Программируемый Интернет

Интернет все глубже проникает в нашу жизнь и продолжает удивлять новыми возможностями. Но в основном развитие сегодняшнего Интернета происходит на уровне приложений. Каждый день приносит нам сотни, если не тысячи[1], новых “apps”. Спускаясь ниже, к сетевому уровню, мы обнаружим, что здесь развитие происходит существенно медленнее. Безусловно, «количественно» базовая инфраструктура Интернета продолжает стремительно развиваться – растет общая пропускная способность, внедряются новые технологии канального уровня, но архитектурно изменения до последнего времени были весьма незначительны. Одной из причин этому является то, что сетевая архитектура представляет собой достаточно монолитный блок, включающий множество функций, в том числе и сетевые «приложения». Внедрение новой функциональности требует модернизации всего сетевого стека в миллионе устройств. Представьте, что вам пришлось бы делать апгрейд операционной системы компьютера каждый раз, когда вы устанавливаете новое приложение!

Другими словами, инновация на сетевом уровне в рамках сегодняшней архитектуры весьма затруднительна. Новые функции и возможности неимоверно увеличивают сложность системы, их тестирование трудоемко, а внедрение дорогостояще и рискованно. Поэтому многие связывают большие надежды с новой сетевой парадигмой, получившей название SDN (Software Defined Networking) или Программно-Конфигурируемая Сеть.

На очередной конференции IETF (https://www.ietf.org) летом прошлого, 2012 года техническое пленарное заседание почти целиком было посвящено SDN. В октябре, на встрече ENOG4 (http://www.enog.org/meetings/enog-4/) в Москве доклад об OpenFlow вызвал живой интерес среди участников. Наконец, в ноябре, на конференции IETF85 (https://www.ietf.org/proceedings/85/agenda.html) состоялся BOF на тему интерфейса к системе маршрутизации. Open Networking Foundation(ONF, https://www.opennetworking.org/) - организация, существующая чуть дольше полутора лет, занятая развитием SDN и разработкой связанных с ней открытых стандартов, наиболее известным из которых является OpenFlow.

Очевидно, что тема новая и животрепещущая, но попробуем разобраться в сути происходящего и отделить маркетинг от новой архитектурной парадигмы.

Пакеты, коммутация и маршрутизация

Для того, чтобы лучше представить архитектурный сдвиг, предлагаемый SDN, посмотрим сначала как происходит маршрутизация в IP-сетях.

Различают внутреннюю (по отношению к сети) и внешнюю, или межсетевую маршрутизацию. Эти две системы используют различные протоколы, например, для внутренней маршрутизации широко используется протокол OSPF (http://ru.wikipedia.org/wiki/OSPF), а стандартом межсетевой маршрутизации является BGP (http://ru.wikipedia.org/wiki/Bgp). Ключевым элементом сетевой инфраструктуры является маршрутизатор,который выполняет две функции: маршрутизацию и пересылку пакетов с одного интерфейса на другой. Хотя иногда под маршрутизацией понимают обе функции, на самом деле они существенно различаются.

В алгоритмическом плане пересылка пакетов достаточно проста, и основной задачей является производительность. Отправителями и получателями пакетов являются интерфейсы маршрутизатора, а определение адресата осуществляется с помощью т.н. таблицы передачи - Forwarding Information Base (FIB).

Задачей же маршрутизации является построение собственной таблицы, таблицы маршрутизации (Routing Information Base, RIB), которая потом транслируется в таблицу FIB. Для построения этой таблицы и используются протоколы маршрутизации, которые на основе информации, полученной от соседних маршрутизаторов (например, о связности и доступности тех или иных маршрутов), и собственной конфигурации (например, статических маршрутов и ограничений, наложенных сетевой политикой маршрутизации), формируют собственное представление о сети и ее топологии.

В контексте данной темы важным в этой модели является то, что каждый маршрутизатор принимает решения самостоятельно и относительно независимо.

Такая модель работает замечательно в простых сетях, особенно когда основной задачей является

[1] http://readwrite.com/2013/01/07/apple-app-store-growing-by

обеспечение связности. Она проста и надежна. Однако по мере усложнения сетевой архитектуры и политики внутренней и внешней маршрутизации, ограничения модели становятся все более заметны.

