Намиот Д.Е.[1]

[1] Московский Государственный Университет имени М.В. Ломоносова, г. Москва, к.ф.-м.н., старший научный сотрудник факультета ВМК, dnamiot@gmail.com

ИНТЕРФЕЙСЫ ПРИКЛАДНОГО УРОВНЯ В SDN

КЛЮЧЕВЫЕ СЛОВА

Software Defined Networks, API, REST, Parlay.

АННОТАЦИЯ

В этой статье мы обсуждаем интерфейсы прикладного уровня для Software Defined Networks. В то время как программные интерфейсы для взаимодействия с сетевой аппаратурой широко обсуждаются в литературе, интерфейсы для прикладных программ получают меньше внимания. Вместе с тем, вполне очевидно, что именно интерфейсы прикладного уровня очень важны. В настоящей работе мы хотим остановиться именно на этом аспекте Software Defined Networks и сравнить предлагаемые подходы с решениями для интерфейсов прикладного уровня в телекоммуникационных приложениях.

Введение

Концепция Software Defined Networks (SDN), по определению, основана именно на разнообразных программных интерфейсах. В целом, SDN контроллер есть именно набор программных интерфейсов. На рисунке 1 представлена классическая модель SDN.

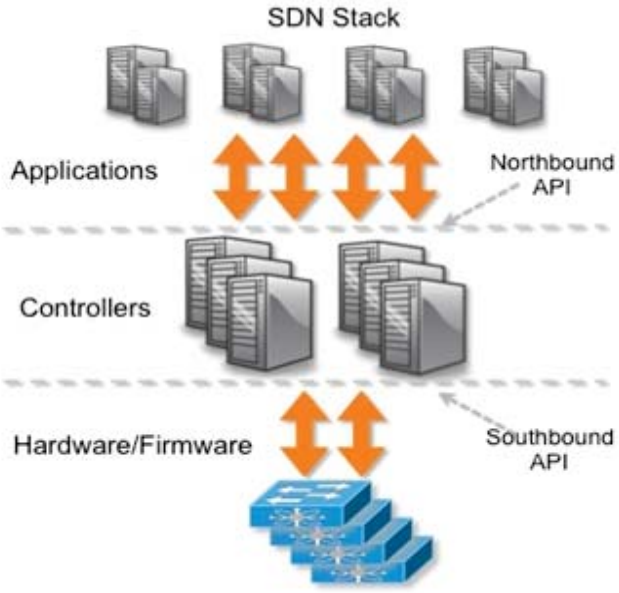

Рис. 1. SDN APIs [1]

В этой работе мы хотим остановиться на элементе, который называется северный API. Northbound API (Application Program Interfaces) предназначен для создания экосистемы приложений. Подобная экосистема и есть основная (по крайней мере, одна из основных) цель SDN. Пользователями данного API являются все разработчики сетевых приложений.

Естественно, имя в виду API, мы говорим именно о едином интерфейсе, который используется всеми разработчиками. В действительности же, насчитывается много различных предложений для интерфейсов прикладного уровня. В работе [2] упоминается более 20 проектов различных SDN контроллеров, каждый из которых вводит свой набор интерфейсов для прикладных программ. Изучением этих интерфейсов занимается консорциум Open Networking Foundation (ONF) [3].

Одной из причин, обуславливающей такие различия, являются разные требования, которые выдвигаются приложениями. Одна из популярных точек зрения состоит в том, что общие API должны быть предложены рынком, а не каким-либо комитетом [4].

API для приложений

Возможно, наиболее подходящим примером для программных интерфейсов прикладных приложений является Parlay. Parlay (Parlay X) представлял собой попытку представить общий API для телекоммуникационных приложений [5]. Основная идея Parlay состояла в объединении сетевого (сете-центрического) подхода в распространении телекоммуникационных услуг (то, за что отвечают интеллектуальные сети) и механизмов представления сервисов на уровне систем управления предприятиями. Parlay API позволяет скрыть сложные механизмы управления в телекоммуникационных сетях, сохраняя в то же самое время уровни безопасности, поддерживаемые телекоммуникационными операторами (рис. 2).

