Gradus Vol 7, No 2 (2020) 230-238
ISSN 2064-8014

REVIEW ON NETWORKS DEFINED BY SOFTWARE

Sara I. Boucetta [1,2*] *and Zsolt Csaba Johanyák* [2]

[1] Department of Information Technology, Institute of Information Technology, University of Miskolc, Hungary
[2] Department of Information Technology, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary

Keywords:
software defined networks
network automation
SDN architecture
OpenFlow
API

Article history:
Received 24 Apr 2020
Revised 2 May 2020
Accepted 10 May 2020

Abstract
Heretofore, most network equipment had to be configured individually by connecting manually into it. This approach is time consuming for large networks and prone to human errors. The Software Defined Networking paradigm defines several standards and protocols in order to read the network states and act on its configuration from distant servers. These protocols authorize a reconfiguration of the network in a centralized way by the use of transactions that acts on one or more devices. In general, transactions are implemented as APIs for use by third-party programs and on software components separate from the orchestrator called controllers for more modularity. Nowadays, SDN receives a lot of interest from researchers and manufacturers aiming for the modernization of the networks especially with the emergence of the IoT, 5G and WAN technologies

1 Introduction

A computer networks is formed from a set of interconnected equipment to establish communication. Throughout the years, several networking devices have been developed and deployed. Repeaters and bridges, routers and switches are used in multiple network environments, filtering and forwarding packets over the network toward their destination.

Software-defined networking (SDN) is a technology that tends to move toward cloud computing, created to simplify network management and enabling efficient network configuration by the use of programming in order to improve network performance and monitoring.

The architecture of traditional networks is static, decentralized and complex while modern networks need flexibility as well as simple and quick troubleshooting. SDN aim to concentrate network intelligence in one network component, the SDN controller.

2 SDN Fundamental characteristics

SDN is characterized by the fundamental features [1] presented in the following subsections.

2.1 Plane separation

The separation is done by unbundling the forwarding process of network packets (data plane) from the routing process (control plane). The fundamental actions performed by the forwarding plane can be described by how it deals with arriving packets (forward, drop, consume, modify or replicate an incoming packet), relying on some packets characteristics like MAC address, IP address, and VLAN ID. The device should determine the exact output port by the help of a lookup in the address table of the hardware. A packet could be dropped due to buffer overflow conditions or due to specific filtering rule resulting from a QoS rate-limiting function. Furthermore, packets requiring processing

* Corresponding author: boucetta@iit.uni-miskolc.hu

by the control or management planes are extracted and passed to the appropriate plane. Finally, there is a special instance of forwarding related to multicast, where the incoming packet should be replicated before forwarding the various copies throughout the different output ports. The logic and algorithms used to program the forwarding plane are implemented in the control plane. They depend upon global knowledge of the network architecture. The control plane determines how the forwarding tables and logic in the data plane should be configured. In Fig. 1, the control plane is moved off from the switching device into a centralized controller.

