контроллеров

Контроллер NOX. Контроллер необходимо устанавливать на виртуальную или физическую ОС Ubuntu 11.04, на версии 12.04 и выше возможны проблемы

совместимости. Для установки необходимых библиотек можно воспользоваться автоматизированной установкой зависимостей, для этого необходимо выполнить следующие команды:

а) команду «cd /etc/apt/sources.list.d» для перехода в нужный каталог;

б) команду «sudo wget http://openflowswitch.org/downloads/debian/nox.list» для закачки списка серверов обновлений;

в) команду «sudo apt-get update» для обновления списка пакетоа;

г) команду «sudo apt-get install nox-dependencies git libboost-thread-dev» для установки необходимых библиотек;

д) команду «sudo apt-get install libtbb-dev && wget http://citylan.dl.sourceforge.net/project/boost/boost/1.48.0/boost_1_48_0.tar.bz2 && tar --bzip2 -xf ./boost_1_48_0.tar.bz2 && cd boost_1_48_0 && ./bootstrap.sh --prefix=/usr && sudo ./b2 install » для установки остальных библиотек;

е) команду «wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.68.tar.bz2 && tar --bzip2 -xf autoconf-2.68.tar.bz2 && cd autoconf-2.68/ && ./configure && make && sudo make install» для установки программы autoconf;

ж) если подключение к Интернету происходит через прокси-сервер, то через «git clone» скачиваться не будет:

1) способ, когда подключение к Интернету без прокси сервера. Команду «git clone http://github.com/noxrepo/nox && cd nox && ./boot.sh && mkdir build && cd build && ../configure && make» для скачивания и компиляции исходного кода контроллера NOX;

2) способ, когда подключение к Интернету через прокси-сервер. Команду «sudo apt-get install unzip && wget http://github.com/noxrepo/nox/archive/verity.zip && unzip verity.zip && mv nox-verity nox && cd nox && ./boot.sh && mkdir build && cd build && ../configure && make»

После компиляции запуск контроллера осуществляется командой «./src/nox_core –i ptcp:6633 switch», при этом запускается только приложение «switch». При прототипировании лучше использовать приложение «pyswitch», написанное на языке python и не требующее перекомпиляции. Некоторые аргументы

запуска NOX: «-t» –количество потоков; «-d» –запуск в фоновом режиме; «-v» – выводить информацию о подключении на экране.

Если контроллер запущен в фоновом режиме, то по окончанию работы его нужно остановить выполнив команды:

а) ps –ax|grep nox;

б) kill -9 «id».

Контроллер Floodlight. Floodlight написан на Java и, следовательно, работает в JVM. Перед установкой контроллера сначала потребуется установить, необходимые для работы, библиотеки:

а) JDK (Java Development Kit) – бесплатно распространяемый компанией Oracle Corporation (ранее Sun Microsystems) комплект разработчика приложений на языке Java, включающий в себя компилятор Java (javac), стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему Java (JRE);

б) Ant – утилита для автоматизации процесса сборки программного продукта, является платформонезависимым аналогом утилиты make.

Для установки контроллера необходимо выполнить следующие команды:

а) команду «sudo apt-get install build-essential default-jdk ant python-dev git», с помощью этой команды устанавливаем необходимые библиотеки и утилиты;

б) команду «git clone git://github.com/floodlight/floodlight.git && cd floodlight && ant», для загрузки ветки из git, и выполнения сборки проекта.

Результатом сборки стал исполняемый jar файл floodlight.jar. Запустить контроллер можно командой «java -jar target/floodlight.jar».

2.1.4 Тестирование работоспособности программно-конфигурируемой сети в mininet

Для проверки работоспособности программно-конфигурируемой сети необходимо сначала запустить контроллер, затем систему mininet с указанием адреса контроллера и типа контроллера «remote» командой «sudo mn --topo=linear,3 -

-mac --controller remote --ip=192.168.56.1». Эта команда создаст виртуальную сеть из трех коммутаторов и трех узлов. Коммутаторы включены последовательно. При правильной настройке команда «pingall» должна выполниться с результатом, как показано на рисунке 19.

