SELÇUK
ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

SUJEST, c.4, s.2, 2016
SUJEST, v.4, n.2, 2016
ISSN: 2147-9364 (Elektronik)

AN OVERVIEW OF SOFTWARE DEFINED CAMPUS NETWORKS

[1]Mahir KUTAY, [2] Tuncay ERCAN

[1] Yaşar Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü, İZMİR
[2] Yaşar Üniversitesi, Bilgisayar Mühendisliği Bölümü, İZMİR
[1] mkutay@altekbt.com, [2] tuncay.ercan@yasar.edu.tr

(Geliş/Received: 10.11.2015; Kabul/Accepted in Revised Form: 26.03.2016)

ABSTRACT: The number of users and overall workload on computer networks increase continually. However, the management of the computer networks infrastructure in which taking the advantage of rapid changes in technology is still complicated and exteremely expensive. The concept of Software Defined Networks (SDN) proposes the development of software tools with new applications to decouple the network devices from the network traffic as similar as virtualization technologies. This will help computer networks being more flexible and contribute the management of Information Technologies with a higher functionality and lower cost.

Key Words: *SDN, Open Flow, Software Defined Network, Controller, Network Parameters.*

Yazılım Tanımlı Yerleşke Ağlarının Geliştirilmesi için Bir Derleme

ÖZ: Bilgisayar ağlarındaki kullanıcı sayıları ve yükleri sürekli olarak artmakla birlikte, ağ yönetim metodolojileri iletişim teknolojisindeki hızlı gelişmeye ayak uyduramadığı için karmaşık ve yüksek maliyetli bir işletme fonksiyonu olarak geride kalmıştır. Yazılım Tanımlı Ağ kavramı bilgisayarlarda uygulanan sanallaştırma ile benzer yöntemleri kullanarak ağ trafiğini düzenleyen cihazların yönetiminin ayrı bir yönetim katmanından yapılmasını sağlayacak uygulamaların geliştirilmesini önermektedir. Sanallaştırma ve ayrı bir katmandan yönetim, bilgisayar ağlarının daha esnek kullanılmasına olanak sağlayacak, Bilgi Teknolojileri yönetiminin düşük maliyetli ve kolaylıkla gerçekleştirilmesine katkıda bulunacaktır.

Anahtar Kelimeler: *SDN, Açık Akış, Yazılım Tanımlı Ağ, Yönetim Birimi, Ağ Parametreleri.*

INTRODUCTION

A huge increase in the number of smart devices, social media applications, mobile based services and management of network components have put pressure on the network operators trying to satisfy their customers. They require necessary skills for the installation and operation of network devices. A number of management operators still use limited CLI (Command Line Interface) commands (Kim *et al*., 2013). Due to the dynamic nature of network environment, operator's reaction to changes remains slower to write commands or use pre-defined scripts to handle them. Misconfigurations can lead to partial or total network downtime occurrences (Kim *et al*., 2011). Using traditional methods to apply a complete network-wide policy for overall network management becomes incredibly difficult.

DOI: 10.15317/Scitech.2016218527

In recent years, the management of larger networks including multi-vendor network devices has become more costly while the revenue obtained from their operation decreases (Sezer *et al*., 2013). SDN is an emerging alternative technology which accelerates configuration deployment and operational response time while reducing the network operational costs. SDN enables policy based workflow automation by using cloud-like architectures (Pavitra *et al*., 2014).

In this paper, we overview considerable amount of recent studies in the area of the architectural SDN frameworks for campus networks, planned future developments and the open issues that should be improved. The paper is presented as follows; in section 2, emergence of SDN concept is provided. SDN architecture is explained in section 3. Conclusions and open research areas are given in section 4.

EMERGENCE OF SDN CONCEPT

The historical background of SDN architecture is based on healing some disadvantages of static networks. In 1995, "Open Signaling Working Group (OPENSIG)" formed at Columbia University began a research about open network control issues in ATM, Internet and mobile networks. They focused on the definition, implementation and experimentation of open programmable networks (Campbell *et al*., 1999). In this study, researchers stated that the hardware and the control software in network devices should have been separated to provide more powerful, flexible and programmable network management systems. Unfortunately, their effort became unsuccessful because of the application difficulties in hierarchically placed router switch architecture.

