On Software-defined networking and the design of SDN Controllers

Doan B. Hoang
iNEXT: Centre for Innovation in IT Services and Applications
University of Technology, Sydney
Sydney, Australia
Doan.Hoang@uts.edu.au

Minh Pham
iNEXT: Centre for Innovation in IT Services and Applications
University of Technology, Sydney
Sydney, Australia
Minh.Pham@student.uts.edu.au

Abstract— **Software-Defined Networking (SDN) has emerged as a networking paradigm that can remove the limitations of current network infrastructures by separating the control plane from the data forwarding plane. The implications include: the underlying network state and decision making capability are centralized; programmability is provided on the control plane; the operation at the forwarding plane is simplified; and the underlying network infrastructure is abstracted and presented to the applications. This paper discusses and exposes the details of the design of a common SDN controller based on our study of many controllers. The emphasis is on interfaces as they are essential for evolving the scope of SDN in supporting applications with different network resources requirements. In particular, the paper review and compare the design of the three controllers: Beacon, OpenDaylight, and Open Networking Operation System.**

Keywords— Software-Defined Networks; SDN Controller; Northbound Interface; Southbound Interface; East/Westbound Interface

I. Introduction

The Internet has scaled enormously over the last 45 years partly due to its appropriate and universal level of abstraction at the IP layer (network layer), supporting any type of underlying physical networks and vast number of applications. However, the Internet has reached a point where it is extremely difficult to explore new architectures that are more suitable for emerging applications. After various efforts in finding alternative design, Software-Defined Networking (SDN) [1] has emerged as a networking paradigm that can remove the limitations of current network infrastructures. The intent of SDN is to separate the data forwarding plane from the control plane by centralizing the network state and the decision making capability, such that:

1. The network device becomes much simpler as it does not have to deal with complicated and distributed information and decision making,
2. The controller can be implemented in software by any vendors/providers provided it can communicate with network devices through the communication channel,
3. Programmability. The controller understands the needs of the applications through its north bound interface and resources available of the underlying network infrastructure through its south bound interface [2].
4. Openness. It is important to have all the application program interfaces open to gain a maximum benefits from the new paradigm in terms of applications, service providers, controller providers, network providers

A few recent papers have surveyed specific architectural aspects of SDN [3], [4]. However, none of them focuses on the implication and evolution of SDN or on the SDN controllers, their internal architecture and the trends in their future development. In this paper the generic SDN architecture is described in section II, SDN controller architecture in section III, and their structure and core components are examined in section IV. Section V concludes the paper with a discussion.

II. SDN Architecture

The Open Networking Foundation, ONF [5] defines a high-level architecture for SDN with three main layers:

A. Infrastructure Layer or data plane

It consists mainly of SDN devices (both physical and virtual) that perform packet switching and forwarding. Specifically an SDN device is composed of an application program interface API (the south bound interface) for communication with the controller, an abstraction layer that abstract the device as flow tables, and a packet-processing component decides where the packets will go based on the matching of their header fields.

B. Control Layer or control plane

It consists of a set of controllers that interact with applications and devices through three application programming interfaces: the SDN controllers control, through a southbound API, all the SDN devices that make up the network infrastructure, they provide an abstract view of the entire network to the applications through a northbound API and the controllers interact with one another through their east/west bound APIs to provide a consistent view of the entire network infrastructure.

C. Application Layer

It consists of the end-user applications that utilize the SDN communications and network services [6] such as network security, quality of service, traffic engineering, access control management, and load balancing as a service and other network function virtualization services [7]. The applications ultimately affect the flows on the SDN devices by communicating their requirements through the northbound API.

III. SDN CONTROLLER

To an SDN device, the controller is like a device driver software that initiates and configures the device to some specific operational settings by reading from and writing to the device's memory. To the underlying network, the controller is a piece of software that manages and shares all the resources of the underlying network infrastructure among applications.

Figure 1 depicts main components of an SDN controller including its core functional modules, its interfaces and a set of common application modules.