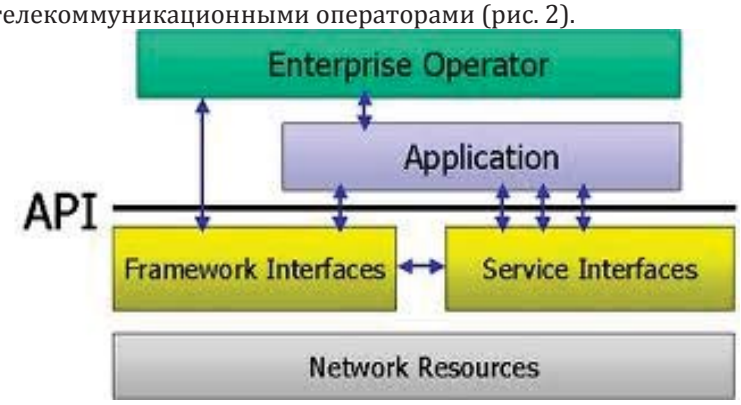

Рис. 2 The Parlay API [6]

Базовые принципы API весьма прозрачны и очевидны:

1) Parlay API посвящен программным интерфейсам, а не сетевым протоколам.
2) Parlay API не зависим от сети.
3) API обеспечивает безопасность на всех уровнях.
4) Простота применения.
5) Расширяемость и возможность пошагового внедрения (использования).

На первый взгляд, это выглядит весьма подходящим для концепции SDN. Но что можно сказать, если мы предметно рассмотрим компоненты Parlay API [7]:

1) Call Control API. Охватывает процесс установки соединения

2) User Interaction API. Отправка SMS, рингтонов и так далее. Чисто телекоммуникационные сервисы.

3) Terminal Capabilities API. Опрос характеристик пользовательских устройств.

4) Connectivity Management API. Это API для Quality of Services (QoS).

5) User Status API. Опрос состояния мобильного терминала.

6) Data Session Control и Account management. Это то, что касается биллинга.

Вывод полностью очевидный. Parlay API описывает, по факту, прикладные телекоммуникационные сервисы. А в случае SDN, мы имеем дело не с сервисами, а с другими приложениями. Название API для приложений нужно понимать здесь именно буквально – API для управления другими приложениями. Для SDN нет такого уровня практической ориентированности, как в случае Parlay API.

Виртуализация функций

ONF занимается также и развитием направления Network Functions Virtualization (NFV). Цель – предложить общие программные интерфейсы для построения виртуальных устройств [8]. Это позволит ускорить процесс создания новых сетевых сервисов. Обычно, операторы используют много типов различного сетевого оборудования. Естественно, что это увеличивает стоимость запуска новых сервисов. NFV призвана создать стандартную технологию виртуализации для консолидации сетевого оборудования в дата-центрах. Иными словами, это концепция сетевой архитектуры, предлагающая использовать технологии виртуализации для виртуализации целых классов функций сетевых узлов в виде составных элементов, которые могут быть соединены вместе или связаны в цепочку для создания телекоммуникационных услуг (сервисов) – рис.3.

Можно сказать, что NFV дополняет SDN, но не заменяет его. В части общих интерфейсов,

SDN Application API, в идеале, должно выступать в качестве программного интерфейса для SDN. Типичные области применения следующие:

- Виртуальные маршрутизаторы и VPN шлюзы;
- Виртуальные сетевые устройства (файрволл, например);
- Виртуальные сетевые сервисы. Например, средства мониторинга;
- Виртуальные приложения. Например, хранилища данных.