Например, необходимость включения в политику маршрутизации требований обеспечения определенных параметров качества, производительности и стоимости зачастую наталкивается на ограничения существующих протоколов маршрутизации. Попытки удовлетворить эти требования в рамках протоколов уже привели к их значительному усложнению и созданию закрытых расширений.

Эти проблемы также присутствуют и в сетях Ethernet или MPLS, когда автономность элементов и связанная с этим необходимость распределенной конфигурации существенно усложняют управление сетью.

Программно-конфигурируемая сеть

Концепция SDN позволяет превратить разработку и конфигурирование сети в задачу программирования.

В традиционной архитектуре уровень управления (т.н. control plane), к которому относится, например, процесс маршрутизации, и уровень передачи данных (т.н. data plane), отвечающий за пересылку пакетов с одного интерфейса на другой, сосуществуют в одном устройстве. Концепция SDN предусматривает передачу управляющих функций центральному серверу - т.н. контроллеру, таким образом заменяя традиционную распределенную модель маршрутизации на централизованную. Соответственно и процесс управления сетью, включающий создание маршрутов, является ни чем иным как программированием сети в целом. Для сравнения на рис.1 приведены традиционная сетевая архитектура и сеть SDN.

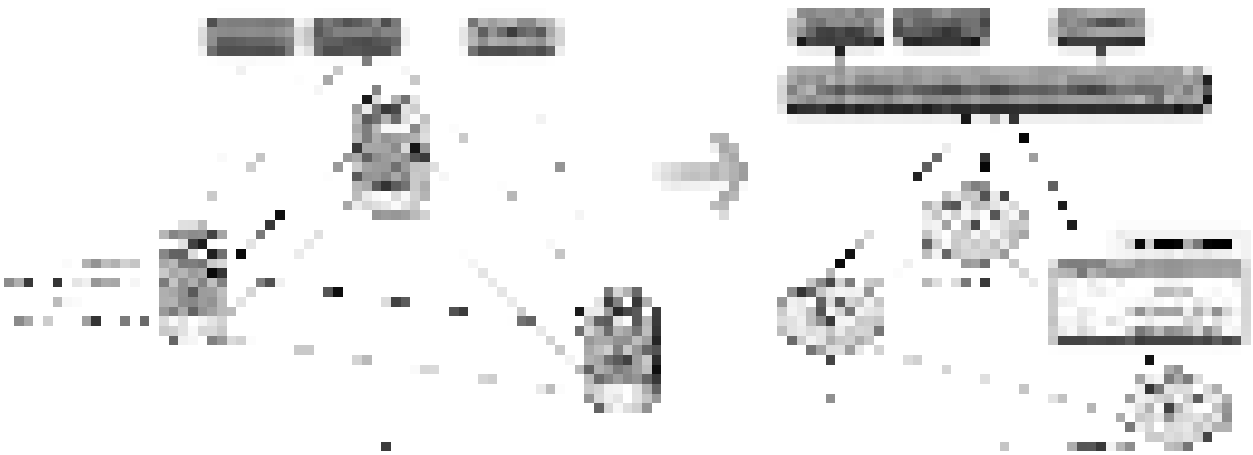

Рисунок 1. Разделение системы управления и передачи в архитектуре SDN. (a) Традиционная архитектура с автономными сетевыми элементами, b) архитектура SDN с централизованной управляющей функцией)

Такой подход обладает рядом существенных преимуществ.

Во-первых, существенно упрощается сам процесс создания маршрутов. В отличие от сегодняшней сети, где маршрутизация - это распределенный итеративный процесс, при котором рабочая топология сети «вычисляется» совместно всеми устройствами, в SDN - это программа моделирования сети с заданными параметрами. Использование этой модели открывает новые возможности для создания сети с требованиями, немыслимыми в рамках традиционного инжиниринга трафика и с использованием стандартных протоколов маршрутизации.