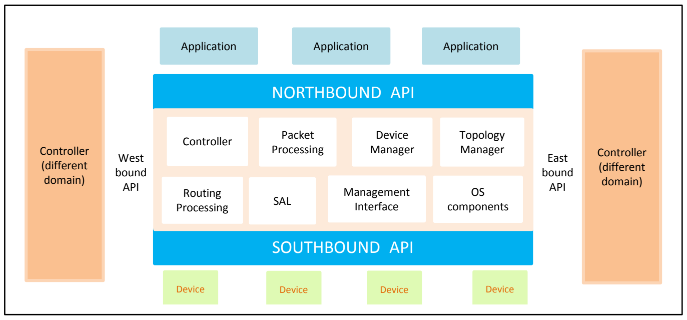

Figure 1: SDN Core components

A. Core Component Functions

The core network services include the components as described in the following [3], [8]:

1) Controller and Switch Management Unit

The controller is modeled as a container with all switches connecting to it. The controller registers to switch listeners and to message listeners to be notified about changes in switches and arrival of messages. The switch is modeled with its input and output message stream and its state.

2) Packet Processing Unit

Packet processing is the design element that processes packet headers and its payloads in the network based on the network protocol; each packet is modeled with its source addresses, destination addresses, its message type, and its payload.

3) Device Manager

Device Manager manages devices in the network; each device is modeled with a data link layer address, network addresses, the switch and switch port that it connects to.

4) Topology Manager

The topology manager identifies topology changes in the network. The topology manager maintains an up-to-date topology of the network and sends the topology updates to applications.

5) Routing

Depending on the protocol the routing manager implements the routes between the source and destination addresses provided.

6) OpenFlow Implementation

OpenFlow module exists in every controller to provide functions related OpenFlow messages, actions, table entry, and flow rules, matching of flow rules, message queues and statistics.

7) Service Abstraction Layer and Plug-ins

The service abstraction layer and plug-ins are used to add new device protocols to the controller.

8) Management Interface

Management Interfaces are usually provided as web interfaces to access the controller. These interfaces provide a quick and visible way to access functions that the controller provides.

IV. EXISTING SDN CONTROLLER IMPLEMENTATION

Available on the market today both open source SDN controllers and commercial SDN controllers. The selection of the SDN controllers is based on the history of SDN development, from the time controller proof of concept is required; to the popular controllers in today network community. These controllers are Beacon, OpenDaylight and ONOS.

A. Beacon Controller

BEACON was created in 2010 at Stanford University as research project; it is a java-based open source controller. Beacon is designed with three main objectives: performance, ease of applications development, and runtime configuration. Beacon is built on popular frameworks OSGi, Spring framework and Spring web. Beacon includes OpenFlowj, which is the implementation of the OpenFlow 1.0 specification. Beacon uses the observer pattern to listen to changes in the switches, message arrival and network devices. Beacon uses multithreaded to improve performance, especially when reading messages from switches and writing messages to them [8].

B. OpenDaylight (ODL) Controller

OpenDaylight controller is written in Java and it is an open source project. The ODL project was inspired by the Beacon controller in using OSGi to add more plug-ins. ODL is built as a distributed controller to support the High Availability (HA) with the cluster architecture: all controllers in the same cluster will act as one controller [9]. ODL uses the Service Abstraction Layer (SAL) to add additional device protocols. Besides OpenFlow ODL supports other protocols NETCONF, RESTCONF, Yang, BGP [9]. The NBI is provided as REST API, REST-CONF and Java API. Security is added to its Lithium release in June 2015. ODL has been used in different projects/products such as it is extended and used in Software-Defined Environment (SDE) product at IBM [10].

C. Open Network System (ONOS)

ONOS controller is an open source Java controller designed or telecommunication service provider networks. The expected performance is set very high such as 500k-1M path setups/sec or 500M-1T ops/sec. The service core layer is constructed by sub components: device, host, links, graph, and event. ONOS distributed core layer handles the synchronization between controllers via replication. Core model objects are protocol-agnostic and are used as network and state representation. ONOS represents networks as directed graphs. Users can access ONOS controller via command line interface (CLI), web GUI views, and REST API [11]. Its NBI support two formats: prescribe or fine-grained, in which users dictate what the need exactly and intent-based, in which users only specify the requirement and NBI is responsible for building the functions.

D. Other Controllers

There are other SDN controllers that are widely used in the academia and industry [3]:

- NOX is the first openflow controller created by the University of Berkeley; it is written in C++. There are two NOX versions: the original and the multi-threaded
- POX is the Python version of NOX.
- Floodlight is created by Big Switch Network for switches manufactured by the company, and it is based on Beacon controller.
- Ryu is originated from NTT in Japan. It is built based on components with the main purpose to make it easy to build network applications based on it.
- OpenContrail is open source controller, it is written in C++ with a REST NBI, and it offers integration plug-ins for cloud services: Amazon, Openstack, and Cloudstack.
- DISCO is a distributed controller, it is written in Java, and it has REST NBI and it is fault tolerance.
- ElastiCon is a distributed controller, it is written in Java, it has RESTful NBI and it offers consistent update nodes.