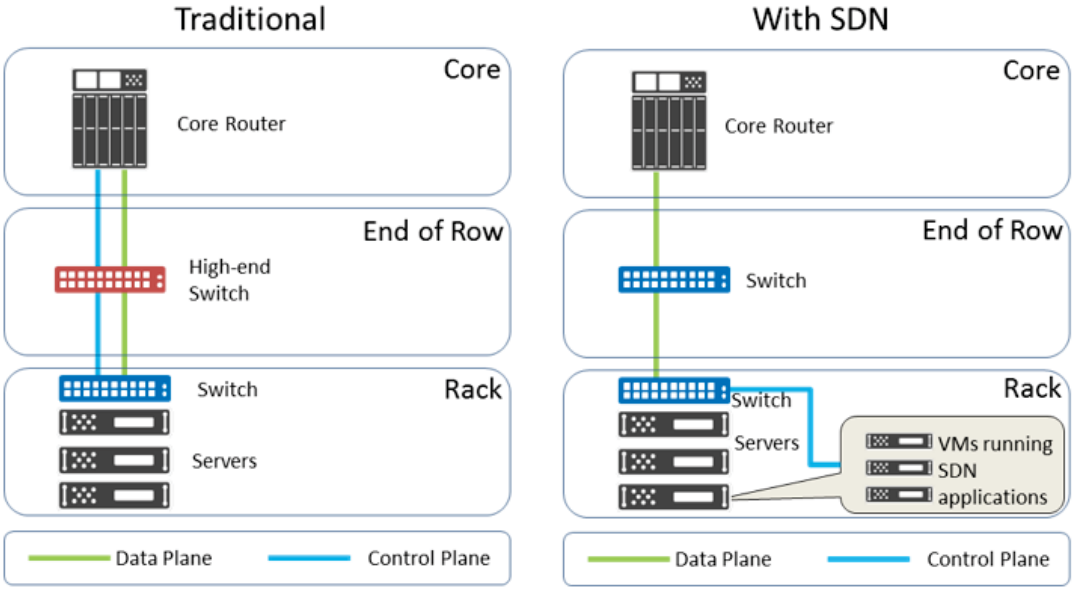

Рис. 3 NFV

Отметим, что понятие “приложение” здесь также весьма отличается от приложений (сервисов) в Parlay.

RESTCONF

Концепция программных интерфейсов в SDN также меняется со временем [11]. Вместо терминов “южный” и “северный” интерфейсы, ONF теперь говорит об интерфейсах данных и приложений [12]. Другая новая модель программных интерфейсов базируется на RSTCONF [13]. В этой связи используется несколько новых терминов: NETCONF, YANG, RESTCONF.

Network Configuration Protocol (NETCONF) [14] предоставляет механизм для управления конфигурацией сетевых устройств. Данные представляются в формате XML. Сами операции загрузки (модификации) конфигураций реализованы как удаленные вызовы (RPCs).

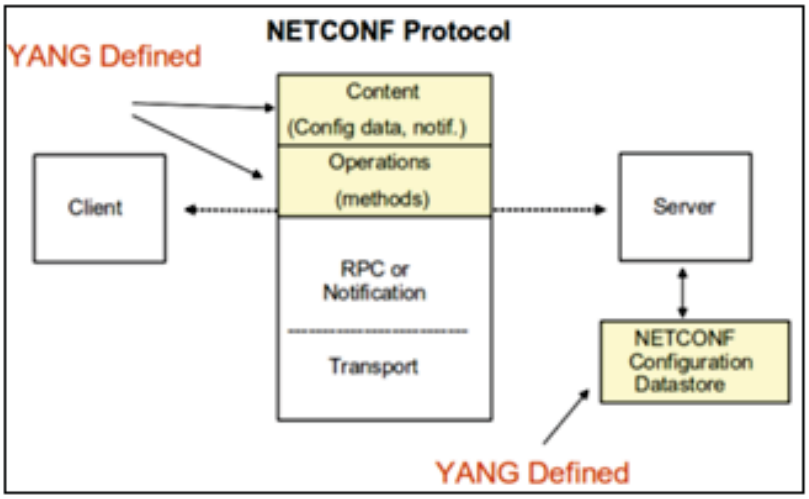

Рис. 4 Уровни NETCONF [15]

YANG представляет собой язык моделирования данных для NETCONF [16]. Модули YANG определяют иерархии данных для NETCONF операций. Модели в YANG представляются в виде

