Computer Networks 81 (2015) 79–95

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey Paper

Software-Defined Networking: A survey

Hamid Farhady, HyunYong Lee *, Akihiro Nakao

Graduate School of Interdisciplinary Information Studies, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

ARTICLE INFO

Article history:
Received 30 January 2014
Received in revised form 16 February 2015
Accepted 17 February 2015
Available online 24 February 2015

Keywords:
Software-Defined Networking
Programmable networks
OpenFlow
Control and data planes

ABSTRACT

Software-Defined Networking (SDN) is considered promising to simplify network management and enable research innovations based on the decomposition of the control and data planes. In this paper, we review SDN-related technologies. In particular, we try to cover three main parts of SDN: applications, the control plane, and the data plane anticipating that our efforts will help researchers set appropriate and meaningful directions for future SDN research.

1. Introduction

Modern Internet infrastructure consists of a set of networking devices with purpose-built application-specific integrated circuits (ASICs) and chips that are used to achieve high throughput, thus realizing *hardware-centric networking*. However, the current hardware-centric Internet infrastructure suffers from several shortcomings, such as manageability, flexibility, and extensibility. Networking devices usually support a handful of commands and configurations based on a specific embedded operating system (OS) or firmware. As a result, network administrators are limited to a set of pre-defined commands, even though it would be easier, simpler, and more efficient to support more protocols and applications if it were possible to program network controls in ways that are more responsive and flexible. In addition, researchers usually have to make their own testbeds or take advantage of simulations rather than real world implementation scenarios to realize their ideas. In other words, innovation and research is costly under the current condition of hardware-centric networking.

To overcome such limitations, the Software-Defined Networking (SDN) concept has been proposed. SDN can be defined as "*an emerging network architecture where the network control is decoupled and separated from the forwarding mechanism and is directly programmable*" [1]. In SDN, there is a logically centralized controller that has a network-wide view and controls multiple packet-forwarding devices (e.g., switches) that can be configured via an interface (e.g., ForCES [2] and OpenFlow [3]). For example, an OpenFlow switch has one or more forwarding tables that are controlled by a centralized controller, thus realizing programmability in the control plane. Forwarding tables are used to control packets (e.g., forwarding or dropping). Therefore, according to the controller policy that manages the forwarding tables, an OpenFlow switch can act as a router, switch, NAT, firewall, or exhibit similar functions that depend on packet-handling rules. Due to its decoupled nature, SDN is believed to be a new networking technology that simplifies today's network operation and management and also enables network innovations and new network designs. Because of the potential benefits of SDN in the current Internet and future Internet architectures, such as information-centric networking [4], it has gained considerable attention from the community.

* Corresponding author.
E-mail addresses: farhadi@nakao-lab.org (H. Farhady), ifjesus7@gmail.com (H. Lee), nakao@nakao-lab.org (A. Nakao).

http://dx.doi.org/10.1016/j.comnet.2015.02.014
1389-1286/

An SDN instance consists of three major parts: application, control plane, and data plane (Fig. 1). The *application* label indicates a part that exploits the decoupled control and data plane to achieve specific goals, such as a security mechanism [5] or a network measurement solution [6]. Applications communicate with a controller at the control plane via the *northbound* interface of the control plane. The *control plane* is the part that manipulates forwarding devices through a controller to achieve the specific goal of the target application. The controller uses the *southbound* interface of the SDN-enabled switch to connect to the data plane. The *data plane* is the part that supports a shared protocol (e.g., OpenFlow) with the controller and handles the actual packets based on the configurations that are manipulated by the controller. Therefore, we believe that deeply understanding each part and investing balanced research attention into each of the three parts is important to maximize the potential benefits of SDN.

In this paper, we survey existing efforts on each part. In a similar spirit, several SDN survey papers [7–10] have been published recently. However, those papers mainly focus on the control plane and the application part, particularly OpenFlow-related work. On the other hand, in this paper, we try to cover all three parts anticipating that a survey of the three parts will help researchers set appropriate and meaningful objectives for future SDN research.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the history of SDN and several SDN-related terms. Section 3 introduces examples of the application and Section 4 surveys the control plane related work. Section 5 surveys the data plane technologies. Finally, after we discuss several future directions for SDN research in Section 6, Section 7 concludes this paper.

2. Background

2.1. Networking paradigm

Networking paradigms can be divided into three types according to the deployment of control and data planes (Fig. 2). In the traditional hardware-centric networking, switches are usually *closed* systems that have their own control and data planes and support manufacturer-specific control interfaces. Therefore, in the traditional hardware-centric networking, deploying new protocols and services

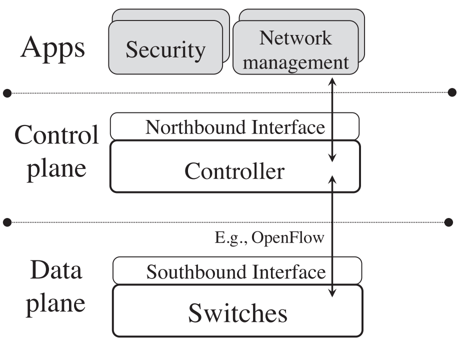

Fig. 1. Components of SDN.

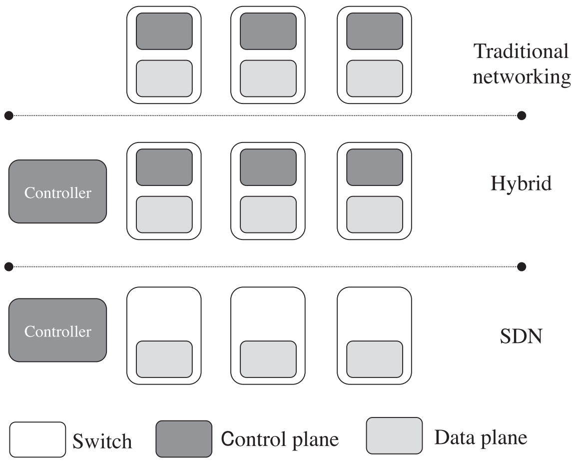

Fig. 2. High-level illustration of networking paradigms.

(and even new versions of existing protocols) is a challenge because all the switches need to be updated or replaced. In contrast, in SDN, switches become simpler (in terms of removing the control plane from the device) forwarding devices and a centralized controller derives the control mechanism of the network. This decomposition of control and data planes allows easier deployment of new protocols and services because the decomposition enables us to *program* switches via the controller. Finally, a hybrid approach supports both distributed and centralized control planes. For example, common commercial OpenFlow switches (see Section 5.1.2) are hybrid switches that support OpenFlow in addition to traditional operation procedures and protocols.

2.2. Historical foundations

Even though SDN is popular nowadays, several SDN concepts have been around for many years. In the following, we briefly review each of them.

2.2.1. Active networks

In active networks, each packet carries a program rather than raw data. When a network node receives a packet, the program inside the packet is executed and then different types of actions can be triggered against the packet (e.g., forward or drop) based on the data plane design. The idea of active networks relates to some in-network processing services and tries to treat network devices as an environment that reacts based on what the packet carries rather than passively transmitting bits from one node to another [11].

The active networking approach shows less interest in the control plane and is instead focused on providing a smart environment similar to end-point PCs compared with current dumb switches that can execute a limited set of procedures. That is, if we consider the end-points of a network, which are PCs, and servers as smart devices, then we can consider network controls (e.g., switches and routers) as dumb devices. They are dumb because they can execute limited types of tasks, albeit they do so rapidly.