Рассмотрим в качестве примера реконфигурацию сети в случае падения канала между какими-либо узлами сети. В традиционной модели маршрутизатор, подключенный к этому каналу, сообщит своим соседям об этом событии и все они независимо займутся разработкой новых маршрутов. При этом они будут распространять информацию об изменяющейся топологии своим соседям и т.д. В конце концов этот итерационный процесс закончится и сеть перейдет в новое состояние. В зависимости от сложности сети и используемых протоколов маршрутизации, этот процесс займет больше или меньше времени, но учитывая задержки при передаче информации, на каждой итерации произойдет совсем не мгновенно. Более того, этот процесс не является детерминированным, другими словами, повторись падение того же канала – не факт, что сеть перейдет в то же состояние.

В случае использования управляющего центра расчет новой топологии производится исходя из знания о всей сети в целом. Мы также можем задать необходимую топологию следующего состояния. Наконец, поскольку создание новой топологии - это число вычислительный процесс, он может быть выполнен значительно быстрее.

Во-вторых, значительно увеличиваются возможности для инновации. В традиционной распределенной модели необходимость приведения функциональности к общему знаменателю для возможности взаимодействия между независимыми устройствами определяет существенную консервативность системы по отношению к новшествам. Это приводит к «технологической окостенелости», что мы во многом наблюдаем в сегодняшней глобальной инфраструктуре Интернета. В SDN же инновация - это вопрос написания нового приложения.

Наконец в-третьих, вместо сложных и дорогостоящих маршрутизаторов можно использовать более простые устройства.

OpenFlow

SDN и OpenFlow – не одно и то же. SDN означает более общую архитектурную концепцию разъединения уровня передачи данных и уровня управления. OpenFlow – это протокол, наиболее проработанный и стандартизованный, реализующий эту концепцию. Далее мы также рассмотрим несколько другой подход к реализации SDN, основанный на программировании сети с использованием существующих протоколов маршрутизации.

Разработка OpenFlow началась с исследовательского проекта, целью которого было создание возможности разработчикам новых сетевых архитектур и протоколов опробовать свои идеи в более или менее реальной среде. Скажем, в рамках университетской сети, причем таким образом, чтобы внедрять их изолированно от существующих услуг и параметров сети, и исключить негативное влияние на текущую функциональность.

Идея OpenFlow проста и основана на наблюдении, что несмотря на существенные различия между сотнями моделей коммутаторов и маршрутизаторов, все они содержат таблицу передачи, определяющую базовую функцию передачи данных – для каждого входящего пакета переправить его как можно быстрее на определенный исходящий интерфейс. Более того, хотя формат этих таблиц различен, можно идентифицировать стандартный набор функций данного уровня.

Каждая запись абстрактной таблицы передачи OpenFlow является “правилом” и связана с так называемым «потоком» данных (flow). Поток определяется заголовком пакета – например, комбинацией адресов MAC, IP и номеров портов источника и получателя данных, хотя в принципе поток может состоять из пакетов с определенным (нестандартным) заголовком – например, для поддержки новых нестандартных протоколов. Не все элементы этой комбинации должны быть определены – например, поток может быть определен как весь трафик к некоторому хосту. В этом случае определенным является только один элемент – IP-адрес получателя данных.

Другим элементом записи таблицы является «действие» (action), определяющее требуемую обработку пакетов потока. Основных действий четыре:

1. Передать пакет на определенный порт (или определенные порты) коммутатора.

2. Передать пакет контроллеру через «защищенный» канал. Контроллер – это управляющий центр сети, включающий центральную сетевую операционную систему и управляющие приложения, рассчитывающий топологию и маршруты, а также осуществляющий другие функции управления. Поэтому, как правило, первый пакет неизвестного потока отправляется контроллеру для определения правила и создания новой записи таблицы передачи.

3. Отбросить пакет. Это действие может быть необходимым, например, в борьбе с компьютерными атаками.

4. Наконец, пакет может быть направлен на «стандартную» обработку, например если OpenFlow-коммутатор также является стандартным коммутатором, маршрутизатором и т.п. Данная функциональность позволяет разделить потоки данных на потоки, управляемые OpenFlow, и потоки, управляемые другими механизмами, например, с помощью существующих протоколов маршрутизации. Благодаря этому, например, исследователи могут изолировать экспериментальный трафик от нормального, используя общую инфраструктуру.

