Эта часть работы выложена в ознакомительных целях. Если вы хотите получить работу полностью, то приобретите ее воспользовавшись формой заказа на странице с готовой работой:
https://studservis.ru/gotovye-%D0%BD%D0%B0%D1%8F%20%D0%BA%D0%B2%D0%B0%D0%BB%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B

Тип работы: ВКР (Выпускная квалификационная работа)

Предмет: Многоканальные телекоммуникационные системы

Содержание

Введение
Программно-конфигурируемые сети (ПКС) вызывают разнополярные отклики со стороны представителей научных центров, компаний-производителей микропроцессорной техники и обычных пользователей.
Так, одно из самых распространенных мнений по поводу ПКС, что они не нужны, потому как и так все хорошо работает. Также есть мнение, что ПКС – уловки маркетологов для того, чтобы продать новый продукт. Есть мнение, что ПКС не приживутся в реальной жизни, а хороши лишь как объект исследований в лабораториях. Если кто-то и отзывается о ПКС с интересом, то все равно считают их ненужными в реальных ситуациях. Тем не менее, забывать эту тему ни ученные, ни производители, не торопятся.
По прогнозам ведущих производителей программного обеспечения, в течении ближайших нескольких лет объем трафика увеличится в четыре раза, а число мобильных устройств, подключенных к интернету будет превышать число людей на планете.
Глава 1 Программно-конфигурируемые сети
1.1 Обзор архитектуры программно-конфигурируемой сети.
Во время выступления в мае 2013 года в университете Стэнфорда являющийся одним из авторов в подходе программно-конфигурируемых сетей, Скот Шенкер подвел итоги пяти лет развития

технологии. При этом он сделал акцент на четырех положениях, которые изначально как показала практика, были ошибочными.
Так, первоначально предполагалось, что:
1) управляющая программа должна самостоятельно выполнять конфигурирование всех коммутаторов,
2) коммутаторы являются функционально однородными и играющими одну и ту же роль,
3) сеть представляет собой набор аппаратных коммутаторов,
4) плоскость данных проста и не включает в себя никаких (middlebox) промежуточных устройств обработки.
Но эти предположения оказались ошибочными.
Так, исследования показали, что физические топологии являются слишком сложными для непосредственного управления. Кроме того, приложения должны взаимодействовать только с виртуальными представлениями. Также, эксперименты показали, что коммутаторы в ядре сети и на ее периметре принципиально отличаются друг от друга.
Также, количество виртуальных программных коммутаторов превышает число физических, а промежуточные устройства обработки применяются массово.
Трудно возразить на данные выводы, но также сложно игнорировать тот факт, что компания Nicira [1], которая была основана Шенкером, начала свою деятельность около десяти лет назад созданием виртуального коммутатора OpenvSwitch, которая имеет в виде основного продукта платформу виртуализации сетей NVP (Network Virtualization Platform).

Затем, в 2012 году компания VMware приобрела Nicira.
Но на интеграцию технических решений потребовалось значительно больше времени, чем предполагалось.
Лишь по прошествии года, в 2013 года получилось проанонсировать систему VMware NSX, которая могла бы собрать воедино NVP и vCloud Networkind and Security (vCNS).
Платформа NSX имеет своим назначением создание уровня виртуализации над существующим оборудованием и предоставление интерфейса к упрощенным логическим устройствам. К этим устройствам относятся порты, коммутаторы, маршрутизаторы, распределенные межсетевым экранам.
В функции данной платформы входит, в том числе и обеспечение безопасности и качества обслуживания, а также, мониторинг.
Программный интерфейс (API) позволил объединить логические сетевые устройства в топологию, создавшую виртуальную сеть с изоляцией трафика. Помимо этого, имеет место полная поддержка большого количества (multi tenant) пользователей.
Базовые компоненты NSX представляют собой:
1) виртуальные коммутаторы под управлением гипервизора,
3
2) шлюзы,
3) кластер контроллеров,
4) компоненты сторонних производителей и управляющая программа NSX Manager.
Кластер контроллеров отвечает за развертывание всей виртуальной архитектуры, принимая запросы от платформ управления (vCloud, OpenStack), рассчитывая виртуальную сетевую топологию и программируя
виртуальные коммутаторы и шлюзы в упреждающем режиме (proactively).
Для того, чтобы решать данные задачи, в доступности находятся данные по всем виртуальным машинам и сервисам NSX [2].
Элементы кластера представляются равноправными и взаимозаменяемыми элементами и для того, чтобы масштабировать систему, достаточным действием будет добавление дополнительных узлов.
В состав виртуального коммутатора в гипервизоре входит высокопроизводительный модуль ядра. Кроме того, здесь имеются и базы данных по конфигурации, а также по программируемой плоскости данных с уровнями L2-L4.
Помимо виртуализации топологии, подобная архитектура позволяет сделать виртуальной и сетевую безопасность. Это позволит отвязать политику безопасности от IP-адресов и дать возможность для реализации межсетевых экранов в модулях гипервизора, непосредственно взаимодействующих с виртуальными машинами.
Таким образом, невозможно говорить о ПКС с отрицательной точки зрения, хотя и положительных изменений пока не наблюдается.
Можно сказать, что архитектура программно-конфигурируемых сетей имеет трехуровневый вид, как это показано на рис.1.
Таким образом, здесь имеет место:
- уровень сетевых устройств,
- уровень линий связи;
- уровень управления;
- уровень приложений, в котором реализуются различные функции обработки сетевого трафика.