Рисунок 19 – Результат выполнения команды «pingall»

2.1.5 Тестирование работоспособности программно-конфигурируемой сети на реальном оборудовании

Самым доступным способом реализовать OpenFlow коммутатор являются альтернативные прошивки OpenWRT для бытовых маршрутизаторов. На сайте http://www.openflow.org/wp/openwrt/ можно получить последнюю версию программного обеспечения для рзличных аппаратных платформ или скомпилировать свою версию из исходного кода. После смены ПО на маршрутизаторе необходимо настроить протокол OpenFlow в соответствии с рисунком 20, где 192.168.1.102 – адрес контроллера.

```
cd /etc/config
vi openflow
config 'ofswitch'
        option 'dp' 'dp0'
        option 'ofports' 'eth0.0 eth0.1 eth0.2 eth0.3 '
        option 'ofctl' 'tcp:192.168.1.102:6633'
        option 'mode'  'outband'
```

Рисунок 20 – Настройка OpenFlow на OpenWRT

Настройка OpenFlow на коммутаторах существенно отличается. Коммутатор обрабатывает пакеты в каждой виртуальной сети VLAN отдельно в соответствии с

настройками, для включения OpenFlow одна сеть VLAN должна иметь IP адрес и должна быть возможность коммуникации между этой сетью и контроллером. Остальные VLAN можно использовать как OpenFlow сети.

На коммататорах HP Procurve для этого необходимо выполнить команду «openflow номер_vlan controller tcp:адрес:порт», где «номер_vlan» – это номер той сети, которая будет под управлением OpenFlow, «адрес:порт» – адрес контроллера. В таблице 7 показаны основные команды настройки таких коммутаторов

Таблица 7 – Команды настройки коммутаторов HP

Команда	Описание
1	2
openflow <vlan_id> {enable/disable}	Включает и выключает OpenFlow в сети «vlan_id»
openflow <vlan_id> backoff <backoff>	Задает таймаут «backoff» соединения с сервером в сети «vlan_id»
openflow sw-rate <rate>	Задает предел производительности при обработке пакетов в программном режиме
openflow hw-rate <rate>	Задает предел производительности при обработке пакетов в аппаратном режиме
openflow <vlan_id> listener ptcp:<tcp_port>	Задает возможность удаленного управления коммутатором при помощи команды dpctl в сети «vlan_id»

Коммутаторы HP не имеют встроенных средств управления записями OpenFlow, для этого необходимо использовать команду «dpctl» на контроллере.

Коммутаторы NEC имеют другую команду для включения OpenFlow на VLAN «setvsi номер_vlan список_портов tcp адрес:порт», где «список_портов» – это перечисленные через запятую без пробелов номера портов для работы в OpenFlow.

2.2 Задание на лабораторную работу

В рамках данной лабораторной работы необходимо сделать следующее:

а) с помощью эмулятора mininet создать программно-конфигурируемую сеть с любой топологией на Ваш выбор;

б) изучить основные команды mininet и их параметры;

в) изучить основные команды программы dpctl;

г) установить контроллеры NOX и Floodlight;

д) протестировать работоспособность программно-конфигурируемой сети.

3 Лабораторная работа №2 – Измерение производительности программно-конфигурируемой сети

3.1 Теоретические сведения

3.1.1 Задача измерения производительности программно-конфигурируемой сети

Задача измерения задержки обработки пакетов и потерь при отсутствии записи в таблице OpenFlow коммутатора и потерь пакетов при обработке в OpenFlow контроллере NOX представляются наиболее интересными для исследования. Для исследования поведения задержки и потери пакетов необходимо построить экспериментальный сегмент сети в составе:

а) коммутатор OpenFlow;

б) компьютер с минимум тремя сетевыми картами.

На рисунке 21 представлен пример схемы экспериментального сегмента сети с пропускной способностью 1 Гб/с. На компьютере включены три гигабитных сетевых интерфейса, которые подключены к OpenFlow коммутатору. Контроллер NOX доступен на сетевом интерфейсе eth0, а интерфейсы eth1 и eth2 выполняют задачи клиентских машин.

