Solving Management Constraints of Traditional Networks using the Concept of Software Defined Networking

Muhamed Begović, Himzo Bajrić

Abstract: Computer networks, and in particular the Internet, have significantly evolved in relation to their beginnings. The number of users is constantly increasing, there are more and more mobile and wireless users, the quantity and type of traffic being transmitted is no longer the same, the way the storage and provision of content changes, etc. Traditional network architecture has not been made to meet the demands placed before it by the traffic that travels through the networks today, which significantly complicates the process of managing these networks. Software defined networks provide the opportunity to make the most of the architecture tailored according to the behavior and expectations of today's users as well as content and services providers by simplifying network management, expanding the range of opportunities and fostering innovation.

Keywords: computer networks, computer network management, network architecture.

I. INTRODUCTION

There is a lot of definitions of network management. It can be simply defined as a process of configuring the network to achieve a different kind of tasks. In a broader sense [1], network management is defined as the execution of the set of functions required for controlling, planning, allocating, deploying, coordinating, and monitoring resources of a telecommunication network or a computer network, including performing functions as initial network planning, frequency allocation, predetermined traffic routing to support load balancing, cryptographic key distribution authorization, configuration management, fault management, security management, performance management, and accounting management.

A key aspect to network management is configuring the network. Juniper White paper [2] shows that human factors are responsible for 50 to 80 percent of network outages, and the most worrying thing is that the cause is no longer incompetence, but system complexity with large number of devices with multiple interactions and hard predicting relationships that make it very hard or impossible to know with certainty how will the implementation of a given policy affect the rest of the network.

The interactions between multiple routing protocols can in traditional architecture lead to unpredictability due to configuration that is distributed across hundreds or more devices. Furthermore, each autonomous system on the Internet is independently configured, and the interaction between their policies can lead to unwanted behavior.

Revised Version Manuscript Received on October 27, 2017

M.Sc. Muhamed Begović, Teaching Assistant, Faculty of Traffic and Communications, University of Sarajevo, Bosnia and Herzegovina.
E-mail: m.begovic@live.com

Dr. Himzo Bajrić, Associate Professor, Faculty of Traffic and Communications, University of Sarajevo, Bosnia and Herzegovina.
E-mail: himzo.bajric@bih.net.ba

Compared with traditional vendor-specific low-level configuration, Software Defined Networking (SDN) on a high level provides exactly the primitives that operators need to run the network better.

Since the Open Networking foundation is the biggest, and the most influential organization in terms of SDN, their definition of SDN is accepted in wider circles. ONF defines SDN as an emerging network architecture where network control is decoupled from forwarding and is directly programmable [3]. More pragmatic definition [4] views SDN as a functionality that enables the network to be accessed by operators programmatically, allowing automated management and orchestration techniques; application of configuration policy across multiple routers, switches, and servers; and the decoupling of the application that performs these operations from the network device's operating system. Kreutz et al. define SDN through four main pillars [5]: (1) The control and data planes are decoupled. (2) Forwarding decisions are flow-based, instead of destination-based. (3) Control logic is moved to an external entity, SDN controller. (4) The network is programmable through software applications running on top.

II. RELATED RESEARCH

The idea of programmable network is not so new as one might think. It is a rather new concept but some ideas that this new network architecture is based on has its roots in technologies that evolved in final years of last millennium. SDN even uses early telephony networks principle [6] of clear separation of control and data planes in order to simplify network management and the deployment of new services.

The history of SDN can be divided into three stages [6], each with its own contributions:

- active networks introduced programmable functions leading to greater innovation;
- control and data-plane separation, which developed open interfaces between the control and data planes;
- the OpenFlow API and network operating systems, which represented the first widespread adoption of an open interface.

DARPA research community introduced active networking concept [7] around 1996 as an answer to rising problems that the traditional networks architecture causes in networking systems development, growth and innovations. Networks are called active because switching elements can perform computations on packets and modify the content they are carrying and processing can be customized on a per user or per application basis.

The idea is that a user (or network administrator) of an active network sends a program which executes on the switching element when their packets are processed. There were two proposed approaches [8]. First approach is called programmable switch approach (1) and it "maintains the existing packet/cell format, and provides a discrete mechanism that supports the downloading of programs". It is clear that even in this early idea we have a separation of injecting the programs from processing the messages. Capsule approach (2) replaces traditional packets by "active miniature programs that are encapsulated in transmission frames and executed at each node along their path".