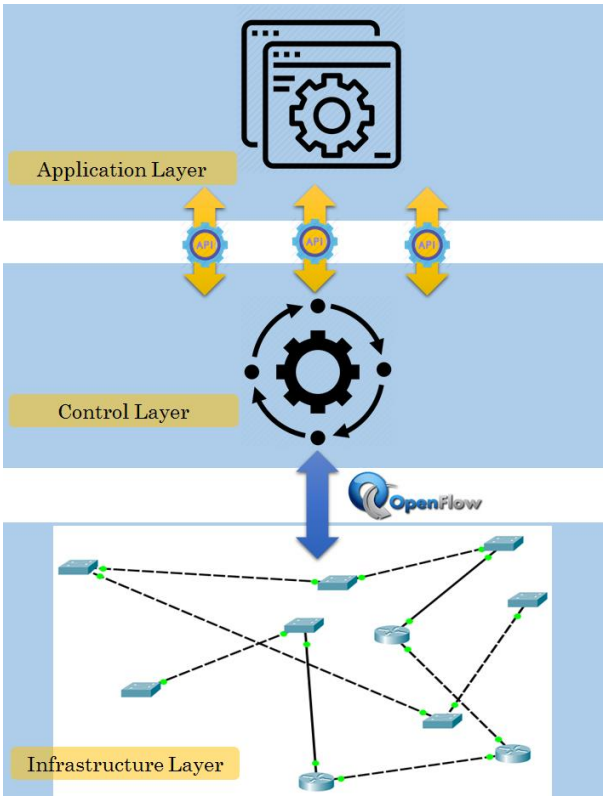

Figure 1. A layered view of the SDN network planes

2.2 Simple device and centralized control

Network devices are simplified and controlled by a centralized running management system and control software. The manufacturer's built in software is moved from the device towards a centralized controller, which manages the network based on high level rules and provides primitive instructions when necessary to allow devices to make faster decisions about how to handle incoming packets.

2.3 Network Automation and Virtualization

SDN stem from the concept of forwarding, distributed state, and configuration. Resulting from the decomposition of the network control issue faced by networks when trying to simplify abstractions, there is three fundamental conceptualizations that form the foundation of SDN:

- ***The forwarding*** concept helps the network administrator to configure the forwarding rules of the network devices without needing to master any vendor-related configuration. Thus, the programming languages used must be a common feature to maximize the effectiveness of network hardware.
- ***The distributed state*** concept provides the network administrator with a global network view hiding the heterogeneous nature of the network devices, each with its own status, collaborating to solve networking issues.

- ***The configuration*** or ***specification concept***, allows to specify the required purposes of the whole network without struggling with their implementation details in the physical network.

The centralized SDN controller implements an open interface on the controller to enable an automated network management. In Fig. 2, and in the perspective of Open SDN, the designation of northbound and southbound are both used in order to discern whether the interface applies to the applications or to the network devices.

The northbound interface enables the communication between an element of the network with an upper-level component.

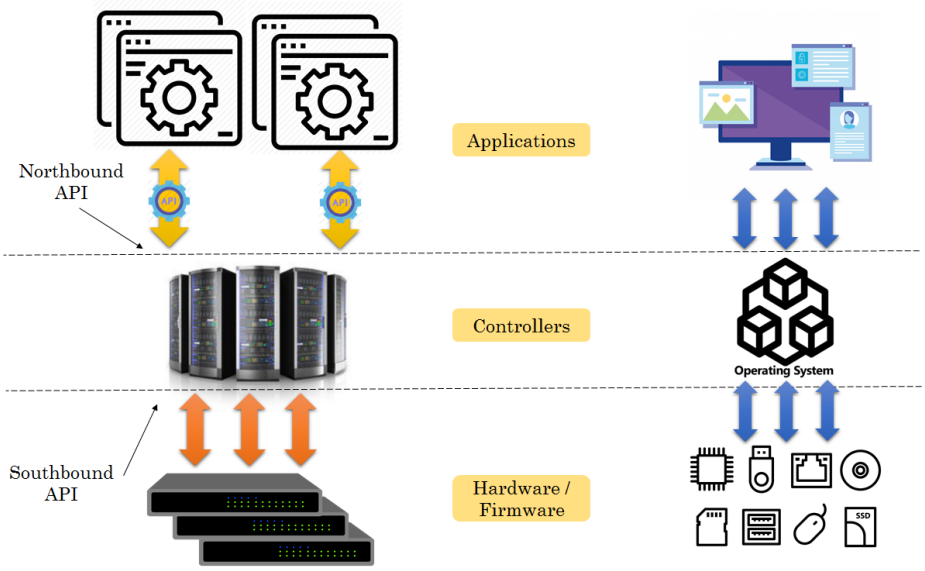

Figure 2. SDN Stack VS operating systems abstraction layers

On the other hand, the southbound interface allows an upper-network element to establish communication with a lower-level element. OpenFlow is an example of a southbound API interface that the SDN controller uses to program and manage the network devices. The controller allows software applications to be plugged into it by enabling a northbound API, thus enabling software to provide the algorithms and protocols that run efficiently on the network, the northbound API of the controller is relied on to provide an impression of the network topology and devices.

