Quick Survey of Benefits from Control Plane and Data Plane Separation in Software-Defined Networking

Pulkit Tanwar[1], Rajender Gohil[2] and Mudit Tanwar[3]

Submitted in November, 2015; Accepted in February, 2016

Abstract — Software-Defined Networking (SDN) involves programmable networks. It has recently gained popularity because of simplicity in networks, easy evolvement and providing innovation in network programmability. In this paper, we survey the SDN infrastructure and the OpenFlow standard, examine the need for Control Plane and Data Plane separation, and study various SDN controllers and SDN application in Data Centers.

Index Terms - Control Plane, Data Plane, Software-Defined Networks, Survey, Comparative study, Networking, Virtualization.

1.0 INTRODUCTION

Earlier Traditional networks were referred to as "Internet ossification" because it was highly coupled network [25]. There was very tight coupling between Control Plane and Data Plane. This tight coupling discourages the evolvement of networks. All the decisions of data flowing through the network were made on boards' network element.

Software-Defined Networking, SDN is the newest approach to Computer Networking that separates the data plane i.e. the network devices that forward traffic from the control plane i.e. the software logic. This separation of data plane and control plane allows the network operator to control the network behavior from a single high level control program. This technology is being applied in mobile open networks and mobile transaction systems [26] to efficiently handle mobility in the context of future 5th Generation of mobile networks [27].

The deployment of Software-Defined Networking is used to solve complex Network management problems in real networks. SDN started when distributed network configuration was found to be unpredictable, faulty in operation and buggy. [1] To overcome this, in 2004, Portal Gateway Protocol (PGP) [2] was introduced and the central control point was named as Routing Control Platform (RCP) [1]. In 2005, this architecture was generalized and named as 4D architecture [3]. This 4D architecture spotlight the separation of the routing decision logic and the protocols of the network. In 2008, OpenFlow standard [4] was introduced which standardized the information exchange between data plane and control plane. An industrial driven organization is formed called as Open Network Foundation [5] to promote Software-Defined Networking and OpenFlow standard protocol.

[1, 2, 3] Department of Computer Engineering,
Delhi Technological University, New Delhi, India
Email Id: [1]pulkit.tan45@gmail.com, [2]rajgohil04@gmail.com and [3]mudit1102@gmail.com

In this paper, we survey the Software-Defined Networking paradigm and study in detail the infrastructure and architecture of the same. We examine the need for control and data plane separation. We then see the overview of various SDN controllers. At the end, we study SDN application in Data Centers.

2.0 DECOUPLING OF DATA PLANE AND CONTROL PLANE

In Data Communication networks, end user devices are interconnected with network infrastructure. This network infrastructure includes many switching elements such as switches and routers and they are shared between hosts. These routers and Switches are closed devices which have very limited interfaces. So it is very cumbersome for this network to evolve.

Software-Defined Networking infrastructure consists of two parts:

Control Plane

Control Plane is the logic that controls the forwarding behavior in the network. It is also regarded as the brain of the network.

Examples:

- Routing Protocols
- Network middlebox configuration
  i. Firewall configuration
  ii. Load balancer configuration.

Data Plane

Data Plane forwards the traffic according to control plane logic.

Examples:

- IP forwarding
- Layer 2 switching.

The separation of data plane and control plane can be used to evolve and develop them independently. This separation also helps the network to be controlled from a single high level software program which makes debugging easier. Both software and hardware can be evolved independently.

Continual Challenges in separation of control and data plane includes:

- **Scalability:** Control element responsible for a large number of forwarding elements.
- **Reliability:** Controller may fail or compromise.
- **Consistency:** Ensuring consistency across multiple control replicas.

Various opportunities from Control and Data Plane separation include:

- Server load balancing [16]
- Virtual Machine Migration in Data Centers
- Network virtualization

- New routing services in the wide area
- Using multiple wireless access points
- Seamless mobility and migration
- Security in Enterprise Networks
- Denial-of-service attack detection
- Dynamic access control
- Adaptive traffic monitoring
- Energy-efficient networking

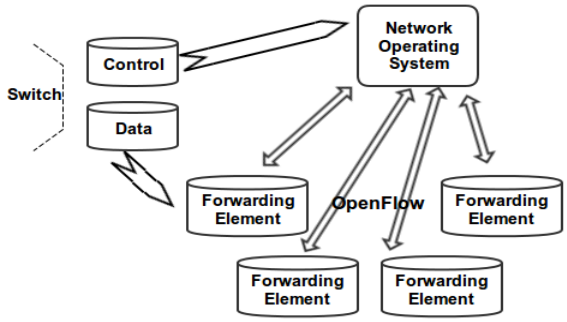

Figure 1. Separation of control plane and data plane. Switches only have forwarding elements.

3.0 CONTROL PLANE

3.1 OpenFlow Communication Protocol Specification

OpenFlow protocol [5] defines the message format and separates the data and control plane. OpenFlow controller communicates with "OpenFlow switch" over a secure channel and the protocol effectively instructs the "OpenFlow" switch to update its flow table entries.