TABLE 1. CONTROLLERS' ATTRIBUTE COMPARISON

Attributes	**Beacon**	OpenDaylight	ONOS
Objective	Performance, Ease of applications development, Runtime configuration	Modular, Pluggable, Flexible controller	SDN OS for network providers. Modular, Scalability, High availability, Performance,
Architecture	Centralized OGSI-based	Distributed, Cluster-based	Distributed, Three tiers
Core Network Services	Core network services	Controller platform	Core services and Distributed core layer
Southbound API/protocols	OpenFlowj, Device manager	OpenFlow, ovsdb, netconf, pcep,snmp,bgp	OpenFlow, NETCONF, PCEP
Northbound API	Ad-hoc API	REST API	Application Intents
East/Westbound API	Not yet provided	Not yet provided	Not yet provided
Service Abstraction Layer / Plug-ins	OSGI plug-ins	Support additional protocols	Southbound API
Management Interfaces	Web UI	Java	Java
Programming Language	Java	Java	Java

V. DISCUSSION AND CONCLUSIONS

Software-defined networking has grown beyond its initial intention of managing network devices. It is more about supporting applications, allowing them to *program* its requirements on to the underlying network infrastructure through the control plane. The controller is capable of providing mechanisms (hypervisors) for *virtualizing network* resources, especially the intent-based approach [12] and allocating virtual networks to relevant applications or tenants. Northbound interface(s) is (are) being developed, taking into account disparage application requirements. Logically centralized control is still crucial for device simplification and management automation. The East/Westbound APIs are being defined to provide interoperability between different controllers, or synchronize the state of network devices in a federated underlying network. The southbound interface is extended to support other APIs, plug-in protocols, and services other than OpenFlow for complete management of network devices, such as Domain name service (DNS), Dynamic host configuration protocol (DHCP), OVSDB, NETCONF. The challenge is adopt a SDN controller that facilitates application innovation and opens up wider markets for vendors and service providers.
As SDN gains more and more adoption, the requirements for SDN controllers are to close the gaps so SDN controllers can fulfill their roles in production systems of large datacenter networks and WAN networks: high availability and security. To satisfy the new requirements new controllers and releases of existing controllers are being built with these objectives: scalability, high availability, and security.

REFERENCES

[1] N. Mckeown, "How SDN will shape networking", https://www.youtube.com/watch?v=c9-K5_qYgA, accessed 30 Aug 2015

[2] ONF, "Software-Defined Networking: The new norm for networks" , https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, accessed 30 Aug 2015

[3] Kreutz, D., Ramos, F.M.V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S., 'Software-Defined Networking: A Comprehensive Survey', Proceedings of the IEEE, 2015, 103, (1), pp. 14-76

[4] Jarraya, Y., Madi, T., and Debbabi, M.: 'A Survey and a Layered Taxonomy of Software-Defined Networking', Communications Surveys & Tutorials, IEEE, 2014, 16, (4), pp. 1955-1980

[5] ONF Solution Brief: 'SDN security considerations in the data center', 2013

[6] Big Switch Networks, "The Open SDN Architecturre", http://www.bigswitch.com/sites/default/files/sdn_overview.pdf, accessed 30 Aug 2015

[7] P. Goransson, C. Black, Software-Defined Networks: A Comprehensive Approach, Morgan Kaufmann, 2014

[8] Erickson, D., 'The beacon openflow controller'. Proc. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Hong Kong, China2013

[9] Extreme Networks Inc: 'OpenDayLight Technical Study'

[10] Racherla, S., Cain, D., Irwin, S., Ljungstrom,P., Patil, P., Tarenzio, A.,: Implementing IBM Software Defined Network for Virtual Environments IBM.com, September 2014

[11] ONOS, "Architeture Guide", https://wiki.onosproject.org/display/ONOS/Architecture+Guide, accessed 30 Aug 2015

[12] Cohen, R., Barabash, K., Rochwerger, B., Schour, L., Crisan, D., Birke, R., Minkenberg, C., Gusat, M., Recio, R., and Jain, V.: 'An intent-based approach for network virtualization', pp. 42-50