There are three main advantages of the northbound API:

- It transforms network programming language syntax to one, that is common to and understood by developers.
- It creates an abstraction of the network topology and the network layer enabling the programmer of the application to handle the network as a complete entity instead of handling every single node separately.
- It creates an abstraction of the network protocols, hiding the technicalities of OpenFlow or BGP from the application developer by using this approach, applications could be developed to run over a wide range of heterogenous vendors equipment regardless of their implementation details. As a result, the possibility to implement network virtualization disassociating the network services from the physical structure of the network. These services are presented to the host devices in a manner that they ignore the network resources that they are using are virtual instead of the physical ones for which they were originally created.

2.4 Openness

One of Open SDN principal characteristics is related to the fact that its interfaces have to remain standard, thoroughly documented, and not licensed. Software should be given enough control in order to play around and control various control plane possibilities offered by the defined

APIs. Consequently, the pace at the development and deployment of the network technology is considerably enhanced, involving an outstanding and an accelerated technological advancement in the structure and operation of networks. Furthermore, by supporting research and experimentation, open interfaces enable equipment from diverse manufacturers to interoperate. producing a competitive framework reducing network equipment costs for consumers.

3 Functioning of SDN

SDN devices carry forwarding processes to determine the action to perform on each single incoming packet. They also hold the data leading to those forwarding behaviors. The data is depicted by the flows specified by the controller.

A flow characterizes a set of packets transferred between two endpoints or a set of them. It is unidirectional in those packets moving between the same two recipients in the reverse direction forming each an independent stream [2].

In Fig. 3, the flows are depicted on a device as a flow entry. An endpoint can be characterized by an IP address, a TCP/UDP port pairs, VLAN endpoints, layer three tunnel endpoints, and input port... etc. A set of rules resolved by the device details the behavior of all packets belonging to that flow.

A flow table resides on the network device and is composed of a set of flow items and the actions to execute when a packet corresponding to that flow reaches the device. When a packet is received by the SDN equipment, it looks up its flow tables seeking for a correspondence. Flow tables are built when the flow rules are downloaded to the SDN equipment by the SDN controller [3]. If a correspondence is found by the SDN device, the correlated configured action is executed, involving mainly forwarding the packet to the corresponding port. Otherwise, the device can either reject the packet or send it up to the SDN controller, determined by its configuration and OpenFlow's version.