It is important to note that the very first concept of separating control and forwarding processes can be found in early 1990s by the BSD 4.4 routing sockets [9].

Previous to 2004, configuration was distributed which lead to complexity, buggy and unpredictable behavior and low manageability [10]. Around 2004, emerged the idea to control the network from a logically centralized high-level program [11]. Even in that time, it was obvious that the main limitation of traditional network architecture was distributed approach in computation of paths through the network on standard routers. Routing Control Platform (RCP) was introduced as a solution that provides routes selection on behalf of the IP routers in each AS and exchange reachability information with other domains [12,13].

In 2005 researchers generalized the concept of RCP for different planes [14]. 4D planes philosophy consists of the decision plane which computed the forwarding state for devices in the network, the data plane, which forwarded traffic based on decisions made by the decision plane, and the dissemination and discovery planes, which provide the decision plane the information that it needs to compute the forwarding state, which ultimately gets pushed to the data plane.

Idea of new architecture emerged embracing the 4D philosophy and driven by the shortcomings of classical architecture in ensuring the effective functioning and management of large and complex enterprise networks, with various vendor equipment, multiple protocols and security mechanisms. The new Ethane network architecture [15] was presented as a new solution to the mentioned problems that will not make network more complicated and will not only "hide complexity" as it is done in some earlier methods such as middleboxes and adding new functionality to existing networks (access lists, VLANs, spanning tree algorithms). Ethan supports a new concept "by not allowing any communication between end-hosts without explicit permission." The central controller has a global view of the network and calculates the path and movement through communication nodes, and simple job of forwarding is performed by Ethan switch that works according to direct orders of the controller.

Growing security issues, that are especially big problem in enterprise network which do not tolerate data loss or confidential information leaks, lead to proposing SANE architecture (Secure Architecture for the Networked Enterprise) [16]. It represents a concept of architecture with security as the main goal, and one of the three basic principles is also to have "only one trusted component" so the policy enforcement is centralized.

Around 2008, software defined networking concepts effectively hit the mainstream, thanks to the emergence of OpenFlow under Open Networking Foundation (ONF) [3] consortium that is leading the advancement of SDN and standardizing elements of SDN architecture.

III. PLACEMENT OF NETWORK MANAGEMENT FUNCTIONS IN SDN ARCHITECTURE

Architecture of SDN networks includes three layers according to the reference model [3] proposed by the ONF: infrastructural layer, control layer and application layer. Layers perform mutual communication via interfaces. Fig. 3. shows simplified SDN architecture modeled by the ONF proposal and supplemented with position and short description of some basic processes in SDN enabled networks.

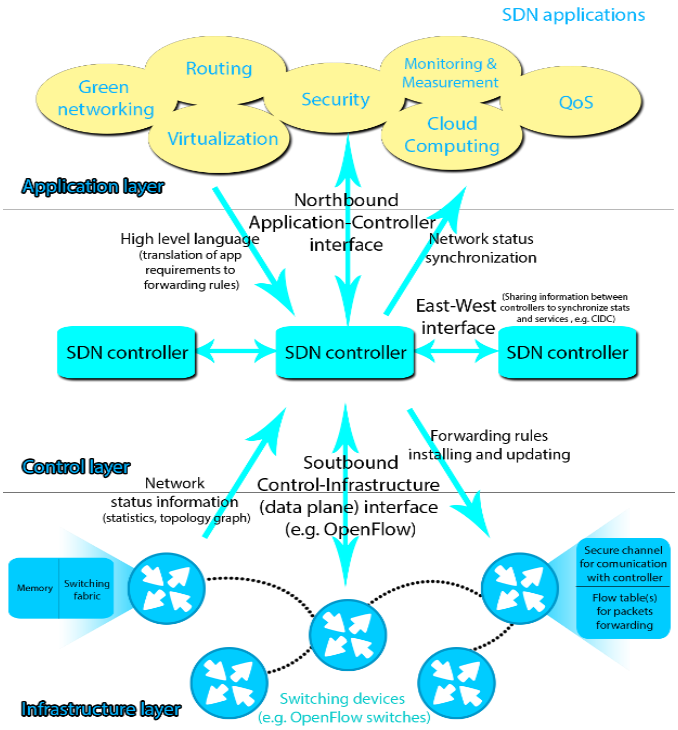

Fig. 1.SDN architecture model with basic processes.