OPENSIG research had ignited a chain of new researches. An IETF workgroup (Doria *et al*., 2002) proposed "General Switch Management Protocol (GMSP)" to establish and maintain the control state of an ATM, frame-relay or MPLS switch. The last version, GMSPv3 is able to establish both unicast and multicast switch connection to control switch system resources and QoS features.

"Active Network Structure" proposal by Teannenhouse, based on programmable switch approach maintains the existing packet/cell format (Teannenhouse *et al.,* 2002). It provides a discrete mechanism that supports a program selection and downloads functionality by network administrators instead of users.

4D protocol proposed by Greenberg, decomposes the functions of network control into four planes (Greenberg *et al.,* 2005).

- A decision plane that is responsible for creating a network configuration
- A dissemination plane that gathers information about network state to decision plane and distributes decision plane output to routers.
- A discovery plane that enables devices to discover.
- A data plane for forwarding network traffic.

"Sane/Ethane Projects" developed by a work group in Stanford University can be considered as the first approach version of the Openflow protocol (Casado *et al.,* 2006). Sane is a clean-slate protection architecture for enterprise networks (Casado *et al.,* 2007). It defines a single protection layer that governs all connectivity within the network. All routing and access control decisions are made by a logically-centralized server that grants access to services by handing out capabilities according to access control policies. Ethane allows network administrators to define a network-wide policy to be applied to all network devices. Ethane design includes some differences (naming, policy declaration and security checks) from SDN which are performed by a central controller.

The 5-year research collaboration between Stanford and California Universities resulted by the release of the first version of the Openflow protocol (version1.1) and the establishment of "Open Networking Foundation" in 2011.

SDN ARCHITECTURE

SDN architecture basically depends on the Openflow protocol (MckKeown *et al.*, 2008). A traditional switch or router device performs both data and control path operation itself. In the open flow architecture, the network device contains one or more additional flow tables that contain the necessary forwarding information of the processed packets. Flow entries typically contain the following data (Astuto *et al.*, 2014).

- Match fields (or matching rules) hold the information like packet-header, entrance port and metadata.
- Counters collect the statistics of the flows like flow-duration, number of packets and bytes received.
- Action sets define how to handle matching packets like drop, forward or modify header.

When a packet arrives to the switch, matching fields are compared with flow entries. If a match is found, the predefined action is performed. If there is no match, switch looks up the second miss-flow table that defines the rules for no match cases like drop the packet or send it to the controller.

Controller and switch communicate over a secure channel by using a message set (Hayward *et al.*, 2013). Controller is a remote device which can delete, add or update flow entries of the switch. Openflow switch operation flowchart is given in Figure 1.

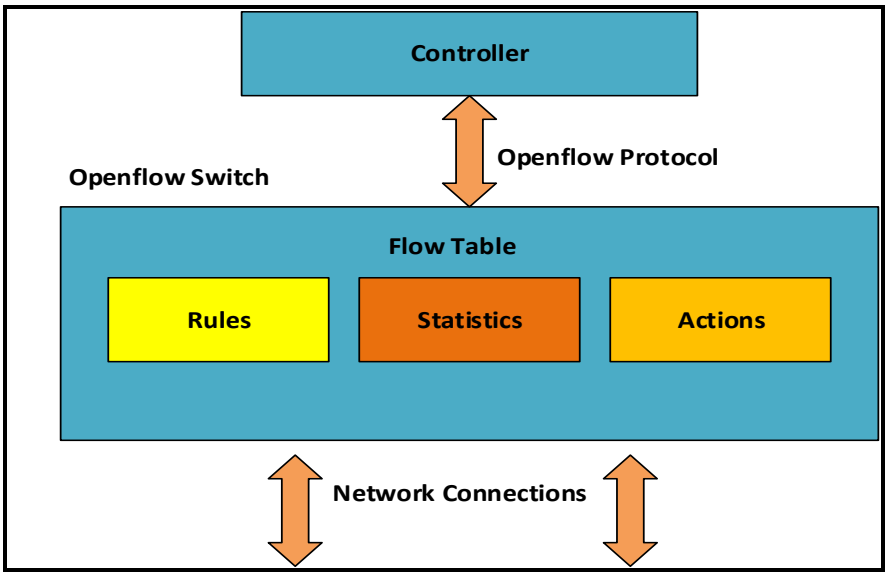


Figure 1. Openflow switch operation flowchart

Controller operation

Control and management operations of SDN are performed by a central controller. Controller has the full network topology and address information. When a switch sends a packet-in request to the controller for a packet which is not in the flow tables, controller should handle the forwarding or taking any other actions like drop, modify header and change route or update flow table for subsequent flows (Luo *et al.*, 2012). In this reactive mode, controller listens to the traffic, takes actions or defines routes on demand. Openflow controllers are programmed with a control interface by network operators (Fernandez, 2014)."