Наконец, последний элемент содержит различную статистику – продолжительность потока, число полученных и переданных пакетов и т.п.

Рисунок 2. Упрощенная структура таблицы передачи, OpenFlow 1.0

На самом деле сегодняшняя архитектура OpenFlow, определенная текущей спецификацией 1.3.1, принятой в сентябре 2012 г., немного сложнее. В частности, она предусматривает наличие нескольких таблиц правил с возможностью каскадирования (см. Рис. 3).

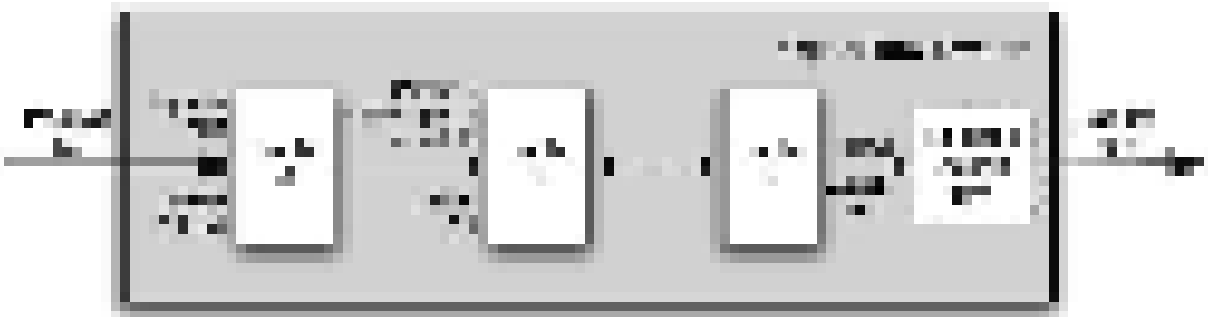

Рисунок 3. «Каскадирование» правил и действий позволяет сделать обработку более гибкой. Источник: спецификация коммутатора OpenFlow – OpenFlow Switch Specification 1.3.1, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

Такая модель открывает уникальные возможности. Коммутатор OpenFlow может быть чем угодно – от простого коммутатора до сетевого экрана и маршрутизатора. В конечном итоге все определяется таблицей передачи:

	Порт комм.	MAC src	MAC dst	Eth.type	VLAN ID	IP src	IP dst	Proto/ sport	Proto/ dport	Действие
Коммутатор	*	*	00:1F:...	*	*	*	*	*	*	Port6
Сетевой экран	*	*	*	*	*	*	*	*	22	Отбросить
Маршрутизатор	*	*	*	*	*	*	198.51.100.1	*	*	Port6
Коммутация потоков	Port3	00:20:...	00:1F:...	0800	Vlan1	192.0.2.10	198.51.100.1	6/27089	6/80	Port6

Рисунок 4. Примеры реализации различных специализированных функций коммутатора с помощью правил таблицы передачи.

Контроллер обменивается информацией с «подконтрольными» ему коммутаторами OpenFlow с помощью одноименного протокола. В рамках стандартной модели SDN в задачу коммутаторов входит без задержек передавать пакеты с одного порта на другой, осуществляя некоторую обработку в соответствии с правилами. В ответ на запрос коммутатор может сообщить контроллеру о своих возможностях

и конфигурации, а также сигнализировать об изменениях в своем состоянии, например, потерю канала или возникновение ошибки. В остальном, коммутатор полностью полагается на контроллер. Коммутатор не знает ни о топологии сети, ни даже о своих непосредственных соседях.

Задача определения «общей картины» и конфигурирование коммутаторов возлагается на контроллер, а точнее, на приложения, его использующие. Это они задают топологию сети и рассчитывают оптимальные маршруты. С помощью контроллера они устанавливают нужные правила, следят за состоянием устройств, осуществляют мониторинг трафика и сбор статистики.