A. Infrastructure layer

The infrastructure layer includes transmission media and switching devices that have the task of forwarding packets, and collecting data about the state of the network. SDN-enabled switching devices are exempt from the process of selecting the best route. SDN switch receives instructions from SDN controller about packet forwarding rules, these rules are stored in one or more tables in its memory, and based on them decides where to forward packets. This device has no intelligence of its own, nor can autonomously make decisions about forwarding packets because they are no longer required to understand the routing process and the principles of routing protocols, but only need to understand the language in which the SDN controller addresses them.

Routing is not exclusively based on the destination IP or MAC address, but also on other parameters from the headers such as TCP or UDP port, or VLAN tag. SDN switching devices can be implemented as software running on a host operating system, on vendor's switch, or on an open network hardware [17]. Since the ONF SDN architecture is the most widespread, it is important to briefly describe how does ONF OpenFlow enabled switch works. Specifications of OpenFlow switches are defined by ONF and the latest version is 1.5.1 released in March 2015 by ONF TS-025. OpenFlow switch is characterized by three components (1) flow table – one or more connected through "pipelines", (2) secured channel - connection with the controller, (3) OpenFlow protocol – way of communication with the controller. Structure of one OpenFlow entry is shown in Table I.

TABLE I. MAIN COMPONENTS OF A FLOW ENTRY IN A FLOW TABLE [18]

Match Fields	Priority	Counters	Instruc-tions	Time-outs	Cookie	Flags

Open Flow switch works using one or more flow tables that contain rules. of matching packets to a particular flow, actions to be taken when the belonging flow of the package is determined, as well as counters that maintain statistics of forwarded packets. When a packet arrives, lookup process starts in the first flow table. If a match is found, it is forwarded to corresponding outgoing port. If a match is not found, the packet is forwarded to the next flow table. If there is no match in any of the existing tables, packet can be forwarded to the controller for the decision, or it can be dropped.

B. Control layer

SDN controllers are situated at the control layer and they are in charge of importing packet forwarding rules based on the policy received from the application layer, and in this way, they govern the operations of switching devices on the infrastructure layer. Control layer in SDN networks is very important because it is a place where most of the complex network issues happen.

Since SDN controller is placed in the very middle of the SDN architecture - control layer (see Fig. 3.), besides lower infrastructure layer, it also has a bond toward upper application layer. Applications use open APIs and through controllers access and govern the behaviour of switching devices at the bottom of the SDN architecture to meet the requirements. Controller-infrastructure interface is refered to as southbound interface. In order to be able to perform an essential function of SDN - dynamic network management, it is necessary to have the information about the current status of the network. This information is collected by one or more SDN controllers from statistics kept by the devices on the infrastructure layer.

Wenfen Xia et al. [17] presented possible logical design of the SDN controller considering its primary functions and distinguish four main building components of SDN controller. *Communication in north - south direction*: High level language (1) is responsible for delivering the policy requirements of applications and transforming them into packet forwarding rules and needs to be developed as a language whose syntax enables simple translation of application requirements in order to ensure the desired behavior of the network. Controller then needs to generate packet forwarding rules (2) extracted from the policy received from the applications, install them into switching devices, and update the rules as the network needs to be dynamically controlled. *Communication in south - north direction*: Application layer needs to have unique view of the network, which enables the process of making decisions. SDN controllers collect status of the network (3) from switching devices that keep statistics on forwarding, or they themselves report that statistic to the controller. If there are two or more SDN controllers governing the behavior of switching devices, it is crucial to maintain synchronized view of the network status (4), otherwise applications that receive unsynchronized information can make wrong network management policies resulting in unwanted network behavior.

Table II contains examples of SDN controllers from first OpenFlow controller realization – NOX, to latest proposals like ParaFlow.

TABLE II. CONTROLLER IMPLEMENTATION EXAMPLES

Controller	Progra m. language	Description
NOX [19]	C++	First OpenFlow controller. Supports centralized architecture and has ad-hoc API.
Open-Daylight [20]	Java	Enables support of multiple southbound protocol plugins and a diverse set of services and apps, brings network applications closer to the network and allows developers and researchers to focus on SDN APIs rather than protocols used to communicate with devices.
Onix [21]	Python, C	Onix allows operating with a global network view, and use of basic state distribution primitives provided by the platform. Ensures a general API for control plane implementations, while allowing them to make their own trade-offs among consistency, durability, and scalability.
ParaFlow [22]	C++	Multithreaded SDN controller that supports fine-grained parallelism by exploiting application parallelism and utilizing multi-/many-core resources to accelerate event processing. Provides a flow-based programming interface for application developers to program with network flows rather than various types of low-level events.

