Contemporary Engineering Sciences, Vol. 7, 2014, no. 32, 1747 - 1753
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.411234

A Comparative Study on Control Models of Software-Defined Networking (SDN)

Habinshuti Francois Xavier and Soonuk Seol

School of Electrical, Electronics and Communication Engineering
Korea University of Technology and Education (KOREATECH)
Cheonan, Chungnam, Republic of Korea

Abstract

Software-Defined Networking (SDN) is fast emerging as a significant building block for next-generation carrier services and networks. In SDN, the control function and data plane elements are separated to provide high flexibility. However, as the network size becomes larger, SDN's controllers cannot handle all requests. So, the controllers can potentially become the bottlenecks in the SDN applications. Different control models have been proposed to handle the bottlenecks. This paper categorizes control models of SDN. We also compare SDN architecture with Telco's policy and charging control (PCC) architecture which SDN needs to consider for meeting operator and subscriber needs.

Keywords: Software-Defined Networking, SDN, Policy and Charging Control

1 Introduction

In computer networking, network administrators need to configure a wide range of network switches and routers in order to handle different network events and applications requirements. Software-Defined Networking (SDN) has been introduced with a key idea of separating control function out of underlying system that forward data traffic. SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services. This decoupling enables flexible, manageable, and adaptable network

architectures for the high-bandwidth, dynamic nature of today's applications [1]. However, a common concern of SDN is on performance and scalability. Due to the separation, a centralized controller can potentially become a bottleneck, dealing with every request from growing network devices. There have been a number of literatures on SDN architecture and the control models. In this paper, we compare different control models with respect to performance and scalability. The rest of this paper is organized as follows. Section 2 describes architecture of SDN and introduces controller implementations. Different control models are categorized in Section 3. Section 4 compares SDN architecture with policy control architecture of Telco networks. Finally, we conclude our paper in Section 5.

2 Overview of Software-Defined Networking

Virtualization is one of good explanations for the basis of SDN. It allows software to run independently from the underlying network devices, providing a global view on the network infrastructure. In other words, SDN separates the network's control and forwarding planes to make it easier to optimize each.

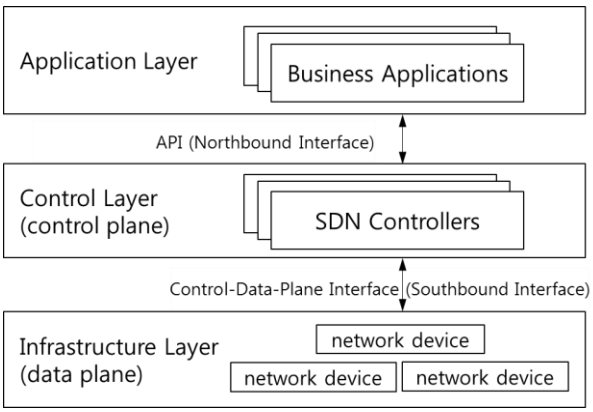


Figure 1. SDN Architecture

Figure 1 depicts a three-layer model of SDN architecture. A controller plays an important role like a brain and provides an abstract and centralized view of the overall network. Network operators can quickly and easily decide and configure how the network devices of the forwarding plane will handle the traffic. Currently OpenFlow protocol is commonly used for the southbound interface between controllers and the network devices. Application Programming Interface (API) shown in northbound of control layer is used to support all the services and applications. However, no standardized API or a high level programming language exists for SDN application developments, and in practice, the control software provides its own API to applications. We will compare SDN's reference model with other similar architecture of telecommunication networks in section 4.

At the lowest layer in the SDN reference model, infrastructure layer consists of switching devices such as switches and routers. Main roles of this data plane is data forwarding, and to monitor local information. In a conventional network, routers and switches handle both fast packet forwarding and high-level routing decisions on the same device. However, in SDN, the forwarding device, or OpenFlow switch separates these two functions and thus the high-level routing decisions are moved to a separate controller. An OpenFlow switch maintains one or more flow tables which tell how packets of a flow should be processed and forwarded. In general, the flow tables consist of match field, counters, and a set of instructions. The match field is a matching rule used to match incoming packets. The counters are used for gathering statistics for a flow such as the number of packets and bytes, and the duration of the flow. The instructions are the actions to be applied. When a switch receives a packet with no match in flow entries, it can drop the packet, continue matching with the next flow table, or forward the packet to the controller via OpenFlow protocol. Upon receiving such a packet, the controller makes a decision on how to handle this packet. It can drop the packet, or it can add it to a flow entry directing the switch on how to forward similar packets in the future.

With respect to SDN controllers, quite a lot implementations are available in different platforms including Python, C++, and Java, and from different organizations including Stanford, Rice University, Big Switch, NEC, NTT OSRG group, Linux Foundation and so on [2]. OpenFlow compliant controllers are summarized in [3] and most of them are available as an open source. In the following section, we describe different control models and their pros and cons.