1.2 Протокол Open Flow

В основе архитектуры программно-конфигурируемых сетей лежит
открытый протокол OpenFlow, осуществляющий управление сетевой архитектурой.
Преимуществом этого протокола является то, что он не принадлежит конкретному производителю сетевого оборудования, и, следовательно, применим для широкого спектра устройств различных производителей [6].
4
Протокол OpenFlow позволяет задавать конфигурацию сетевых устройств с централизованного аппарата управления вычислительной сетью – контроллера ПКС. Основными функциями контроллера ПКС являются добавление и удаление записей из таблицы сетевых потоков на

устройстве. Контроллер обеспечивает динамический анализ состояния всей сети, благодаря чему предоставляет системному администратору возможность оценки состояния сетевой инфраструктуры в данный момент. На контроллере ПКС установлена сетевая операционная система, которая поддерживает систему управления сетью. К задачам такой системы управления сетью относятся: сбор сведений о компонентах сети, сбор статистики взаимодействий внутри сети, настройка сетевого оборудования. Администратор сети имеет возможность задавать правила передачи данных в сети.

Глава 2 Архитектура приложения. Критерии хорошей архитектуры

Применимо к программированию речь чаще всего идет об архитектуре программного обеспечения. Однако внутри понятия «архитектура программного обеспечения» можно столкнуться с такими составляющими этого общего понятия, как корпоративная архитектура, системная архитектура, организационная архитектура, архитектура информации, архитектура аппаратного обеспечения, архитектура приложения, архитектура инфраструктуры и так далее. Рассмотрим критерии хорошей архитектуры.

Несмотря на то, что общепринятого термина «архитектура программного обеспечения», нет, все равно, на практике большинство разработчиков знают и так, чем отличаются хорошие коды от плохих. Поэтому можно сказать, что хорошей архитектурой является выгодная архитектура. Она должна делать процесс разработки и сопровождения программы проще и эффективнее. Программу, у которой хорошая архитектура, легче расширять и изменять. Также будет легче даваться тестирование, отлаживание и понимание. Таким образом, представляется возможным сформулировать список критериев хорошей архитектуры [9]:

- Эффективность системы.

Программа, прежде всего, конечно должна справляться с решением поставленных задач и быть способной хорошо выполнять свои функции в разнообразных условиях. Здесь же нужно упомянуть надежность, безопасность, производительность, способность справляться с увеличением нагрузки (масштабируемость).

- Гибкость системы.

Приложения склонны к тому, чтобы их со временем меняли. Насколько быстро и удобно можно вносить изменения в существующий функционал, насколько меньше проблем и ошибок это вызовет, настолько система является гибкой и конкурентоспособной. Так как в процессе разработки важно оценивать то, что получается, на предмет того, как вам это потом, возможно, придется менять.

-Расширяемость системы.

Этот термин обозначает допустимость добавления в систему новых функций, без нарушения ее базовой функции. Начальный этап характеризуется загрузкой в систему только основной и самой необходимой функции и при этом архитектура должна давать возможность легкого наращивания дополнительных функционалов по мере надобности.

Требование, чтобы архитектура системы обладала гибкостью и расширяемостью (то есть была способна к изменениям и эволюции) является настолько важным, что оно даже сформулировано в виде отдельного принципа — «Принципа открытости/закрытости» (Open-Closed Principle — второй из пяти принципов SOLID): Программные сущности (классы, модули, функции и т.п.) должны быть открытыми для расширения, но закрытыми для модификации.

Иными словами: Должна быть возможность расширить/изменить поведение системы без изменения/переписывания уже существующих частей системы.