Controllers must have a way to communicate in order to exchange various types of information (e.g. synchronize network status and topology view). This communication is achieved through east-west interface (see Fig.1.).

C. Application layer

SDN application layer contains applications that create rules based on collected data on the status of the network and forward them to the controllers that ultimately govern the switching devices. Application-controller interface is referred to as northbound interface.SDN applications have a view of the entire network, thanks to the data on the current status of the network obtained by the controllers and delivered to them through the northbound interface. Thanks to the dynamic programs it is possible to have flexible

network management and network resources allocation in real time, changing the behavior according to the needs of users, administrators and network operators.

Through open APIs and programmable platform, it is relatively easy and not as much time-consuming (as in traditional networks) to develop and implement applications without worrying about multi-vendor environment, and without being tied to details of their implementation since SDN applications are not so much "network-aware" as "network-capability-aware" [3]. SDN applications enable enhancement of existing traditional networks main functions (e.g., routing of packets through network and security management), as well as many new capabilities that were hard or impossible to realize with traditional architectures, e.g., energy aware routing or saving energy in Data center networks [23] and QoS over heterogenous networks [24].

IV. IDENTIFIED POTENTIAL PROBLEMS IN INTRODUCING AND EXPLOITATION OF SDN WITH PROPOSED SOLUTIONS

A. Scalability

Since the idea of SDN emerged, and the first SDN controller was made and tested [25], there were concerns over the scalability. One SDN controller can manage very large networks [26], even those which are composed of several thousand of switches (tens of thousands of ports), but regardless of this, at some point in time, the exhaustion of CPU or memory will inevitably and undoubtedly happen. It is a simple math: bigger the network - larger the requests and information sent to the controller. This is especially a problem in network environments with high flow initiation rates like in data centers and big enterprises [27], [28].

If, by its nature, SDN is centralized, it would mean that there are real problems of scalability as the network (more precisely SDN controller) in certain situations is unable to ensure expanding. However, in reality, the term used in SDN is "logically centralized", which essentially means that "it is a centralized programmatic model, but was really distributed" [26]. Thus, the devices in charge of computation "appear as a single machine but in practice they can be replicated" [29].

B. Inter-controller communication

Concept of distributed control plane elements rises another issue. To ensure that we preserve one of the main benefits of SDN – a unique global view of complete network – those distributed control elements must have a standardized way to communicate and exchange various information about network status. If more controllers are present in the network, it is necessary to synchronize the information gathered by the various controllers to be able to forward the information to applications that lie on the layer above. There are two approaches to this problem. The first speaks of the logically centralized control plane in which there is a shared database for all controllers to be used for synchronization of decisions. Example of this approach can be found in OpenDayLight (ODL) controller [20]. Because of the limitations of this approach in case of large and distributed networks, other ideas appeared on logically distributed control plane in which each controller takes care of its domain and distributes the necessary data to other controllers. Example of logically distributed approach is Disco [30] (Distributed multi-domain SDN controllers) control plane organization that provides manageable inter-controller channel with agents for sharing network-wide information and supporting end-to-end network services.

Although it is clear what is the importance of the interface between controllers, there is still no standardized way for controllers to exchange information in distributed environment, so CIDC - Communication Interface for Distributed Control plane [31] was proposed as an east-west SDN interface to ensure reliable message exchange and support for achieving distributed network services. First simulations and testing showed that CIDC has significantly better results than previous solutions like ODL "in terms of delay, overhead, and system consumption such as CPU and Memory".

C. Network topology, increased delays, failure recovery

Some other questions that rise are how to determine the exact number of controllers that we need and what will be their geographical position in the network [32], what will control plane topology look like, is it better to have flat or hierarchical organization, and how to ensure their reliability and security.