Другими словами, OpenFlow обеспечивает базовые функции управления аппаратным обеспечением – коммутаторами, но не программную начинку SDN. Здесь картина более фрагментарна. Хотя многие ведущие производители сетевого оборудования, и в первую очередь – производители коммутаторов, заявляют о своей поддержке OpenFlow, программный интерфейс от контроллера к приложениям либо вообще недоступен, либо является собственной разработкой производителя.

Это, безусловно, не позволяет архитектуре SDN полностью раскрыть свой инновационный потенциал, поскольку делает невозможной разработку «сетевых мозгов», независимых от производителя контроллера.

Наиболее заметно присутствие OpenFlow на рынке коммутаторов. Стратегия ведущих производителей, включая Brocade, BigSwitch, IBM, HP и NEC, включает развитие SDN на основе OpenFlow. Правда OpenFlow реально поддерживают отдельные модели.

Наиболее впечатляющим примером использования OpenFlow на практике является распределенная внутренняя сеть Google, обеспечивающая обмен данными между датацентрами (т.н. сеть G-scale), показанная на рис. 5. Для построения этой сети компания вынуждена была разработать собственные коммутаторы, но обмен данными между ними и контроллерами использует OpenFlow.

Рисунок 5. Распределенная сеть Google (G-scale), использующая протокол OpenFlow.
(Источник: Доклад Urs Hoelzle, Google на конференции Open Networking Summit, апрель 2012г.
http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf)

Программирование системы маршрутизации - I2RS

Отсутствие стандартного подхода и программного интерфейса верхних уровней SDN также сдерживает внедрение этой технологии. Задача становится еще сложнее, если мы говорим о миграции традиционной сети в SDN. Особенно, если эта сеть обеспечивает услугу третьего уровня – маршрутизацию пакетов.

Здесь стоит также заметить, что как и многие новые архитектурные решения, SDN как концепция, и особенно в исполнении OpenFlow, вызывает некоторое отторжение как у сетевых инженеров и администраторов, так и у производителей сетевого оборудования, особенно маршрутизаторов.

Для первых SDN означает необходимость переключения на новый стиль управления сетью, если не переквалификацию в программиста. При этом неочевидно, что накопленный опыт разработки, управления и отладки сложных сетей может быть эффективно перенесен на новую платформу. Не все задачи могут быть решены через систему управления сетью или NMS (http://ru.wikipedia.org/wiki/Система_управления_сетью), и SDN представляется как одна такая мета-система.

Производители маршрутизационного оборудования видят в SDN/OpenFlow угрозу собственной конкурентоспособности. Найдется хотя бы несколько компаний, оборудование которых переключает пакеты быстрее, чем, скажем, маршрутизаторы Juniper. Наверное немало компаний смогут разработать программное обеспечение лучше, чем, например, Cisco. Эффективное объединение этих компонентов в одном продукте и громадная внедренная база – это преимущество может быть утрачено с приходом SDN, когда потребитель получит возможность сам выбирать и комбинировать лучшее сетевое программное обеспечение с лучшим аппаратным решением. Кстати, это является одной из причин, почему и Cisco и Juniper предлагают SDN в собственной закрытой упаковке, означающей по-существу улучшенный (по сравнению с пользовательским интерфейсом командной строки!) программный интерфейс к своим маршрутизаторам.

Однако, возвращаясь к вопросу миграции к новой технологии и архитектуре, лучшая программируемость сети – это уже существенный шаг вперед. Особенно если он позволяет решить некоторые насущные проблемы сетевых администраторов.

Попытки решить эту задачу предпринимались давно. Помимо повсеместно присутствующего интерфейса командной строки, сегодня существует ряд механизмов, позволяющих программно взаимодействовать с маршрутизатором.

Наиболее распространенным является протокол SNMP (Simple Network Management Protocol, Простой протокол управления сетью, http://ru.wikipedia.org/wiki/SNMP), который позволяет получать от устройства информацию о его состоянии и конфигурацию, а также различную статистику. Для этого каждое устройство содержит т.н. базу управляющей информации, или MIB (Management Information Base, http://ru.wikipedia.org/wiki/Management_Information_Base), где хранятся параметры, доступные по SNMP. Обычно SNMP используется для сбора статистики и мониторинг состояния устройств и их конфигурации, однако эта система крайне редко используется для конфигурации устройства. Отсутствие необходимых функций, таких как возможность «отката», понятия тестовой конфигурации и т.п. не позволяет использовать SNMP в этом режиме в производственных условиях.