То есть, приложение надо бы спроектировать так, чтобы изменение его поведения и добавление новой функциональности достигалось бы за счет написания нового кода (расширения), и при этом не приходилось бы менять уже существующий код и в подобном случае новых требований не повлечет за собой модификацию существующей логики, а сможет быть реализовано прежде всего за счет ее расширения. Именно этот принцип является основой «плагинной архитектуры» (Plugin Architecture). О том, за счет каких техник это может быть достигнуто, будет рассказано дальше.

- Масштабируемость процесса разработки.

Возможность сократить срок разработки за счёт добавления к проекту новых людей. Архитектура должна позволять распараллелить процесс разработки, так чтобы множество

людей могли работать над программой одновременно.
- Тестируемость.
Код, который можно будет с легкостью протестировать, будет иметь меньшее количество ошибок и будет надежнее работать. Однако тестируемость не только хорошая для качества кода, но и на хороший дизайн. Существует целая методология разработки программ на основе тестов, которая так и называется — Разработка через тестирование (Test-Driven Development, TDD).
Возможность повторного использования. Систему желательно проектировать так, чтобы ее фрагменты можно было повторно использовать в других системах [10].
Когда речь идет о построении архитектуры программы, создании ее структуры, под этим, главным образом, подразумевается декомпозиция программы на подсистемы (функциональные модули, сервисы, слои, подпрограммы) и организация их взаимодействия друг с другом и внешним миром. Причем, чем более независимы подсистемы, тем безопаснее сосредоточиться на разработке каждой из них в отдельности в конкретный момент времени и при этом не заботиться обо всех остальных частях.
6
В виду того, что, как говорилось выше, программное приложение является частью программного обеспечения, значит все критерии «хорошей архитектуры», применимые к архитектуре программного обеспечения можно применять и для архитектуры программного приложения [11].
На рисунке 7 приведена предлагаемая для реализации структура программного приложения.
Корпоративная сеть в наше время представляется с использованием различных служебных и специализированных сервисов, компьютеров, мобильных ресурсов. Они сочетают в себе различные комбинационные наборы по видео трафику, технологиям интеграции с социальными сетями, современным мульти-платформенным приложениям и т.д.
лава 3 Разработка метода выбора архитектуры приложений для программноконфигурированных сетей.
Первый этап на пути разработки метода выбора архитектуры приложений для программноконфигурационных сетей был в данном исследовании проведен в главе 2, где были сформулированы критерии «хорошей архитектуры».
Записи о потоках могут также указывать на группы, в которых осуществляется дополнительная обработка. Группы представляют собой наборы инструкций для различных вариантов совместной обработки потоков: широковещательной рассылки, агрегирования каналов. Наличие групповых методов обработки позволяет эффективно применять общие выходные действия для потоков.
Установка, обновление и удаление правил выполняется коммутатором. Могут быть выделены два режима функционирования:
- проактивный – установка правил до получения пакетов;
- реактивный – установка правил в ответ на пришедшие пакеты.
Коммутатор проверяет все записи на наличие перекрывающихся записей, перед тем как выполнять запрос OFPFC_ADD с установленным флагом OFPFF_CHECK_OVERLAP. Две записи в таблице потоков считаются перекрывающимися, если пакет соответствует обеим записям. У этих записей должен быть установлен одинаковый приоритет. Если имеется конфликт перекрытия между существующей записью и запросом добавления, коммутатор игнорирует добавление и реагирует посылкой сообщения:
ofp_error_msg.
Если запрос не требует проверки на наличие перекрывающихся записей, то коммутатор вставляет данную запись в таблице потоков. Если идентичная запись уже есть в таблице, то такая запись должна быть удалена, а на ее место помещена новая запись. При выполнении данной операции коммутатор не обязательно генерирует оповещение о проведенной операции [13]. Запрос на уведомление осуществляется с использованием специального флага.
7
Если коммутатор получает запрос на удаление, то он выполняет поиск записи в таблице. Если соответствующая запись имеется в таблице, то она удаляется. При выполнении данной операции коммутатор не обязательно генерирует оповещение о проведенной операции. Запрос на уведомление осуществляется с использованием специального флага OFPFF_SEND_FLOW_REM, установленного в 1. Если запись не была найдена в таблице, то

удаления записи в таблице не происходит. При этом коммутатор не обязательно генерирует оповещение о неуспехе проведенной операции.
Если коммутатор получает запрос модификации записи, он выполняет поиск записи в таблице. Если соответствующая запись имеется в таблице, то поле инструкции этой записи обновляется значением из запроса. При этом остальные поля записи остаются неизменными кроме случая, когда установлен специальный флаг сброса счетчиков (FPFF_RESET_COUNTS). Если запись не была найдена в таблице, модификации таблицы не происходит. При этом коммутатор не обязательно генерирует оповещение о неуспехе проведенной операции.