As depicted in Figure 2, larger networks require a high level decision policy layer which will include more SDN controllers namely SDN application layer (Alsher, 2015). This layer is responsible from network virtualization, traffic engineering, routing monitoring and quality of service applications.

Communication between the application layer and SDN controller(s) is realized by a set of application programming interface programs (APIs) which is called "North Bound" (Akyildiz *et al.,* 2014). Similarly, the interface between SDN controller and Openflow network devices is called "South Bound".

SDN architecture allows a unified and global view of complicated networks, and provides a powerful control platform for the network management over traffic flows through the interactions among these layers.

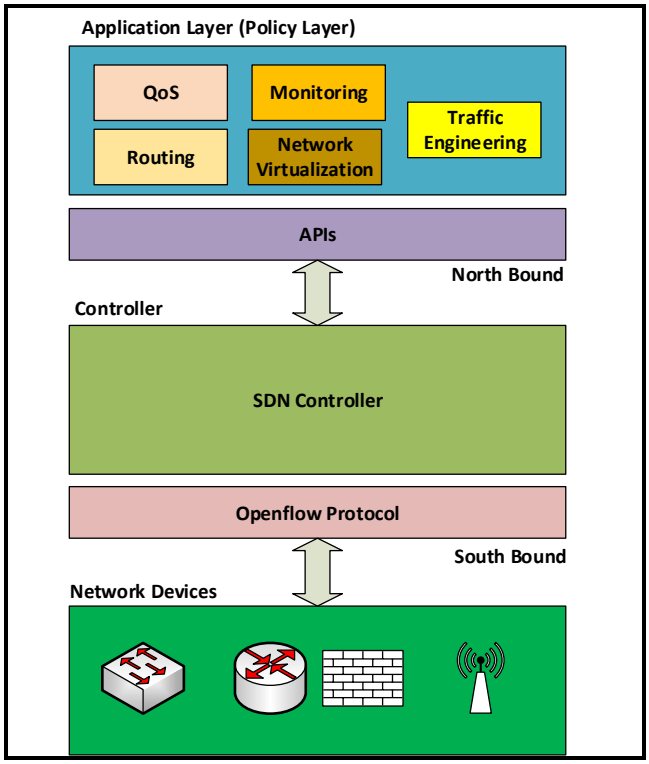


Figure 2. SDN layers

Central management tool with a GUI provides both wired and wireless networks to be more flexible since they may have been programmed centrally and more cost effective from the point of management and operational views. The overall representation of the campus SDN infrastructure is given in Figure 3.