Другой сетевой протокол конфигурации, NetConf (http://ru.wikipedia.org/wiki/NETCONF), также разработанный в IETF, снимает большинство ограничений SNMP, однако отсутствие стандартной модели данных ограничивает его применение.

Система ForCES (Forwarding and Control Element Separation – разделение элементов управления и передачи) предлагает принципиально другой метод конфигурирования маршрутизатора. Эта архитектура, разработанная в IETF еще в 2004 году, также как и в случае OpenFlow, предусматривает разделение монолитного устройства на управляющий и передающий элементы (более подробно о различиях между OpenFlow и ForCES см. http://tools.ietf.org/html/draft-wang-forces-compare-openflow-forces). Однако, как и в OpenFlow, программный интерфейс относится к уровню передачи, тем самым не позволяя использовать «интеллектуальную начинку» маршрутизатора – логику протоколов маршрутизации и построения маршрутизационной таблицы.

Для решения задачи обеспечения программного интерфейса к системе маршрутизации устройств, в IETF недавно создана Рабочая Группа I2RS (Interface to the Routing System, http://datatracker.ietf.org/wg/i2rs/).

Подход I2RS направлен на использование существующей информации, собранной самим маршрутизатором о топологии сети, ее состоянии, а также о собственном состоянии и конфигурации. Ограничиваясь интерфейсами уровня системы маршрутизации, I2RS также использует существующие средства преобразования маршрутизационной информации в таблицы передачи FIB и LIB. Общая архитектура I2RS приведена на рис.6.

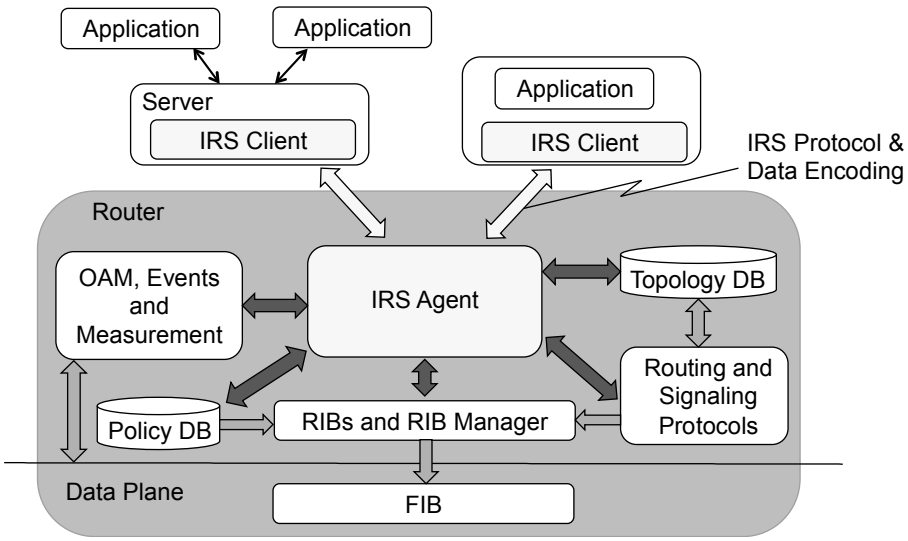

Рисунок 6. Архитектура системы I2RS. Желтым цветом обозначены элементы и протоколы I2RS.
(С разрешения A. Farrel, Juniper Networks, Old Dog Consulting)

Для иллюстрации возможностей I2RS рассмотрим один из примеров возможного применения – установку статического маршрута – часто встречающуюся задачу конфигурации сети, особенно при инжиниринге трафика. Традиционные способы включают использование интерфейса командной строки и MIB, но оба не поддерживают программного интерфейса. Обеспечивая доступ к RIB, LFIB, и Multicast RIB, I2RS позволяет решить эту задачу на программном уровне.

Другим примером является возможность программного доступа к функциям «policy-based routing» (PBR, http://en.wikipedia.org/wiki/Policy-based_routing), а именно, к правилам обработки определенного трафика.