3.2 OpenFlow Switch Components

- **Flow Table:** The flow table performs packet lookup. Flow tables consist of flow entries, which determine where the packet is to be forwarded. Flow entries consist of: (1) match fields, used to match incoming packets; (2) counters, used to count statistics like number of packets received, number of data bytes sent; and (3) a set of actions, to be applied upon a match. The lookup function compares the fields in each packet to a flow table that reside in the switch and looks for a match. The action of a switch depends on the match found. If no match is found, traffic is sent to the controller.
- **Secure Channel:** It is used for the communication of switch and the controller.

3.3 SDN Controller

Some of the important SDN Controllers include:

- **NOX [7] :** It was one of the widely used open-source, First Generation OpenFlow controller. In NOX, Users implement the control in C++ programming language. It supports OpenFlow v.1.0. In NOX, the programmer first writes a control program that registers for events and then writes event handlers that respond to those events.

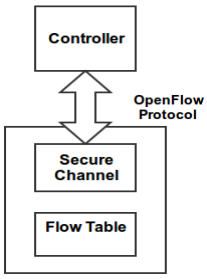

Figure 2. OpenFlow Switch Components consisting of flow table and secure channel. Control logic is moved to the controller.

- **POX [8] :** It is NOX implemented in python programming language. POX only supports OpenFlow v.1.0.
- **Ryu [9] :** It is an Open-Source python controller. It supports OpenFlow v. 1.0, 1.2, 1.3, and 1.4. Ryu also has OpenStack [17] integrations.
- **Floodlight [6] :** It is an Open-Source Java controller. It supports OpenFlow v. 1.0. Floodlight is maintained by Big Switch Networks. It was integration with REST API.
- **OpenDaylight [10] :** It is a common industry supported platform. OpenDaylight has OpenStack integrations.
- **LoxiGen [14] :** It is an existing controller library that generates OpenFlow language-specific bindings. If the input is "Wire-protocol descriptions" of a control protocol, the output would be "protocol specific bindings" in different languages.

Network Virtualization is the abstraction of the physical network that allows the support of multiple logical networks running on a common shared physical substrate. Flowvisor [18] is a proxy controller which can be employed to add network virtualization to OpenFlow networks. It permits diverse controllers to concurrently control the overlapping sets of switches.

Various aspects of the network that can be virtualized include:

- **Nodes:** Virtual Machines
- **Links:** Tunnels
- **Storage**

SDN separates the Control Plane and the Data Plane, whereas Virtual Networks separate logical and physical networks. Network virtualization provides: 1. Rapid innovation, services are delivered at software speeds. 2. New forms of network control, it makes possible to instantiate multiple logical networks on top of a single physical network. 3. Vendor Choice 4. Simplified programming and operation by allowing the network operator to see the network through a logical abstraction

Example: Virtual Machine Environment: Xen [20].

Xen hosts allow multiple guest operating systems that run on the same shared physical hardware. Domain0 runs control software in the XenoLinux environment.

4.0 PROGRAMMABLE DATA PLANE

In a conventional data plane, first the router receives the packet. It then examines the packet header for its destination. It looks for the forwarding table for the next hop output interface. It then modified the header and passes the packet to the appropriate output interface.

Data plane can easily be customized as it consists of streaming algorithms that act on packets.

The data plane can perform the following functions:

- Forwarding
- Shaping and Scheduling
- Deep Packet Inspection
- Traffic Monitoring
- Access control
- Mapping header fields
- Buffering and marking

Two open platforms being used include:

- **Click [19] :** Software data plane in user space or the kernel, Open, extensible and configurable router framework.
- **NetFPGA [11] :** Hardware data plane based on FPGAs

5.0 SDN DATA CENTER NETWORKS

One of the applications of Data Centers is mobile cloud computing [22]. Data centers have multiple tenants or independent users which allows the users of the data center to advertise the cost of the shared infrastructure. One of the key enabling technologies behind data centers is Visualization, the ability to run multiple virtual machines on one shared physical machine.

In conventional Data Center topology, the core of the data center is connected to the internet with layer-3 routers. The servers at the edge of the data center were connected to one another with layer-2 switches and the access layer was connected to the core with aggregation switches. This topology has numerous drawbacks like single point of failure, over subscription of links higher up in the topology.

A solution to this topology problem is to use an alternate topology [28] or to design the data center as a multi-rooted fat tree [12], where capacity increases towards roots of the tree. The PortLand [13] design introduces a fabric manager which in combination with positional pseudo MAC address, allows a layer-2 network to scale to tens to thousands of servers inside a data center.

In 2012, Google presented a SDN based network connecting its data centers at the Open Network Summit [15]. B4 [24], one of the largest SDN deployments, is a Wide Area Network connecting Google's Data Centers.

6.0 SDN NETWORK VERIFICATION

Verification of Software-Defined Networks is necessary for the proper functioning of the Networks without any faults. Proper configuration provides the flexibility for realizing operational goals and behavior of the network. For network verification, correctness specification and constraints for global internet routing are being employed. Router Configuration Checker (rcc) is a static configuration tool which is generally used for fault detection.