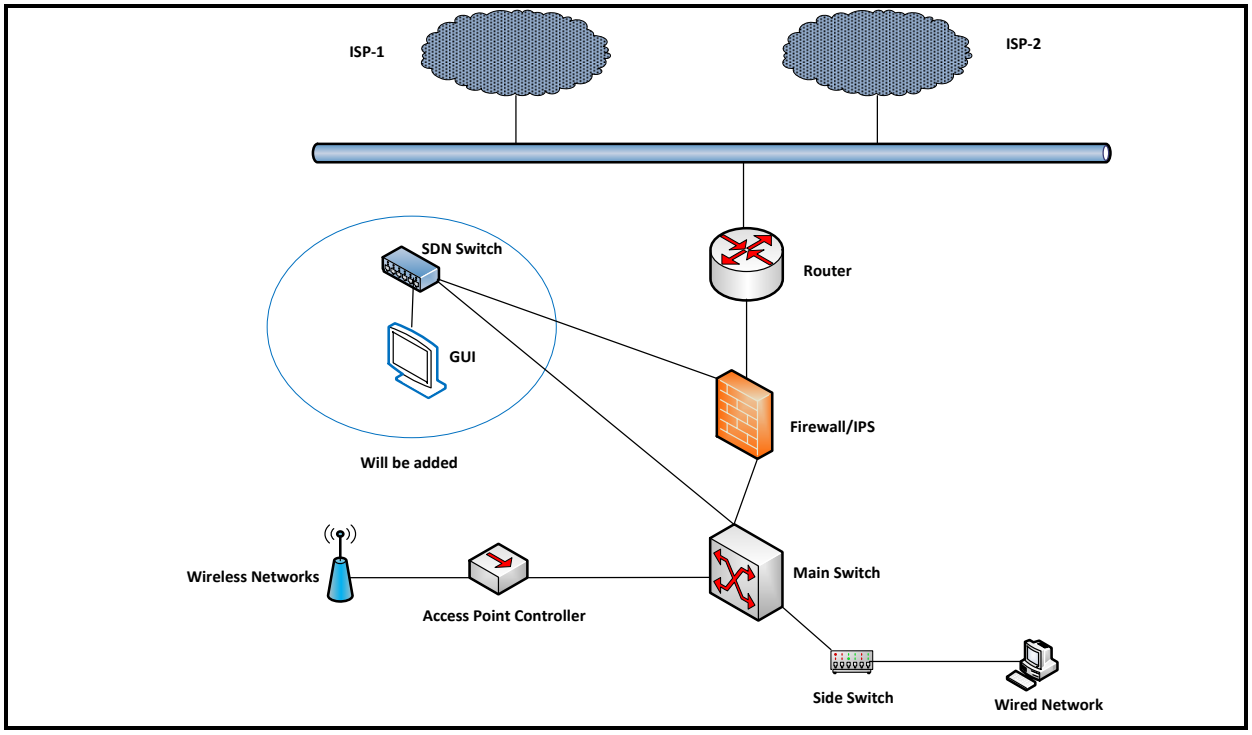


Figure 3. Campus SDN infrastucture

Selection of Controller Parameters

Selection of controller parameters requires a detailed traffic analysis on the wired and the wireless networks within the campus environment. Traffic data (running applications, protocols, ports, device interfaces, latency, throughput etc.) on the network devices should be archived on a time-based recording. The matching fields of these records (ports, packet headers, and other metadata) in the flow entry data including match fields, priority, counters, instructions, and timeouts attributes in the Openflow protocol should be analyzed (Openflow-*spec- v1.4.0.*, 2015).

Following the integration of Openflow switch device into the network, similar measurements should be executed, the effect of those changes should be evaluated on latency and throughput values. Additionally, softbound parameters of the controller software will be adjusted on web-based GUI and on northbound APIs. A flowchart of the selection of controller parameters methodology is given in Figure 4.