3 Control Models

A simple control design is to move all control functionalities to a centralized controller, providing a whole network view to upper application layer. This will make application development much easier. However, as the network size becomes larger, SDN controllers cannot handle all requests. So, the controllers can potentially become the bottlenecks in the SDN applications. The early controllers NOX [4] and Ethane [5] showed 10~30k flow requests per second which is not suitable for carrier network environment. There have been a number of controllers and SDN designs proposed in order to improve the performance and scalability. Table 1 summarizes different approaches.

Better performance can be achieved by making the controller respond fast to data path requests and by reducing the number of requests. Such improvements can be realized either by redefining the protocol or by redesigning controller placement. In a reactive control model, when a packet of a new flow reaches a switch, it must be sent to a controller each time. DIFANE reduces the number requests to the controller by proactively pushes all state to the data path. Other approaches such as DevoFlow and CMQ selectively consult a controller by looking

at characteristic of flows. In DevoFlow, only larger flows are forwarded to the controller. CMQ asks any switch to send only one packet-in message during each RTT, for each source-destination pair, upon multiple flow table misses [9].

Placement of controllers can be categorized into centralized, physically distributed, hierarchically distributed, and logically distributed models. A physically centralized controller has a single point of failure issue. Logically centralized but physically distributed control plane can decrease the look-up overhead by enabling communication with local controllers, while inconsistency problem may occur due to distributed state.

Table 1. SDN's control models and approaches

Category	Approach	Idea
Parallelism and batching	NOX-MT [6]	Multi-threaded and I/O batch
Number of requests to the controller	DIFANE [7]	Proactively pushes all state to the data path
	DevoFlow [8]	Only long-lived, high-throughput flows are managed by the controller
	CMQ [9]	Flow aggregation
Placement of controllers	Onix [10]	Physically distributed with APIs to facilitate access to network state
	HyperFlow [11]	Physically distributed with a synchronized view
	Kandoo [12]	Hierarchical controllers
	Flowvisor [13]	Logically distributed (network virtualization)
Others	authority switches in DIFANE [7]	Packets are diverted through authority switches as needed to access appropriate rules
	Palette [14]	distribute small SDN tables to the whole SDN based on graph-theory
	BalanceFlow [15]	Load balancing at a super-controller

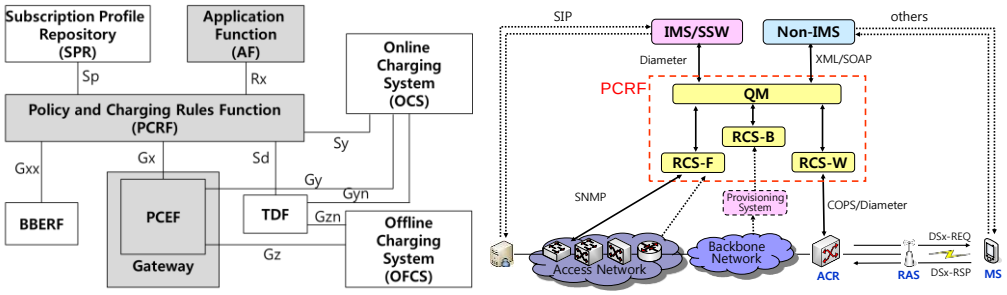
4 Architecture Comparison

In this section, we compare SDN architecture with similar architecture in Telco domain, i.e., Policy and Charging Control (PCC) architecture standardized by 3rd Generation Partnership Project (3GPP) which is shown in Figure 2 [16]. Both are based on a three-layer model. Also, like SDN architecture, PCC provides an API for third parties that operate their own application servers.

PCC works as follows. Policy and Charging Rules Function (PCRF) receives service information from application function (AF) through Rx interface shown in Figure 2(a). Service information mainly consists of flow information and QoS requirements. A typical example of AF is proxy call session control function (P-CSCF) of IP multimedia subsystem (IMS). PCRF, then, determines PCC rules for each service flow, and commands the Policy and Charging Enforcement Function

(PCEF) to enforce the rules through Gx interface. Both Rx and Gx interfaces are based on DIAMETER protocol. PCRF controls the PCC rules for various networks including UTRAN, GERAN, WiMAX, and Wi-Fi. Unlike SDN, PCC has the following differences which SDN need to consider to support.