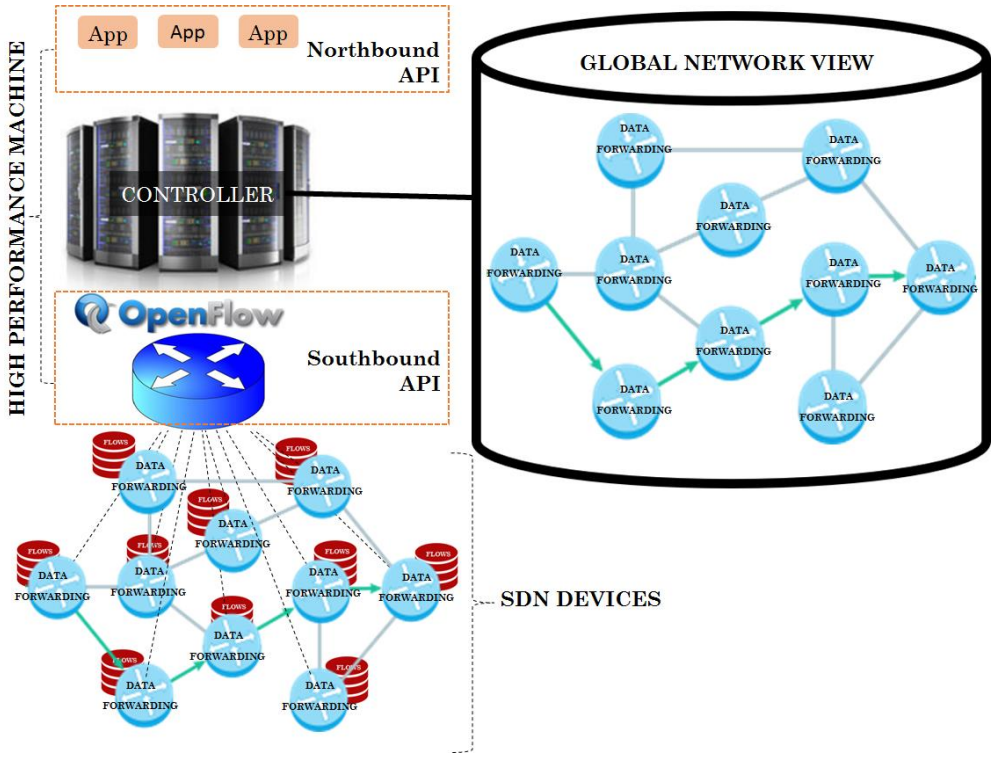

Figure 3. SDN operation overview and Controller to device communication

The SDN controller:

- is in charge of the abstraction of the network composed by SDN devices, it commands and offers an image of the existing network resources to the SDN applications executed above.

- enables the definition of the flows on devices by the SDN application and helps the application to respond to packets received by the SDN devices.
- keeps a view of the entire network under his control; therefore, it can compute optimal forwarding rules for the entire network in a deterministic and foreseen way. These computations are processed by an efficient computer with a high CPU and memory capacity usually granted to the network devices.

SDN applications are developed above the controller, they are key-part of network layers two and three of the OSI model, they interface with the controller in order to configure proactive flows on the devices and to process packets transmitted to the controller. Proactive flows are set by the application when it starts up, and the flows will remain unless some configuration change is operated, this proactive flow is designated as static flow. A second kind of proactive flow is described by when the controller decides to modify a flow based on a certain amount of traffic load managed by the network device.

There are flows that are set upon the receipt of a packet by the controller (reactive flows). When new incoming packets are received by the controller, the SDN application will define how the controller should react in response to the received packet, and when necessary, will configure new rules on the device allowing it to react internally if it receives a packet that belong to that flow.

This allows to program applications that implement forwarding, routing, multi path, and access control functions, etc. There exist certain reactive flows that are set or modified following the reception of a different event than the controller's packets.

4 SDN Devices

An SDN device is composed of an API to communicate with the SDN controller, an abstraction layer, and a packet processing function. When the switching device is virtual (Fig. 4), the packet processing function corresponds to the packet processing software

On the other hand, in the case of a physical switching device (Fig. 5), the packet processing function is embedded within the hardware for packet processing logic.

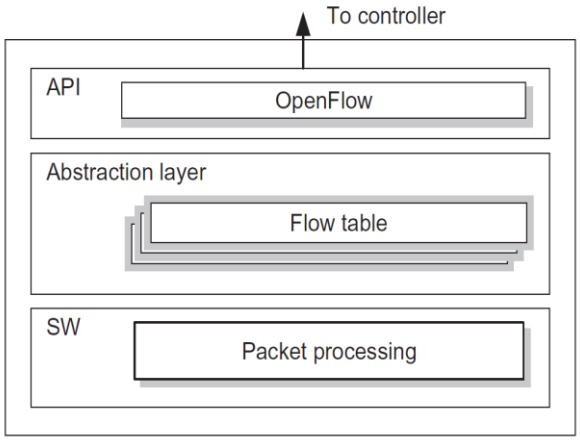

Figure 4. SDN software switch anatomy [1]