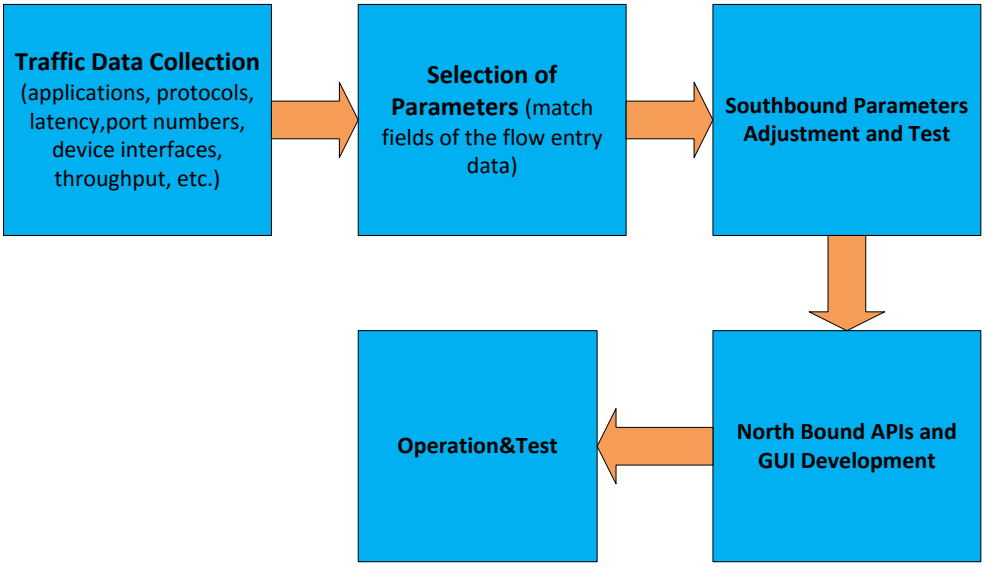


Figure 4. Selection of controller parameters

Traffic Analysis

An industry standard WinPcap or other packet sniffers can be used to analyze campus network traffic. The WinPcap capture packets at the network adapter level, store and execute offline analysis with protocol and application types by using filters (Jiang *et al.,* 2014). WinPcap network monitoring tool interactively analyze traffic load changes by time. Traffic analysis is very important to determine the current performance of the campus network for the following issues:

- Average service access time of the users
- Distribution of users between access points
- Distribution of users between applications
- Average number of mobile users
- Decision on SDN matching attributes

Data collection should be performed on wireless and wired networks of the campus until sufficient information collected. By the use of WinPcap analysis tools, traffic information on network devices with parameter (device-port, bandwidth, latency, application, protocol etc.) variations with time should be documented.

Southbound Interface Adjustment

Controller-switch communications is controlled by southbound interface. The switch and controller communicate through a TLS connection by using standard message set. The TLS connection is initiated by the switch on startup to the controller, which is located by default on TCP port 6653. The switch and controller mutually authenticate by exchanging certificates signed by a site-specific private key. Each switch must be user-configurable with one certificate for authenticating the controller (controller certificate) and the other for authenticating to the controller (switch certificate) (Openflow-*spec- v1.4.0.,* 2015).

The southbound interface, regardless of solution implemented in a specific controller, is well standardized. Openflow is the most commonly used standard. This leads to a situation where underlying network devices in the SDN controller network should be interchangeable and non-vendor specific.

Northbound Interface Adjustment

Northbound interface is a set of APIs that is controlled by a web-based GUI to dominate the Openflow switch parameters. There is a lack of standardization in this area, despite the fact that "Open Network Foundation" has a working group called "Northbound Interface Working Group" (Openflow-*NIWG Charter v1.1.,* 2015). The working group has no releases yet and the situation has escalated into one where every vendor has their own proprietary northbound API with varying feature sets.

The on market paid vendor specific northbound solutions are HP VAN SDN Controller" (HP Company, 2015), "Juniper's OpenContrail" (Juniper Networks, 2015) and "Cisco APIC as a part of the Cisco ONE –platform" (Cisco Networks, 2015).

There are at least three significant and continuously developed northbound interfaces which are released as open source: "Project Floodlight" (Project Floodlight Community, 2015), OpenDaylight (Linux Foundation, 2015), and the duo of NOX and POX controllers (NOXRepo Community, 2015). Of these both Project Floodlight and OpenDaylight are also released as paid, closed source versions by major vendors participating in their open source development; IBM's proprietary SDN controller is based on OpenDaylight and Project Floodlight acts as a base for "Big Switch Networks' Big Network Controller".

Northbound APIs are "Representational State Transfer" (REST) based (Kreutz *et al.,* 2014). REST as an architectural style is the most used approach for the modern web services. As the main features which attract developers, we can name here the uniform interface, statelessness and self-descriptive messages. They can be written in any programming language of choice like Java, C, C++, Python, C#, or PHP. One example from "Neutron API (OpenStack)" is given in Figure 5 (Denton, 2014).

```
GET /v2.0/networks?limit=2
Accept: application/json
(Part of the Response):
{
  "networks":[
  {
"status":"ACTIVE",
    "subnets":["a318fcb4-9ff0-4485-b78c-e6738c21b26"],
    "name":"private",
    "admin_state_up":true,
    "tenant_id":"625887121e364204873d362b553ab171",
    "id":"9d83c053-b0a4-4682-ae80-c00df269ce0a",
    "shared":false
  }]
}
```

Figure 5. The request is a typical REST

Northbound APIs are named as "Administration and Management Application" (Feng *et al.,* 2009) and accessed by a user-friendly "Graphical User Interface (GUI)". Communication dataflow diagram of a SDN system is given in Figure 6.