Заключение

По прогнозам ведущих производителей программного обеспечения, в течении ближайших нескольких лет объем трафика увеличится в четыре раза, а число мобильных устройств, подключенных к интернету будет превышать число людей на планете.
Существующая архитектура сетей развивалась по принципу «ласточкиного гнезда», а именно, при появлении проблем к стеку протоколов TCP/IP добавлялся новый, который эту проблему решал. Однако позже становилось ясно, что этот протокол также имеет свои минусы, таким образом, добавляя его, происходило увеличение общей погрешности.

Список использованной литературы

1. ComputerWorld. Nicira: A classic Silicon Valley Story [Электронный ресурс] – URL:https://www.computerworld.com/article/2504909/networking/nicira--a-classic-silicon-valleystory--complete-with-big-payday.html
2. Brand Hedlund. Network Virtualization: a next generation modular platform for the data center virtual network. [Электронный ресурс] – URL: http://bradhedlund.com/2013/01/28/networkvirtualization-a-next-generation-modular-platform-for-the-virtual-network/
3. 001 Wikipedia Digital Diagnostics monitoring. [Электронный ресурс] https://ru.wikipedia.org/wiki/DDM Дата обращения 31.03.2020
4. Habr. Гипервизоры. Что это и как работает виртуальный сервер.[Электронный ресурс] – URL: https://habr.com/company/vps_house/blog/349788/
8
5. Ананченко И. В., Шамилова Р. В., Завальнюк Д. А. О некоторых аспектах защиты информации в многоканальных телекоммуникационных системах [Электронный ресурс]. - 2019. - URL:
6. Yamahata, I. Software Defined Networking, openflow protocol and its controllers [Электронный ресурс] – URL: https://www.valinux.co.jp/english/
7. Смелянский Р. Программно-конфигурируемые сети [Электронный ресурс]. – 2012. – URL: http://www.osp.ru/os/2012/09/13032491/
8. Перепелкин Д.А. Концептуальный подход динамического формирования трафика программно-конфигурируемых телекоммуникационных сетей с балансировкой нагрузки // Информационные технологии. - 2015. - Том 21. -№ 8. - С. 602-610.
9. Черняк, Л. SDN — первое знакомство [Электронный ресурс] / Л. Черняк // Журнал сетевых решений/LAN. – Электрон.журн. – 2011. Режим доступа: http://www.osp.ru/news/articles/2011/29/13009476/, свободный. Дата обращения: 10.03.2020
10. Introduction to Test Driven Development (TDD) http://agiledata.org/essays/tdd.html
11. Завальнюк Д. А., Каламбет М. В., Картузова Е. И. Рассмотрение архитектурных решений программно-конфигурируемых сетей [Электронный ресурс]. - 2019. - URL:
12. Uppal H., Brandon D.. OpenFlow Based Load Balancing [Электронный ресурс] – URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.5150&rep=rep1&type=pdf
13. M. Reitblatt, N. Foster, J. Rexford, D. Walker. Consistent Updates for Software-Defined Networks: Change You Can Believe In! [Электронный ресурс] – URL: http://conferences.sigcomm.org/hotnets/2011/papers/hotnetsX-final3.pdf
14. Спецификация: OpenFlow Switch Specification version 1.3.2 [Текст]. – 2013. – 56.
15. Завальнюк Д. А., Каламбет М. В. Яготинцева Н. В. Концептуальная модель и методы контроля программного-конфигурируемых сетей [Электронный ресурс]. - 2019. - URL:
16. B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, “A Slick Control Plane for Network Middleboxes,” [Электронный ресурс] – URL: http://nextstepesolutions.com/Clients/ONS2.0/

pdf/2013/research track/poster papers/final/ons2013-final51.pdf
17. Brunner, M. Programmable Flow-based Networking with OpenFlow [Электронный ресурс] / http://docbox.etsi.org/workshop/2010/201003_FNTWORKSHOP/7_ARCHITECTURE/BRUNNER_OpenFlow.pdf
18. Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, “SIMPLE-fying Middlebox Policy Enforcement Using SDN.”/ Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, ACM SIGCOMM, August 2013.
19. Phillip Porras, Steven Cheung, Martin Fong, Keith Skinner. “Securing the Software-Defined Network Control Layer”. [Электронный ресурс] – URL:http://www.csl.sri.com/users/porras/SEFloodlight.pdf.

Эта часть работы выложена в ознакомительных целях. Если вы хотите получить работу полностью, то приобретите ее воспользовавшись формой заказа на странице с готовой работой:
https://studservis.ru/gotovye-
%D0%BD%D0%B0%D1%8F%20%D0%BA%D0%B2%D0%B0%D0%BB%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B