дерева. Например:

```
list interface {
  key "interface-name";
  leaf interface-name {
    type string;
  }
  leaf speed {
    type string;
  }
  leaf duplex {
    type string;
  }
}
```

А типичный файл NETCONF выглядит так:

```
<interface>
  <interface-name>TenGigabitEthernet 1/0/1</login-name>
  <speed>10Gbps</speed>
  <duplex>full</duplex>
</user>
<interface>
  <interface-name>TenGigabitEthernet 1/0/2</login-name>
  <speed>10Gbps</speed>
  <duplex>full</duplex>
</user>
```

На рисунке 5 показана связь NETCONF и YANG:

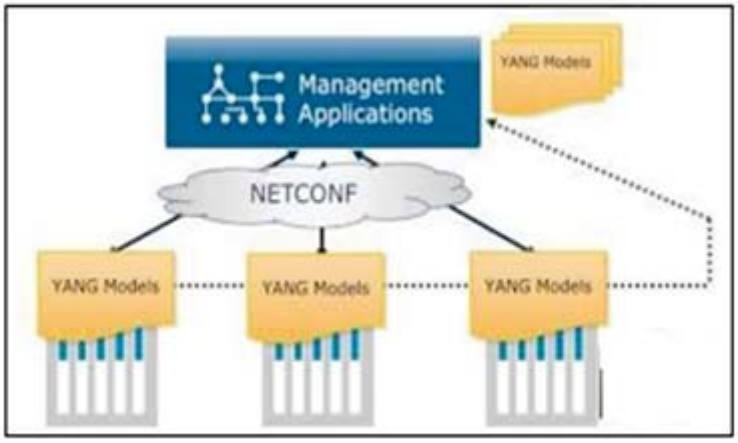

Рис. 5 NETCONF и YANG

RESTCONF описывает основанный на модели REST протокол, который обеспечивает доступ поверх HTTP к данным в YANG, используя хранилище данных в NETCONF [17].

Для SDN это означает появление новой программной модели (модели API), основанной на RESTCONF, NETCONF и YANG.

Пример такой системы приведен на рисунке 6.

В целом, REST-модели от OpenDayLight представляются весь перспективными в плане создания общего API. Схема их построения соответствует общей модели развития программных интерфейсов в межмашинных взаимодействия (M2M) [19] и Internet of Things (Iot) [20].