Flow initiation procedure overhead and resilience to failures [33] are as well things to be worried about. Reactive requesting, computation and installing of flow forwarding rules creates additional delays. The main concern over increased delays are not within controllers and their speed of reacting to new flow rule requests, but in switching devices that have limited hardware capabilities and relatively week management CPUs. Reacting to failures and convergence of the network is not a problem bigger than in traditional networks as long as the failed link or switch malfunction does not effect on switch-controller or controller-controller communication. However, if this happens, it can lead to serious problems due to the fact that a controller cannot react to the problem if itself did not receive the information about the failure. One of the possible solutions can be creating out-of-band control network, but even with this approach all the problems are not solved.

D. Deployment issue

There are proposals on how to overcome the limitations of traditional network architectures by using the existing routing protocols to create programmable networks or how to combine SDN with some other ideas on network management, i.e. how to improve SDN concept and work around its potential flaws.

Stefano Vissicchio et al., present an idea called Fibbing [34], [35]. Main arguments in favor of this idea are that despite its advantages SDN is sacrificing robustness provided by the distributed routing protocols,

building SDN controller to replace the tasks currently performed by scalable, robust and fast responding OSPF or IS-IS protocol is in itself a challenge, as well as the fact that even the simple task of updating forwarding rules on the switch would be a major problem for the central controller that controls the

operation of hundreds of switches. The deployment is cited as another drawback of SDN, because many networks have a huge installed base of devices, management tools, and employees who are not familiar with the principles of SDN.

Fibbing technique relies on a distributed architecture of the current networks and routing protocols, but controls the information centrally by lying routers about network topology, adding false nodes, and thus forwards the traffic according to current needs. Fibbing works by injecting fake link state advertisements (LSAs) to construct a shared view of the topology. So, fibbing creates false routing messages as wanted by the central control, lets routing protocols transmit them through the network, router than computes the routes and stores them in the routing tables, and forwards the traffic according to those routes.

Although it cannot provide the flexibility enabled by SDN, as stated in the "Opportunities and research challenges of hybrid software defined networks", fibbing techniques and algorithms can help "during the transition by providing access to the FIBs of legacy routers to any SDN controller" [36].

E. Combining vertical and horizontal separation of the network

One of the basic postulates of SDN networks is to create a simple, "dumb" hardware in the infrastructure layer, which is responsible for forwarding packets according to orders from higher instances. It is true that the horizontal separation of data and control plane achieved a significant simplification, but SDN switches still have to examine a large number of bits from packet headers to implement a process of matching [18] even in the core network.

One of the proposed solutions is the horizontal separation of the network modeled on the one in MPLS networks [37]. It is possible to have devices on the edges of the network that will be different from those in the core which will forward only on the basis of certain minimum information (such as labels in MPLS networks). In this case, the devices in the core network will truly be able to meet the requirement of simplicity and speed up the process of forwarding because the number of bits that are being tested will be reduced a dozen times.

On the other hand, such an approach would certainly complicate things in the sense that there should be different core edge devices, as well as various edge and core network elements controllers. More specifically, there should be two versions of the OpenFlow protocol. Furthermore, the interface between the edge and core devices would have to be precisely defined in the sense of mapping the edge context to the core (and vice versa) on how to manipulate specific packets.

V. UTILIZING SDN FOR CONCRETE SOLUTIONS

SDN provides three main things to the administrators of the network. Network-wide views (1) of both topology and traffic. The ability (2) to satisfy network level objectives including load balance, QoS, security, and other high-level goals in a dynamic way by controlling the network from a high-level program. Direct control (3) from a logically centralized controller by allowing writing of control program that directly affects the data plane rather than having to configure each device individually and guess what might happen.

SDN limits the consumption of power in data centers [23], reduces the cost of equipment and complexity of network configuration and management in Wireless networks [38].

If we create application programming interface (API) between control and data plane, we will ensure independent evolving. However, software written components of control plane will evolve faster than forwarding elements of data plane that are usually built from highly specialized application-specific integrated circuits (ASICs) [33].

From the perspective of users who do not have adequate technical knowledge, but want to adapt to the performance of their private network, SDN will enable the use of applications like RENEMA [39] that simplify network management, and at the same time will hide the low-level details of the configuration. Regarding routing process, SDN ensures more control over the logic of forwarding decisions e.g. shortest path based on energy consumption [40]. For enterprise networks, SDN gives the opportunity to write security applications e.g. applications that manage network access control [41]. In research networks, the separation of data and control allows to virtualize the network, so that research networks and experimental protocols can co-exist with production networks on the same underlying network hardware.