- Take subscriber's profile into account by communicating with SPR
- Northbound interface to application function is standardized (DIAMETER)
- More interfaces to network devices and charging system
- Policy control (QoS and charging) as primary controlling mechanism rather than routing logic control with flow tables
- IMS (VoIP) is one of typical business applications
- Policy push and pull mechanisms are supported between control and data planes
- Network access service is a key service controlled by PCRF with pull mechanism
- Inter-PCRF communication is defined for roaming scenarios



(a) (b)

Figure 2. 3GPP PCC Architecture (a) and KT's PCC implementation (b)

Despite 3GPP PCC's advantages, it does not address implementation specific issues of the logical entities such as load balancing, parallelism, performance, and so on. In [17], KT's PCC architecture was introduced which reveals a certain level of deployment consideration. As shown in Figure 2(b), PCRF consists of a global coordinator, QoS Manager (QM), which provides APIs to applications and makes policy decisions, and three types of Resource Control Subsystem (RCS) for the different access networks, implying logically and/or physically distributed placement. Both northbound and southbound interfaces support various protocols. In case of backbone networks, only monitoring was considered. In [18], a scalable typology update mechanism was used in order to provide a global topology view.

5 Conclusions

In this paper we have categorized and compared different control models of SDN with respect to performance and scalability. Common approaches include reducing the number of requests and placing controllers. We have also compared

SDN architecture with Telco's policy architecture. Both can reinforce each other by taking benefits and strong points discussed in our paper.

References

[1] Software-Defined Networking (SDN) Definition, https://www.opennetworking.org/sdn-resources/sdn-definition.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, H. Xie, A Survey on Software-Defined Networking, *IEEE Communications Surveys & Tutorials*, (2014). http://dx.doi.org/10.1109/comst.2014.2330903

[3] B. A. A. Nunes, M. Mendonca, Xuan-Nam Nguyen, K. Obraczka, T. Turletti, A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, *IEEE Communications Surveys & Tutorials*, vol.16, no.3, (2014), 1617-1634. http://dx.doi.org/10.1109/surv.2014.012214.00180

[4] NOX, http://noxrepo.org.

[5] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review*, 37(4), (2007), 1-12. http://dx.doi.org/10.1145/1282427.1282382

[6] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, On controller performance in Software-Defined Networks, *Proc. USENIX Hot-ICE*, (2012), 10 - 10.

[7] M. Yu, J. Rexford, M.J. Freedman, and J. Wang. Scalable flow-based networking with DIFANE, *Proc. ACM SIGCOMM 2010 Conf.*, (2010), 351-362. http://dx.doi.org/10.1145/1851182.1851224

[8] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, DevoFlow: scaling flow management for high-performance networks, *SIGCOMM Comput. Commun. Rev.*, vol. 41, no.4, (2011), 254-265. http://dx.doi.org/10.1145/2043164.2018466

[9] Tie Luo, Hwee-Pink Tan, P.C. Quan, Yee Wei Law, Jiong Jin, Enhancing responsiveness and scalability for OpenFlow networks vi control-message quenching, *International Conference on ICT Convergence*, (2012). 348-353. http://dx.doi.org/10.1109/ictc.2012.6386857

[10] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, Onix: A distributed control platform for large-scale production networks, *OSDI*, (2010).

[11] A. Tootoonchian and Y. Ganjali, Hyperflow: A distributed control plane for OpenFlow, *Proc. 2010 INM Conf.*, (2010), 3 - 3.

[12] S. Hassas Yeganeh and Y. Ganjali, Kandoo: A framework for efficient and scalable offloading of control applications, *Proc. HotSDN 2012 Wksp.*, (2012), 19- 24.

[13] J. Matias, B.Tornero, A. Mendiola, E. Jacob, and N. Toledo, Implementing Layer 2 Network Virtualization Using OpenFlow: Challenges and Solutions, *Proc. of European Workshop on Software Defined Networking*, (2012), 30 - 35. http://dx.doi.org/10.1109/ewsdn.2012.18

[14] Y. Kanizo, D. Hay, I. Keslssy, Palette: Distributing tables in software-defined networks, *Proc. of IEEE INFOCOM*, (2013), 545-549. http://dx.doi.org/10.1109/infcom.2013.6566832

[15] Yannan Hu, Wendong Wang, Xiangyang Gong, Xirong Que, Shiduan Cheng, BlanceFlow: Controller load balancing for OpenFlow networks, *IEEE Conf. on Cloud Computing and Intelligent Systems*, Vol. 2, (2012), 780-785. http://dx.doi.org/10.1109/ccis.2012.6664282

[16] 3GPP Rel. 12, TS 23.203: Policy and charging control architecture, (2014).

[17] Sungsoo Cho and Soonuk Seol, Providing QoS and rate limiting for WiMAX mobile hotspots based on Policy and Charging Control architecture, *International Journal of Control and Automation*, Vol.6, No.4, (2013).

[18] Soonuk Seol and Sungsoo Cho, A new scheme for network topology management in Policy and Charging Control architecture, *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, (2013), 529-530. http://dx.doi.org/10.1109/icufn.2013.6614876

Received: October 1, 2014; Published: December 2, 2014