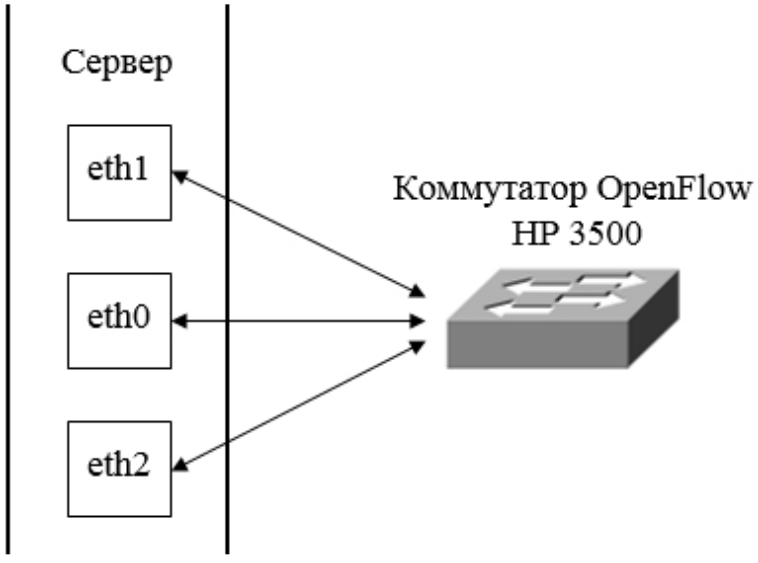

Рисунок 21 – Схема измерительного сегмента

3.1.2 Пакеты программ для измерения параметров OpenFlow

Простейшим способом измерения параметров является готовое ПО Oflops, доступное по адресу: http://archive.openflow.org/wk/index.php/Oflops. Это ПО содержит множество уже готовых алгоритмов для тестирования таких параметров, как задержки обработки коммутатора, производительность протокола OpenFlow в коммутаторе, определение ограничений производительности коммутатора, задержка определенных команд OpenFlow. Для запуска Oflops необходимо создать файл настроек. Параметры, которые необходимо указать в файле показаны в таблице 8.

Таблица 8 – Параметры Oflops

Параметр	Описание
1	2
Control_dev	Интерфейс компьютера для связи с коммутатором
Control_port	TCP порт для эмуляции контроллера
Dev	Интерфейс компьютера для эмуляции клиента. Можно указывать несколько интерфейсов
Port_num	Соответствующий интерфейсу порт коммутатора
Type	Тип перехватчика пакетов, по умолчанию – pcap
Traffic_generator	Указывает тип генератора трафика. 1 – программная генерация, 2 – генератор ядра pktgen, 3 – NETFPGA генератор
Path	Путь до вызываемого модуля с алгоритмом тестирования
Param	Параметры вызова модуля тестирования

Для измерения размеров очереди используются либо стандартные запросы статистики очередей OpenFlow, либо специфические запросы по SNMP протоколу. Основная проблема заключается в том, что каждый производитель оборудования своим образом обеспечивает поддержку очередей в OpenFlow. В большинстве случаев очереди на портах существуют в аппаратной части и поддерживают только

стандартную обработку, предусмотренную механизмами приоритезации. С помощью OpenFlow возможно только перенаправить нужный поток в нужную очередь, а в некоторых случаях и это возможно только косвенно через задание полей VLAN_PCP.

Измерение времени обработки пакета выполняется модулем openflow_action_delay в два этапа. Сначала генерируются пакеты, для которых создается простое правило перенаправления в требуемый порт коммутатора и измеряются интервалы времени между генерацией пакета и получением того же пакета на другом интерфейсе, при этом таблица потоков коммутатора остается неизменной, и событие «PacketIn» генерируется только на первый пакет. Затем в таблицу потоков вносятся изменения, изменяется порт перенаправления пакета и добавляются несколько определенных пользователем действий над пакетом перед действием на перенаправление. Время во втором этапе фиксируется таким же образом, как и на первом этапе измерения. Затем, при сравнении времен вычисляется средняя величина времени на обработку одного действия в таблице OpenFlow. Модуль имеет настраиваемую интенсивность генерации пакетов, настраиваемый список определенных пользователем действий для добавления в таблице на втором этапе измерений. Модуль сразу рассчитывает среднее, дисперсию и коэффициент вариации времени.