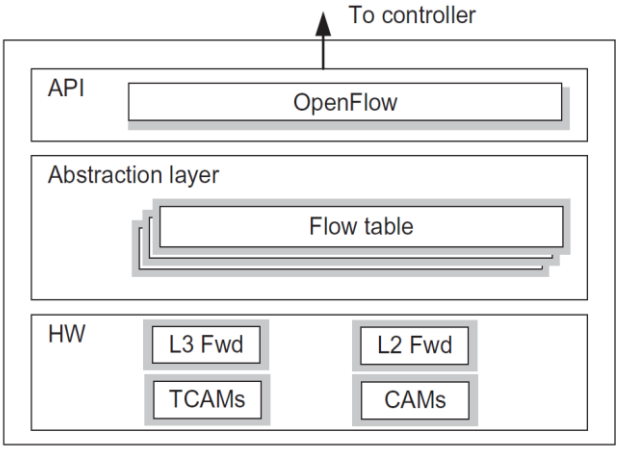

Figure 5. SDN hardware switch anatomy [1]

The abstraction layer holds at least one flow table, the packet processing logic is composed by a set of processes that are executed in response to the evaluation of the incoming packets by identifying the highest priority match.

- When a correspondence is found, the packet is treated locally, unless its explicit destination is the controller.
- If the matching process fails the packet is either dropped or forwarded to the controller for further processing.

4.1 Flow Tables

Flow tables are the principal data structures composing an SDN device. They allow the device to process incoming packets and decide according to the content of the packet received. Flow tables are composed of several prioritized flow entries, consisted of two components, **match fields** and **actions**.

Match fields are used to compare against incoming packets. Upon their reception they are compared against the match fields in priority order and the first full match is selected.

Actions are the instructions that the network device should run if an incoming packet successfully matches the match fields depicted for this flow entry.

4.2 SDN Software Switches

The easiest way to create SDN devices is to implement them in software. Indeed, flow tables, flow entries, and match fields can be represented as software data structures like sorted arrays or hash tables. If two software SDN devices are produced by two independent development companies they will still behave similarly, while it is hard to obtain the same result if it was for two different hardware implementations. However, devices based on software are mostly **slower** and **less effective** compared to those based on hardware. Hardware solutions decrease latency and increase the communication throughput.

- Since wildcard masks are used during the corresponding process, some issues may be faced for standard hash tables. The packet processing function uses advanced software logic to implement effective match field lookups.
- Standard software device implementations are not limited by any processing power or memory size.
- Software SDN device implementations can be found in software-based network devices, like the hypervisors of a virtualization system.

Software SDN device implementations advanced considerably, as a result, greater consistency in software SDN device performance have been observed. Since they are more flexible, perplex functionalities can be easily implemented, which result in a wider set of actions expanding the operating range comparing to hardware-based devices.

4.3 SDN Hardware Devices

Hardware implementations of SDN devices are expected to run faster than their software counterparts as they are most suitable for performance-sensitive environments like data centers and network cores.

In this kind of devices, packet processing logic is substituted by a dedicated hardware that contains the layer two and three forwarding tables, usually built using Content-Addressable Memories (CAMs) and Ternary Content-Addressable Memories (TCAMs).

The layer three forwarding table is used for taking IP-level routing actions by comparing the destination IP address with the entries of the table, and, based on the match, takes the most suitable routing decision. While the layer two forwarding table is used for taking MAC-level forwarding actions by matching destination's MAC address with the entries present in the table and applying the necessary forwarding operation. This option enables the device to both match packets and process them at a high rate. However, it also raises a set of challenges to the SDN device developer.

5 SDN Controller

The SDN controller has to maintain a global view of the network and to control all the SDN devices within its network infrastructure. It provides a northbound API for applications and implements the policy decisions regarding routing, forwarding, redirecting, load balancing, etc.