VI. CONCLUSION

Bringing things back to beginning of the paper and provided definition of network management, it is obvious that SDN is capable to bring improvements to each and every aspect of network management to all types of users, from residential, across enterprise networks, to ISPs networks. A large number of applications already in use, and many more that are being developed, as well as the research results that were conducted for SND on behavioral and programming abstractions, testing, and verification, and extensibility show that it has a great potential to be used for many traditional network architectures problem solving. At this point in time it is clear that the benefits that the concept of SDN brings should not be called into question. The real question would be whether all the fuss about the implementation of SDN is justified, and how much will we gain in accordance with the efforts and resources invested in the implementation of these networks. One must also be aware of the fact that the SDN is still going through a lot of research and testing and that there is still a lot of work to be done to fully solving all detected issues and standardize the solutions.

REFERENCES

1. J. Ding, "Advances in Network Management," Taylor&Francis Group LLC, 2010.g. , pp. 43.
2. Juniper Networks, White paper, "What's behind network downtime?, proactive steps to reduce human error and improve availability of Networks", 2008. [Online]. Available: http://www-05.ibm.com/uk/juniper/pdf/200249.pdf
3. Open Networking Foundation, White paper, "Software-defined networking: The new norm for networks," Palo Alto, CA, USA, April 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn- resources/white-papers/wp-sdn-newnorm.pdf
4. Thomas D. Nadeau and Ken Gray, SDN: Software Defined Networks, First Edition, O'Reilly Media, August 2013