Модуль openflow_packet_in используется для оценки производительности коммутатора по сообщениям «PacketIn», определяющим формирование новой записи в таблице OpenFlow посредством запроса на контроллер. Для измерений модуль очищает всю таблицу коммутатора и начинает генерацию пакетов с заданной интенсивностью и размером. Одновременно по протоколу SNMP снимаются данные о загрузке процессора, потерях пакетов, по окончании эксперимента автоматически рассчитывается среднее, дисперсия и коэффициент вариации времени реакции и пропускной способности.

Модуль openflow_reactive позволяет оценить производительность связки коммутатор-СОС. Для этого с заданной интенсивностью генерируются пакеты со случайными адресами и заданным размером. Затем фиксируются времена генерации

пакета, получения сообщения «PacketIn» от коммутатора, отправки записи о потоке контроллером и время получения пакета на другом интерфейсе. На время эксперимента контроллер использует модуль эмуляции концентратора для того, чтобы не генерировать ARP пакеты. Модуль имеет настраиваемый генератор интервалов времени между исходящими пакетами. По окончании рассчитываются среднее, дисперсия и коэффициент вариации каждого интервала времени и пропускной способности.

Модуль openflow_echo_delay позволяет оценить задержку в канале управления коммутатором со стороны контроллера. Для этого по заданной интенсивности генерируется запрос «Echo», на который коммутатор должен ответить таким же пакетом, время отклика фиксируется. По окончании рассчитываются среднее, дисперсия и коэффициент вариации каждого времени отклика.

3.1.3 Использование Cbench для измерения параметров OpenFlow-сети

Cbench (контроллер Benchmarker) является расширением возможностей Oflops. Представляет собой программу для тестирования OpenFlow контроллеров путем создания пакетов в тесте для новых потоков. Cbench эмулирует определенное количество коммутаторов, которые подключаются к контроллеру, отправляют пакеты в сообщениях.

Для того, что бы установить программу Cbench, необходимо выполнить следующие команды:

а) команду «sudo apt-get install autoconf automake libtool libsnmp-dev libpcap-dev libconfig8 libconfig8-dev», для установки необходимых библиотек;

б) команду «git clone git://gitosis.stanford.edu/oflops.git && cd oflops && git submodule init && git submodule update», для скачивания пакета Oflops;

в) команду «git clone git://gitosis.stanford.edu/openflow.git && cd openflow && git checkout -b release/1.0.0 remotes/origin/release/1.0.0», для скачивания пакета OpenFlow;

г) команду «wget http://hyperrealm.com/libconfig/libconfig-1.4.9.tar.gz && tar -xvzf libconfig-1.4.9.tar.gz && cd libconfig-1.4.9 && ./configure && make», для установки остальных библиотек;

д) команду «cd netfpga-packet-generator-c-library/ && ./autogen.sh && ./configure && make», компиляция необходимого пакета для Cbench в каталоге Oflops;

е) команду «cd .. && sh ./boot.sh && ./configure && make», для компиляции Oflops;

ж) команду «cd cbench && make», для компиляции Cbench.

Ниже в таблице 9 представлены некоторые аргументы запуска Cbench.