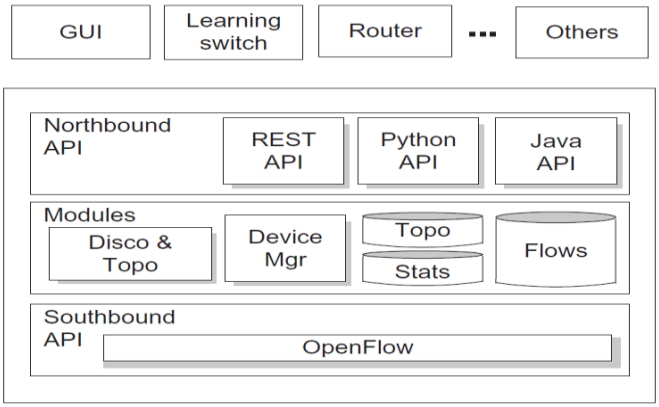

Figure 6. SDN controller architecture [1]

Fig. 6 describes the modules forming the core functionality of the controller, the northbound and the southbound API, and some applications that might run on the controller. the southbound API is used for communication with the SDN devices, OpenFlow in the case of Open SDN and other alternatives like BGP in the case of some SDN solutions[6]. However, the lack of a common standard led to the implementation of northbound interfaces in many disparate forms to overcome this problem.

5.1 SDN Controller Core Modules

The controller hides the implementation details of the SDN controller-to-device protocol communication to enable the applications above to communicate with SDN devices regardless of their configuration.

The fundamental features of the SDN controller include:

- **End-user Device Discovery**: Discovery of end-user devices, like laptops, desktops, printers, mobile devices, etc.
- **Network Device Discovery**: Discovery of network devices which reside within the infrastructure of the network, such as switches, routers, and wireless access points.
- **Network Device Topology Management**: Maintain information regarding the interconnection details between the network devices, and to the end-user devices to which they are directly linked.
- **Flow Management**: keeps a database of the flows being processed by the controller and coordinate between the devices to ensure synchronization of the device local flow entries with that database.

Device and topology discovery and tracking, flow management, device management and statistics tracking represent the core functions of the controller. They are implemented by a set of internal modules into the controller. Their role is to maintain local databases holding the current image of the network topology and its related statistics.

The controller maintains the topology by learning of the existence of switches (SDN devices) and end-user devices and tracking the connectivity between them. It also maintains a flow cache, which describes the flow tables on the different switches it controls and locally keeps up-to-date per-flow statistics gathered from these devices. The controller can be configured such as the functions are implemented via pluggable modules as a result, features set on the controller are adjusted according to the needs of the current network.

5.2 SDN Controller Interfaces

The northbound API is an essential feature provided by the SDN controller. It is a low-level interface providing access to the network devices in a similar and undeviating manner. Therefore, the application is aware of the existence of individual devices, without knowing the details about their configuration and their differences. The controller may also provide high-level APIs that provide an abstraction of the network itself. Consequently, the application developer does not have to care about individual devices, but should deal with the network as a whole.

The controller informs the application of events that happen in the network. Events are communicated from the controller to the application. They can be related to an individual packet received by the controller or to a state change in the network topology, like the loss of a networking link.

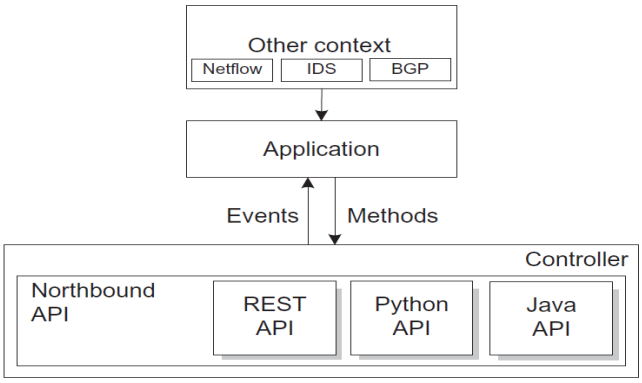

Figure 7. SDN controller northbound API [1]