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

5. D. Kreutz et al., "Software-defined networking: A comprehensive survey", Proceedings of the IEEE, vol. 103, issue 1, pp. 14-76, January 2015.
6. N. Feamster , J. Rexford, and E. Zegura, "The road to SDN," in ACM SIGCOMM Computer Communication Review, Volume 44 Issue 2, pp. 87-98, April 2014.
7. Active Networks home page (DARPA funded program), "Upcoming events and related research efforts in active networks," [Online]. Available: http://www.sds.lcs.mit.edu/darpa-activenet/
8. D.Tennenhouse, J. Smith, D. Sincoskie, D. Wetherall, and G. Minden,,"A survey of active network research," IEEE Communications Magazine, vol. 35, issue: 1, pp. 80 - 86, January 1997.
9. G. Wright and W. Stevens, "TCP/IP illustrated," vol. 2, chapter 20, June 1995.
10. [S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," in SIGCOMM '13 Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 3-14, Hong Kong, China, August 2013.
11. N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in FDNA '04 Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture, pp. 5-12, Portland, Oregon, USA, August 2004.
12. M. Caesar et al., "Design and implementation of a routing control platform," in NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 15-28, May 2005.
13. C. E. Rothenberg et al., "Revisiting routing control platforms with the eyes and muscles of software-defined networking," in HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks, pp. 13-18, Helsinki, Finland, August 2012.
14. Greenberg et al., "A clean slate 4D approach to network control and management," in ACM SIGCOMM Computer Communication Review Homepage archive, vol. 35, issue 5, pp. 41-54, October 2005.
15. Martin Casado et al., "Ethane: Taking control of the enterprise," in SIGCOMM '07 Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, pp 1-12, Kyoto, Japan, August 2007.
16. M. Casado et al., "SANE: A protection architecture for enterprise Networks", in USENIX-SS'06 Proceedings of the 15th conference on USENIX Security Symposium, vol. 15, Article No. 10, Vancouver, B.C., Canada, July 31 - August 04, 2006.
17. W. Xia, Y.,Wen, Y., C.H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," IEEE Commununication Surveys & Tutorials, vol. 17, issue: 1, pp. 27-51, June 2014.
18. Open Networking Foundation, "OpenFlow switch specification", Version 1.5.1 (Protocol version 0x06), ONF TS-025, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/ sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf
19. N. Gude et al., "NOX: Towards an Operating System for Networks", in ACM SIGCOMM Computer Communication Review, vol. 38, issue 3, pp. 105-110, July 2008.
20. J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in IEEE 15th International Symposium on World of Wireless, Mobile and Multimedia Networks WoW-MoM, pp. 1-6, October 2014.
21. T. Koponen et al., "Onix: A Distributed Control Platform for Large-scale Production Networks," in OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation, pp. 351-364, Vancouver, BC, Canada, October 2010.
22. P. Songa, Y. Liua, C. Liua, and D. Qian., "ParaFlow: Fine-grained parallel SDN controller for large-scale networks," Journal of Network and Computer Applications, vol. 87, issue C, pp. 46-59, June 2017.
23. Heller et. al, "ElasticTree: Saving energy in data center networks," in NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation, pp. 17, San Jose, California, April 2010.
24. Duan, Q., "Network-as-a-service in software-defined networks for end-to-end qos provisioning," in Wireless and Optical Communication Conference (WOCC), pp. 1–5, May 2014.
25. A.Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying NOX to the datacenter," in Proceedings of the 8th ACM Workshop on Hot Topics in Networks, pp, 1-6, New York City, New York, 2009.
26. Network heresy, "Tales of the network reformation," [Online]. Available: https://networkheresy.com/2011/06/08/the-scaling-implications-of-sdn/
27. Curtis et al., "DevoFlow: Scaling flow management for high-performance networks," in SIGCOMM '11 Proceedings of the ACM SIGCOMM 2011 conference, pp. 254-265, Toronto, Ontario, Canada, August 2011.
28. T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in IMC '10 Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 267-280, Melbourne, Australia, November 2010.
29. J. McCauley, A. Panda, M. Casado, T. Koponen, and S. Shenker, "Extending SDN to large-scale networks," [Online]. Available: http://www.cs.columbia.edu/~lierranli/coms6998-10SDNFall2014/papers /Xbar-ONS2013.pdf
30. K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in Network Operations and Management Symposium (NOMS) IEEE, pp. 1-4, June 2014.
31. F. Benamrane, M. Ben Mamoun, and R. Benaini, "An East-West interface for distributed SDN control plane: Implementation and evaluation," Computers and Electrical Engineering, vol. 57, issue C, pp. 162-175, January 2017.
32. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in INM/WREN'10 Proceedings of the 2010 internet network management conference on Research on enterprise networking, pp. 3-3, San Jose, California, 2010.
33. S. H. Yaganeh, A. Tootoonchian, and Y. Ganjali, "On the scalability of software-defined networking," IEEE Communications Magazine, vol. 51, issue: 2, pp. 136-141, February 2013.
34. S. Vissicchio, L. Vanbever, and J. Rexford, "Sweet little lies: Fake topologies for flexible routing," in HotNets-XIII Proceedings of the 13th ACM Workshop on Hot Topics in Networks, pp. 3, Los Angeles, CA, USA, October 2014.
35. S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in SIGCOMM '15 Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, pp. 43-56, London, United Kingdom, August 2015.
36. S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," ACM SIGCOMM CCR, vol. 44, no. 2, pp. 70-75, 2014.
37. M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A retrospective on evolving SDN," in HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks, pp. 85-90, Helsinki, Finland, August 2012.
38. K. K. Yap et al., "OpenRoads: Empowering research in mobile networks," in Newsletter ACM SIGCOMM Computer Communication Review, vol. 40, issue 1, pp. 125-126, January 2010.
39. R. Moyano, D. Cambronero, and L. Triana, "A user-centric SDN management architecture for NFV-based residential networks," Computer Standards & Interfaces, vol. 54, issue P4, pp. 279-292, in press.
40. J. Wang, Y. Miaob, P. Zhoub, M. S. Hossainc, and Sk Md M. Rahmand, "A software defined network routing in wireless multihop network," Journal of Network and Computer Applications, vol. 85, issue C, pp. 76-83, May 2017.
41. H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FlowGuard: Building Robust Firewalls for Software-defined Networks," in HotSDN '14 Proceedings of the third workshop on Hot topics in software defined networking, pp. 97-102, Chicago, Illinois, USA, August 2014.

AUTHORS PROFILE

M.Sc. Muhamed Begović, works as Teaching Assistant at Faculty of Traffic and Communications, University of Sarajevo, and is a PhD Candidate at Department of Communication Technologies. Main fields of interest: Computer Networks, Networks Architecture, Software Defined Networks.

Prof. Himzo Bajrić, works as Associate Professor, Faculty of Traffic and Communications, University of Sarajevo. Worked at BH Telecom dd Sarajevo in the position of Executive directorate for technology and development services. Main fields of interest: Computer Communications (Networks), Computer Security and Reliability, Computing in Mathematics, Natural Science, Engineering and Medicine.