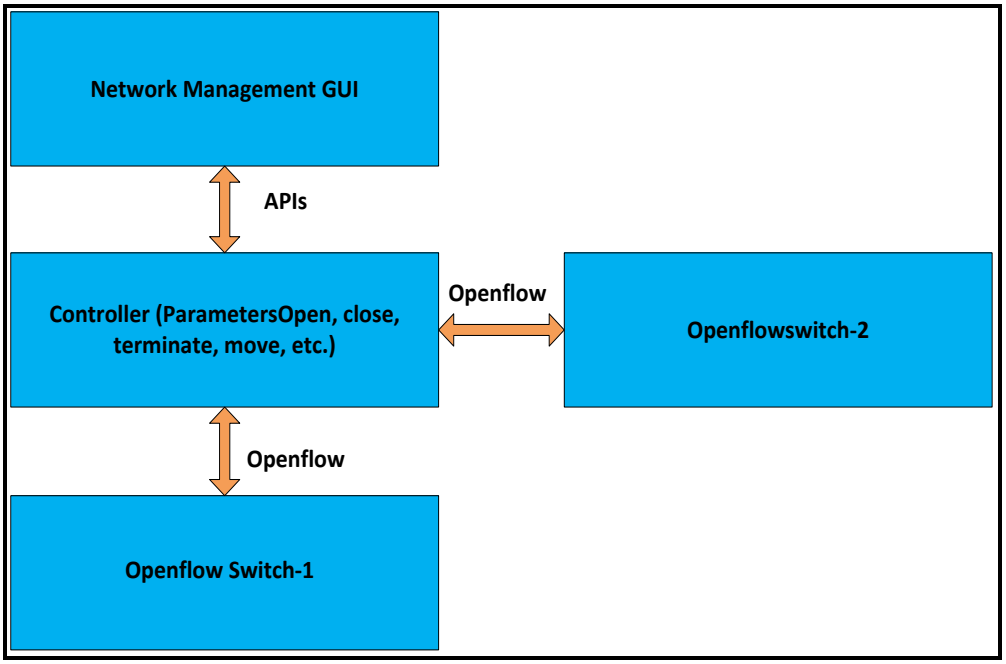


Figure 6. SDN dataflow diagram

CONCLUSIONS AND OPEN ISSUES

Basic function of data networks is to allow or deny data exchanges between remote end points, e.g., hosts and servers. By configuring network devices, operators can decide paths on which data are forwarded. Different data paths correspond to different performance and resource utilizations. This task is becoming more inefficient, time consuming and expensive in large multivendor campus networks.

Addition of software defined networking capabilities to existing campus networks replaces high cost "Controller Devices". A fully software-based controller structure which is open to new developments enables campus networks to provide low cost and high quality service.

An open northbound API allows anybody to develop own application. Only equipment vendors insist on standardization of northbound interface. The Open Networking Foundation, a consortium dedicated to promoting and commercializing software-defined networking, has decided not to issue of northbound API standardization for fear of hampering new innovations. As a result, more than 20 different SDN controllers are currently available having northbound APIs that vary based on the needs of the appliations.

SDN has the potential to be expanded to large GSM mobile networks. Security and dependability of the SDN networks are open issues for today. SDN transfers the intelligence existing on physical network devices to a software layer. SDN networks may become the target of hackers who can take over the control of a large GSM network. SDN is still at the development stage but has matured enough to the point where designed products are being deployed to the market.

REFERENCES

Alsher A., 2015, "An Overview Network Virtualization and Cloud Network as a Service". International Journal of Network Management, 25, 1-30.

Akyildiz I.F., Lee A., Wang P., Luo M., Chou W., 2014, "A Roadmap for Traffic Engineering in SDN-Openflow Networks", Computer Networks, 71, 1-30.

Astuto B., Nunes A., Mendonca M., Nguyen X.N., Obraczka K., Turletti T., 2014, "A Survey of Software Defined Networks: Past, Present and Future of Programmable Networks". IEEE Comm. Survey and Tutorials, 16(3).

Campbell T., Katzela I., Miki K., Vincente J., 1999, "Open Signaling for ATM, Internet and Mobile Networks", ACM SIGCOMM Computer Communication Review, 29(1), 97-108.

Casado M., Freedman M.J., Pettit J., Luo J., McKeown N., Shenker S., 2006, "Sane: A Protection Architecture for Enterprise Networks", 15th Conf. on USENIX Security Symposium, Conference Proceeding (Article no. 10).