При этом речь идет не только о доступе на запись, например, установку маршрута или изменение правила. Возможность получать информацию от системы маршрутизации, например данные RIB или текущую конфигурацию и состояние отдельных элементов, является не менее важной, особенно в контексте моделирования топологии сети.

Будущее SDN

На сегодняшний день SDN – это архитектурная концепция и маркетинговый термин, объединяющий множество закрытых решений построения и управления сетью. Как архитектура, SDN привлекает своей концептуальной простотой, открывая громадный потенциал для построения более сложных систем на ее основе. Также, декомпозиция управляющих и передающих элементов сети открывает новые горизонты для инновации. В этом смысле стандартизация всех интерфейсов SDN и, как следствие, создание открытой архитектуры, когда потребитель сможет самостоятельно комбинировать продукты различных производителей: коммутаторы, контроллеры и программное обеспечение управления сетью, может способствовать стремительному развитию этой технологии.

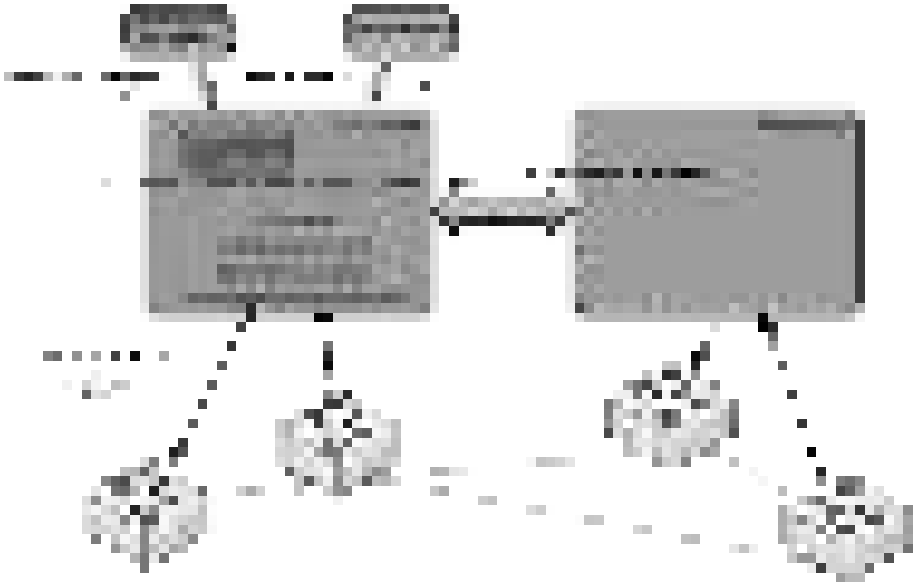

Рисунок 7. Интерфейсы SDN, требующие стандартизации

Сегодня стандарты – протоколы и модели данных - существуют только для т.н. «южного» интерфейса: между контроллером и коммутаторами. Стандартизация «северного» интерфейса обеспечит доступ на рынок для независимых разработчиков программного обеспечения. Определение стандартного интерфейса между контроллерами позволит различным сетям взаимодействовать между собой, открывая возможности построения SDN, охватывающих несколько независимых провайдеров.

Рисунок 8. Организации, участвующие в исследованиях и разработке стандартов, связанных с SDN.
(Источник: Доклад David Ward, Cisco, на пленарном заседани IETF84, июль 2012,
http://www.ietf.org/proceedings/84/slides/slides-84-iab-techplenary-3.pdf)

Достаточно посмотреть на число организаций, участвующих в исследованиях и разработке стандартов, связанных с SDN (см. Рис. 8), чтобы понять – работы здесь непочатый край. Этот рисунок также свидетельство тому, насколько сильна потребность в стандартных компонентах, строительных блоках SDN, способных взаимодействовать между собой. Работы ведутся во многих направлениях и возможно недалек тот день, когда SDN из архитектурной концепции и маркетингового термина превратится в стандартные протоколы и технологии.

И кто знает, может быть Интернет действительно станет программируемым.

Андрей Робачевский

Мнения, представленные в статье, не обязательно отражают официальную позицию Internet Society