Several mechanisms are used by the SDN applications in order to influence the behavior of the network. They are executed following an incoming incident. As a result, the incoming packets can be deleted, changed and/or dispatched to their destination. These applications can also affect the flow management by adding new flows, deleting obsolete flow entries or modifying them according to network evolution. These mechanisms can be invoked by the applications independently from any incoming event into the controller.

6 SDN Applications

SDN applications run on the SDN controller, interfacing to the network via the controller's southbound API. They are responsible for managing the flow entries that are programmed on the network devices, using the controller's API to manage flows. Through this API the applications are able to:

- configure the flows to route packets through the best path between two endpoints;
- balance traffic loads across multiple paths or destined to a set of endpoints;
- react to changes in the network topology such as link failures and the addition of new devices and paths,
- redirect traffic for purposes of inspection, authentication, segregation, and similar security-related tasks.

7 Conclusions and further research plans

SDN solutions have a wide application area in computer networks, IoT, 5G technology [5], etc. They can offer a more efficient environment for flow control which is less vendor dependent than the current devices. SDN ensures the flexibility needed by the current continuously changing environment. In this paper, we presented the fundamental characteristics and functioning of SDN as

well as the SDN architecture and its main features were described. At the end some application possibilities were also outlined briefly.

Future research will be focused on the investigation of the application possibilities of fuzzy logic-based control solutions [7][8] or fuzzy cognitive maps [9] in SDN control problems.

Acknowledgment

This research was carried out as part of the EFOP-3.6.1-16-2016-00011 "Younger and Renewing University – Innovative Knowledge City – institutional development of the University of Miskolc aiming at intelligent specialisation" project implemented in the framework of the Szechenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

References

[1] Goransson, P., Black, C. and Culver, T.: Software Defined Networks, 2nd Edition, 20th October 2016, Pc. 436, E-ISBN: 9780128045794

[2] Jammal, M., Singh, T., Shami, A., Asal, R., Li, Y.: Software defined networking: State of the art and research challenges. Computer Networks. 72, 74-98 ,2014

[3] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig.: Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[4] Dejan Vukobratovic, Dusan Jakovetic, Vitaly Skachek, Dragana Bajovic, Dino Sejdinovic, Güneş Karabulut Kurt, Camilla Hollanti, Ingo Fischer.: CONDENSE: A Reconfigurable Knowledge Acquisition Architecture for Future 5G IoT, Access IEEE, vol. 4, pp. 3360-3378, 2016.

[5] Junfeng Xie, Renchao Xie, Tao Huang, Jiang Liu, Yunjie Liu, ICICD: An Efficient Content Distribution Architecture in Mobile Cellular Network, Access IEEE, vol. 5, pp. 3205-3215, 2017.

[6] Qiang He, Xingwei Wang, Min Huang.: OpenFlow-based low-overhead and high-accuracy SDN measurement framework, Transactions on Emerging Telecommunications Technologies, vol. 29, pp. e3263, 2018.

[7] R.E. Precup and M. L. Tomescu, Stable fuzzy logic control of a general class of chaotic systems, Neural Computing and Applications, vol. 26, no. 3, pp. 541-550, Apr. 2015.

[8] S. Blažič, I. Škrjanc, D. Matko (2014): A robust fuzzy adaptive law for evolving control systems, Evolving Systems, vol. 5, no. 1, pp. 3-10, Mar. 2014.

[9] J. Vaščák and M. Rutrich, Path planning in dynamic environment using fuzzy cognitive maps, Proceedings of 6th International Symposium on Applied Machine Intelligence and Informatics (SAMI 2008), Herľany, Slovakia, 2008, pp. 5-9.