Data plane verification can be done by performing symbolic execution on packets. Veriflow [23], a layer between a software-defined networking controller and network devices check network-wide invariants in real time using data plane state. It monitors the dynamic changes in the network, constructs a model of the network behavior, and uses custom algorithms to automatically derive whether the network contains errors. Veriflow runs its analysis over a forwarding graph and produces a set of invariant violations and a set of packets that caused the invariant to be violated. Veriflow figures out which sets of packets belong to the same equivalence classes to reduce the number of tests that need to be run.

Kinetic, a domain specific language enables verifiable, event based network control. Network policies are represented as finite state machines, where each finite state machine maps to a pyretic [21] policy. Pyretic enables sequential composition of finite state machines.

7.0 CONCLUSION

In this paper, we surveyed the need to decouple the control and data plane in a network. The control Plane is the logic that controls the forwarding behavior in a network whereas the Data Plane forward traffic according to the control plane logic. We examined various SDN controllers. Further, we studied the OpenFlow communication protocol. The application of Software-Defined Networking in Data Centers is also studied. We also saw the challenges in Control and Data Plane separation. Software-Defined Networking has many applications in Optical Networks, Wireless Access Networks and Enterprise Networks.

REFERENCES

[1]. Feamster et al. "The Case for Separating Routing from Routers." Proc. SIGCOMM FDNA, 2004
[2]. Caesar et al. Design and implementation of a Routing Control Platform. Proc NSDI, 2005
[3]. A. Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4d approach to network control and management. ACM SIGCOMM Computer Communication Review, 35(5):41–54, 2005.
[4]. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69–74, 2008.
[5]. Open networking foundation. https://www.opennetworking.org/about.
[6]. Floodlight, an open sdn controller. http://floodlight.openflowhub.org/.
[7]. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3):105–110, 2008.
[8]. POX. http://www.noxrepo.org/pox/about-pox/.
[9]. Ryu. http://osrg.github.com/ryu/.
[10]. Opendaylight, 2013 http://www.opendaylight.org/.
[11]. NetFPGA platform. http://netfpga.org.

[12]. M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center 2 network architecture, in: ACM SIGCOMM 2008 Conference on Data 3 Communication, Seattle,WA, 2008, pp. 63–74
[13]. Niranjan Mysore, Radhika et al. "PortLand: a scalable fault-tolerant layer 2 data center network fabric." SIGCOMM '09 Proceedings of the ACM SIGCOMM 2009 conference on Data communication Pages 39-50
[14]. LOxiGen https://github.com/floodlight/loxigen
[15]. Inter-datacenter wan with centralized to using sdn and openflow. In Open Networking Summit, April 2012.
[16]. Sulata Mitra, Arkadeep Goswami, "Load Balancing in Integrated MANET, WLAN and Cellular Network", BIJIT - BVICAM's International Journal of Information Technology, Vol.3 No.1, January – June, 2011.
[17]. http://www.openstack.org/. Openstack, 2013.
[18]. R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.Y. Huang, P. Kazemian, M. Kobayashi, J. Naous, et al. Carving research slices out of your production networks with openflow. ACM SIGCOMM Computer Communication Review, 40(1):129–130, 2010.
[19]. http://read.cs.ucla.edu/click/. Click
[20]. Barham, Paul, et al. "Xen and the art of virtualizaLon." ACM SIGOPS Opera.ng Systems Review 37.5 (2003): 164 -177.
[21]. J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN Programming with Pyretic," USENIX; login, vol. 38, no. 5, October 2013.
[22]. Rana, "Innovative Use of Cloud Computing in Smart Phone Technology", BIJIT - BVICAM's International Journal of Information Technology, Vol.5 No.2, July-December, 2013.
[23]. Ahmed Khurshid , Wenxuan Zhou , Matthew Caesar , P. Brighten Godfrey, VeriFlow: verifying network-wide invariants in real time, Proceedings of the first workshop on Hot topics in software defined networks, August 13-13, 2012, Helsinki, Finland
[24]. Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pages 3–14. ACM, 2013
[25]. K. Bhatia, A. K. Pal, Anu Chaudhary, "Performance Analysis of High Speed Data Networks Using Priority Discipline", BIJIT - BVICAM's International Journal of Information Technology, Vol.1 No.2, July – December, 2009.
[26]. R. B. Patel, Anu, "A Mobile Transaction System for Open Networks", BIJIT - BVICAM's International Journal of Information Technology, Vol.1 No.1, January – June, 2009.
[27]. Meneses, F.; Corujo, D.; Guimaraes, C.; Aguiar, R.L. "Multiple Flow in Extended SDN Wireless Mobility", Software Defined Networks (EWSDN), 2015 Fourth European Workshop on, On page(s): 1 – 6
[28]. Shafi Patel, Parag Parandkar et al., "Exploring Alternative Topologies for Network-on-Chip Architectures", BIJIT - BVICAM's International Journal of Information Technology, Vol.3 No.2, July – December, 2011.