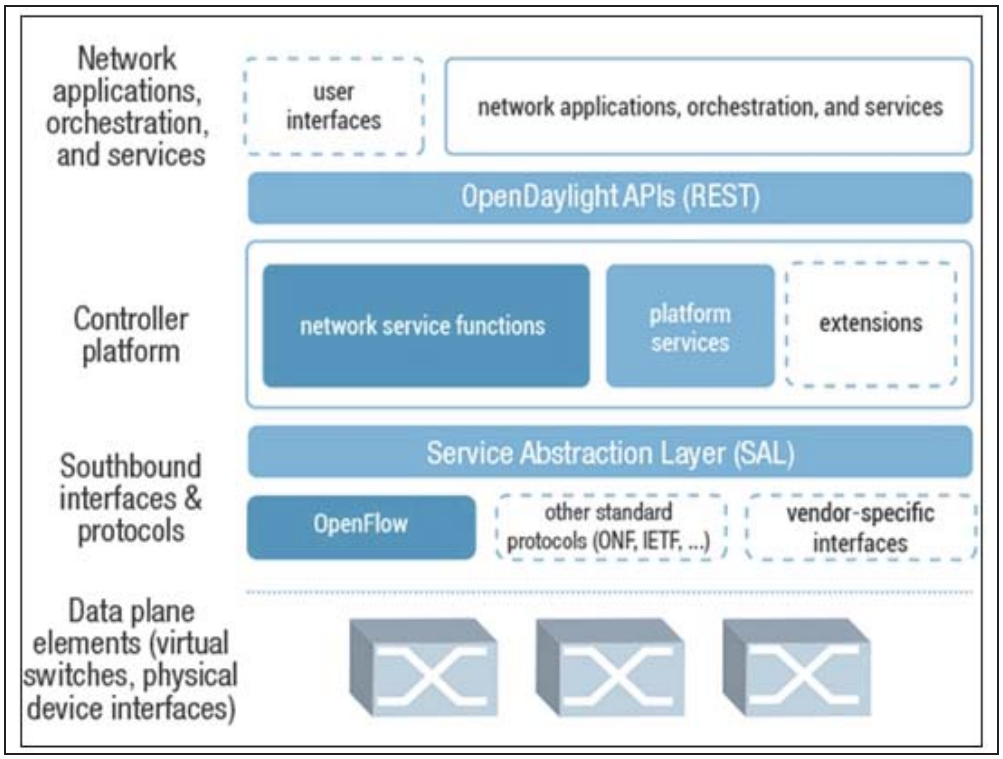

Рис. 6 OpenDayLight модель [18]

Литература

1. The Northbound API- A Big Little Problem http://networkstatic.net/the-northbound-api-2/ Retrieved: Jun, 2015.
2. Sezer, Sakir, et al. "Are we ready for SDN? Implementation challenges for software-defined networks." Communications Magazine, IEEE 51.7 (2013): 36-43.
3. Ortiz, Sixto. "Software-defined networking: On the verge of a breakthrough?." IEEE Computer 46.7 (2013): 10-12.
4. Do SDN northbound APIs need standards? http://searchnetworking.techtarget.com/feature/Do-SDN-northbound-APIs-need-standards Retrieved: Jun, 2015
5. Yates, M. J., and I. Boyd. "The Parlay network API specification." BT Technology Journal 25.3-4 (2007): 205-211.
6. Uve Herzog "OSA & Parlay. Enabling an open services market" http://archive.eurescom.eu/message/messageJun2002/tutorial.asp Retrieved: Jun, 2015
7. Sneps-Sneppe, M., & Namiot, D. (2012, April). About M2M standards and their possible extensions. In Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on (pp. 187-193). IEEE.
8. Schneider, Fabian, et al. "Standardizations of SDN and ITs practical implementation." NEC Technical Journal 8.2 (2014): 16-20.
9. NFV White Paper http://portal.etsi.org/nfv/nfv_white_paper.pdf Retrieved: Jun, 2015
10. Nunes, Bruno, et al. "A survey of software-defined networking: Past, present, and future of programmable networks." Communications Surveys & Tutorials, IEEE 16.3 (2014): 1617-1634.
11. Medved, J., Varga, R., Tkacik, A., & Gray, K. (2014, June). Opendaylight: Towards a model-driven sdn controller architecture. In 2014 IEEE 15th International Symposium on (pp. 1-6). IEEE.
12. Namiot, D., & Sneps-Sneppe, M. (2014, June). On software standards for smart cities: API or DPI. In ITU Kaleidoscope Academic Conference: Living in a converged world-Impossible without standards?, Proceedings of the 2014 (pp. 169-174). IEEE.
13. Bierman, A., Bjorklund, M., Watsen, K., & Fernando, R. (2014). RESTCONF protocol. IETF draft, work in progress.
14. Enns, R., Bjorklund, M., Schoenwaelder, J., & Bierman, A. (2011). Network configuration protocol (NETCONF) (No. RFC 6241).
15. NETCONF layers http://itential.com/tech-briefs/ Retrieved: Jun, 2015
16. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) https://tools.ietf.org/html/rfc6020
17. RESTCONF Protocol http://tools.ietf.org/html/draft-bierman-netconf-restconf-04
18. Baucke, S., Mestery, K., Shaikh, A., & Wright, C. (2013). OpenDaylight: An Open Source SDN for your OpenStack Cloud. An Open-Stack Summit, Hong Kong.
19. Namiot D., Sneps-Sneppe M. On M2M Software //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 6. – С. 29-36.
20. Namiot D., Sneps-Sneppe M. On IoT Programming //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 10. – С. 25-28.