Таблица 9 – Аргументы Cbench

Параметр	Описание
1	2
-l	Количество испытаний в тесте
-s	Количество эмулируемых коммутаторов
-m	Длительность теста
-w	Количество отбрасываемых испытаний в начале теста
-C	Количество отбрасываемых испытаний в конце теста
-D	Определяет начальную задержку
-i	Задержка между подключаемыми коммутаторами
-t	Запуск испытаний с максимальной производительностью
-c	Параметр указывает на контроллер, к которому следует подключаться (обязательный параметр). После адреса контроллера, можно указать вывод результатов в файл (>имя_файла).
-M	Количество генерируемых MAC-адресов на коммутатор

Пример запуска: ./cbench -l 64 -s 64 -m 10000 -w 5 -C 1 -D 500 -M 100000 -t -c IP_адрес_сервера>имя_файла_с_результатами.

3.1.4 Ручное измерение временных характеристик

В OpenFlow сети возможно измерение параметром задержки и потери пакетов без использования специальных программ. Для этого достаточно встроенных средств Linux. Однако необходимо использовать как минимум два компьютера.

Измерение производительности таблиц OpenFlow возможно генерацией случайных MAC адресов (mac flood). Для этого необходим пакет ettercap (доступен по адресу http://ettercap.github.io/ettercap/), запускающийся с параметрами «ettercap –TP rand_flood –i интерфейс», где «интерфейс» – сетевая карта, подключенная к OpenFlow коммутатору. На этом же компьютере и на соседнем необходимо запустить заранее программу для перехвата пакетов «tcpdump –i интерфейс –w файл» для записи трафика в файл.

После прекращения эксперимента и закрытия всех запущенных программ проводиться анализ пакетов в сохраненных файлах:

а) команда «tcpdump –r файл –ttt –q >файл2» используется для анализа временных меток и выдачи интервалов времени между пакетами;

б) команда «tcpdump –r файл –tt –q >файл2» используется для анализа временных меток и выдачи времени перехвата пакетов в микросекундах;

в) команда «tcpdump –r файл –tt –e –q >файл2» используется для анализа временных меток и MAC адресов, выдачи времени перехвата пакетов в микросекундах.

При последующем анализе результатов в табличных редакторах становиться возможным получить количество потерянных пакетов, задержки передачи пакетов.

3.2 Задание на лабораторную работу

В рамках данной лабораторной работы необходимо сделать следующее:

а) с помощью эмулятора mininet создать программно-конфигурируемую сеть с двумя хостами, контроллером и одним коммутатором OpenFlow;

б) оценить производительность коммутатора OpenFlow, воспользовавшись инструментом Oflops;

в) оценить производительность контроллеров NOX и Floodlight, воспользовавшись инструментом CBench;

г) оценить производительность контроллеров NOX и Floodlight, воспользовавшись инструментами ettercap и tcpdump;

д) сделать выводы о работе программно-конфигурируемой сети и контроллеров OpenFlow.

Список использованных источников

1 InfiniBand [Электронный ресурс] // Википедия. – Электрон. дан. – 2012. – Режим доступа : http://ru.wikipedia.org/wiki/InfiniBand. – Загл. с экрана. – (Дата обращения: 12.07.2012)

2 Storage area network. [Электронный ресурс] // Википедия. – Электрон. дан. — 2012. Режим доступа : http://en.wikipedia.org/wiki/Storage_area_network. – Загл. с экрана. – (Дата обращения: 12.07.2012)

3 Липпит, М. Технология FCoE [Электронный ресурс] / М. Липпит, Э. Смит, Э. Пейн, М. А. Де Кастро // Веб-сайт fcoe.ru. – Электрон. дан. – 2011. – Режим доступа : http://www.fcoe.ru/russian/fcoe-tehnology. – Загл. с экрана. – (Дата обращения: 04.08.2012)

4 IEEE 802.1 Data Center Bridging [Электронный ресурс] // Сайт cisco.com. – Электрон. дан. – 2011. – Режим доступа : http://www.cisco.com/en/US/netsol/ns783/index.html. – Загл. с экрана. – (Дата обращения: 15.07.2012)

5 Лебедев, С. Унификация транспорта ЦОД [Электронный ресурс] / С. Лебедев // Веб-сайт компании "Открытые технологии". – Электрон. дан. – 2009. –: http://www.ot.ru/press20090729.html. – Загл. с экрана. – (Дата обращения: 04.07.2012)