Casado M., Garfinkel T., Akella A., Freedman M.J., Boneh D., McKeown N., Shenker S., 2007, "Ethane: Taking Control of the Enterprise", ACM SIGCOMM Computer Communication Review, 37(4), 1-12.

Cisco Networks, (23.10.2015), "Cisco APIC", http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/products-technical-reference-list. html.

Denton J., 2014, "Learning OpenStack Networking (Neutron)", Packt Publishing Ltd.

Doria A., Helstrand F., Sundell K., Worster T., 2002, "General Switch Management Protocol GMSPV3", Internet Requests for Commence (RFC), 2070-3292, ISSN(print), 2070-1721.

Feng X., Shen J., Fan Y., 2009, "REST: An Alternative to RPC for Web Services Architecture", IEEE In Future Information Networks, Conference Proceeding (ICFIN), 7-10.

Fernandez M.P., 2014, "Evaluating Openflow Controller Paradigms", ICN2013 Conference Proceeding, 151-157.

Greenberg A., Hjalmytsson G., Maltz D.A., Myers A., Rexford J., Xie G., Yan H., Zhan J., Zhang H., 2005, "A Clean Slate 4D Approach to Network Control and Management". ACM SIGCOMM Computer Comm. Review, 35(5), 41-54.

Hayward S.S., O'Callaghan E., Sezer S., 2013, "SDN Security: A Survey", IEEE SDN4FNS Conference Proceeding, 56-62.

HP Company, (23.10.2015), "Search HP.com", http://www8.hp.com/us/en/networking/sdn/devcenter-index. html.

Jiang L., Yang X., Li T., 2014, "The Analysis and Design for a Network Analysis System Based on WinPcap". Communications Security Conference Conference Proceeding, Article No. 731.

Juniper Networks, (23.10.2015), "Contrail Open SDN", http://www.juniper.net/us/en/products-services/ sdn/ contrail/.

Kim H., Benson T., Akella A., 2011, "The Evolution of Network Configuration: A Tale of Two Campuses", ACM Internet Measuring Conference, SIGCOMM, Conference Proceeding.

Kim H., Feamster N., 2013, "Improving Network Management with Software Defined Networking", IEEE Comm. Magazine, 51(2), 114-119.

Kreutz D., Ramos F., Verissimo P., Rothenberg C.E., Azodolmolky S., Uhlig S., 2014, "Software-Defined Networking: A Comprehensive Survey". ArXiv preprint, arXiv:1406. 044.

Linux Foundation, (23.10.2015), "Open Daylight Project", http://www.opendaylight.org/.

Luo T., Tan H.P., Quan P.C., Law Y.W., Jin J., 2012, "Enhancing Responsiveness and Scalability for Openflow Networks, via Control Message Quenching", IEEE ICTC Conference Proceeding, 348- 353.

NOXRepo Community, (23.10.2015), "Open Source Control Platforms for Software Defined Networks", http://www.noxrepo.org/pox/about-pox/.

McKeown N., Anderson T., Balakrishnan H., Parulkar G, Peterson L., Rexford J., Shenker S., Turner J., 2008, "Openflow: Enabling Innovation in Campus Networks", ACM SIGCOMM Computer Comm. Review, 38(2).

Open Networking Foundation, (23.10.2015), "Openflow Switch Specification (Wire Protocol 0x5)", http://www. opennetworking.org.

Open Networking Foundation, (23.10.2015), "Northbound Interface Working Group Charter v1.1.pdf", http://www. opennetworking.org.

Pavitra H., Sagar B.M., Srivanasan G.N, Parashant V., 2014, "An Overview of Software Defined Networking", International Journal of Science, Engineering and Technology Research, 3(11), 3053-3055.

Project Floodlight Community, (23.10.2015), "A Big Switched Networks Sponsored Project", http://www. Projectfloodlight.org/floodlight/.

Sezer S., Hayward S.S., 2013, "Are We Ready for SDN? Implementation Challenges for Software Defined Networks", IEEE Communications. Magazine, 51(7), 36-43.

Teannenhouse D.L., Wetherall D.J., 2002, "Towards an Active Network Architecture", DARPA Active Networks Conference and Exposition.