6 802.1Qbb - Priority-based Flow Control [Электронный ресурс] // IEEE 802 LAN/MAN Standards Committee. – Электрон. дан. – 2011. – Режим доступа : http://www.ieee802.org/1/pages/802.1bb.html. – Загл. с экрана. – (Дата обращения: 21.07.2012)

7 802.1Qaz - Enhanced Transmission Selection [Электронный ресурс] // IEEE 802 LAN/MAN Standards Committee. – Электрон. дан. – 2011. – Режим доступа : http://www.ieee802.org/1/pages/802.1az.html. – Загл. с экрана. – (Дата обращения: 11.08.2012)

8 802.1Qau - Congestion Notification [Электронный ресурс] // IEEE 802 LAN/MAN Standards Committee. – Электрон. дан. – 2011. – Режим доступа :

http://www.ieee802.org/1/pages/802.1au.html. – Загл. с экрана. – (Дата обращения: 24.08.2012)

9 Ling, J. Virtual output queue (VoQ) management method and apparatus / J. Ling, J. C. Calderon, J. M. Caia, A. T. Huang, V. Joshi // US7295564 : Патент. – USA, 13.11.2007.

10 Технология Brocade Virtual Cluster Switching [Электронный ресурс] // Веб-сайткомпании Brocade. – Электрон. дан. – 2012. – Режим доступа : http://brocade.ocs.ru/-vcs. – Загл. с экрана. – (Дата обращения: 21.07.2012)

11 Башилов, Г. Программно-аппаратная идиллия или OpenFlow / Г. Башилов // Журнал сетевых решений/LAN. – Москва : Открытые системы, 2011. – с. 9.

12 Intro to OpenFlow [Электронный ресурс] // Open Networking Foundation. – Электрон. дан. – 2011. – Режим доступа : https://www.opennetworking.org/standards/intro-to-openflow. – Загл. с экрана. – (Дата обращения: 15.07.2012)

13 NOX-Classic wiki [Электронный ресурс] // NOX Network Control Platform. – Электрон. дан. – 2012. – Режим доступа : https://github.com/noxrepo/nox-classic/wiki. – Загл. с экрана. – (Дата обращения: 04.08.2012)

14 Using and Programming Maestro [Электронный ресурс] // A scalable control platform writen in Java which supports OpenFlow switches. – Электрон. дан. – 2012. – Режим доступа : http://maestro-platform.googlecode.com/files/programming.pdf. – Загл. с экрана. – (Дата обращения: 09.08.2012)

15 Erickson, D. What is Beacon? [Электронный ресурс] / D. Erickson // Beaconhome. – Электрон. дан. – 2012. – Режим доступа : https://openflow.stanford.edu/display/Beacon/Home. – Загл. с экрана. – (Дата обращения: 15.07.2012)

16 Trema readme [Электронныйресурс] // Trema - OpenFlow controller framework. – Электрон. дан. – 2012. – Режим доступа : https://github.com/trema/trema/blob/develop/README.md. – Загл. с экрана. – (Дата обращения: 24.07.2012)

17 Simple Network Access Control (SNAC) [Электронный ресурс] // Open Networking Foundation. – Электрон. дан. – 2012. Режим доступа : http://www.openflow.org/wp/snac/. – Загл. с экрана. – (Дата обращения: 16.08.2012)

18 Big Network Controller [Электронный ресурс] / Сайт Big Switch Networks – Электрон. дан. – 2013. – Режим доступа: http://www.bigswitch.com/news-events. – Загл. с экрана. – (Дата обращения: 20.04.2013)

19 Floodlight OpenFlow Controller [Электронный ресурс] // Сайт Project Floodlight. – Электрон. дан. – 2013. – Режим доступа: http://www.projectfloodlight.org/floodlight/. – Загл. с экрана. – (Дата обращения: 01.04.2013)

20 McKeown, N. OpenFlow: Enabling Innovation in Campus Networks / N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner // ACM SIGCOMM Computer Communication Review. – New York, 2008. – т. 38, №2. – с. 69-74.