ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20160928094756
Subject review

Implementation of Software-Defined Networks Using Open-Source Environment

Petar ČISAR, Dragan ERLENVAJN, Sanja MARAVIĆ ČISAR

Abstract: The paper gives an overview and analysis of software-defined technologies in the context of the actual problems of conventional computer networks. The virtualization process lies at the basis of software-defined network architecture. The basic concepts of software-defined networks are elaborated, as well as their implementation and configuration in cloud environment using OpenStack open-source software with Microsoft Hyper-V hypervisor. The practical part of this research showed that the implementation of a software-defined network into an existing standard network is relatively simple to realize, remaining all the time in open-source framework. All existing network functionalities were held, and it is also possible to provide additional network operations such as traffic monitoring in a completely open-source environment.

Keywords: data center; hypervisor; IaaS; network instances; OpenStack; OpenFlow; software-defined network; virtualization

1 INTRODUCTION

The world is witnessing the rapid development of solutions and services based on infrastructure of cloud virtualized systems. The emphasis is increasingly moving from the lowest physical and infrastructural level to the highest application level. Software has become the central point of future IT services and applications. In this sense, there is a very strong need for interoperability and automation solutions based on cloud virtual systems. The software-defined network (SDN) and the overlay concept were devised in order to adapt the network to the global virtualization, as well as the necessary advanced technologies in software-defined data centres. SDN has been proposed as a replacement for today's MPLS (Multi Protocol Label Switching) networks in which, for now, none of the big service providers do use mechanisms for traffic engineering because of the instability and the high complexity of management. For this reason, it is necessary to better understand this technology and possibly define a solution that will be acceptable, as seen from the technological and financial aspects, and also contribute to satisfaction of service customers. Virtualization is a combination of software and hardware that allows a single physical computer simultaneously runs multiple operating systems or more equal, which share common resources. The system itself is divided into several separate virtual units that behave as independent systems. Using virtualization, the usage level of hardware capabilities is able to significantly grow. Thus, there is a tendency that a large number of expensive servers with a single operating system, application and low level of use, is being replaced by a smaller number of better utilized servers which simultaneously work on multiple operating systems and applications. The basic concepts of SDN in the context of the future vision of the Internet were the subject of research in many works [1-5].

The goal of this research is to realize the virtualization of existing hardware, networks, their interconnections, and enable the central management of the newly created configuration, using the currently available solutions in the market. The research should lead to an increase in the utilization of the relatively limited hardware resources, as well as increase in system reliability and performance, and providing fast, efficient and easy management. While this paper presents the visualization applied to a specific case, in general terms, this research formulates guidelines for the purpose of hardware virtualization, using VMware and Microsoft Hyper-V hypervisors for implementation of open-source OpenStack tool for managing Infrastructure as a Service (IaaS) in cloud environment with instances connected by software-defined networks.

This work outlines the basic concepts of software-defined networks, as well as their implementation and configuration in cloud environment using OpenStack open-source software on Microsoft Hyper-V hypervisor. OpenStack needs a hypervisor for the virtualization of instances and SDN network controller, and its integration will therefore be realized with Microsoft Hyper-V solution. Instances and networks for different purposes were created in order to examine realistic environment and ascertain possible weaknesses and improvements that come with this new concept.

The paper consists of seven sections. The introduction offers a terminological basis for software-defined technologies and describes the problem of conventional networks, then in the second section an overview of software-defined data centres is presented. The software-defined networks are in general described in the third section, followed by the fourth section dealing with the SDN architecture. The fifth and sixth sections are devoted to the concepts of OpenFlow and OpenStack. The implementation and analysis of SDN in OpenStack environment are given in the seventh section. Finally, the paper closes with the conclusion based on the analysed cases.

2 SOFTWARE-DEFINED DATA CENTERS

In the software-defined data centres, everything is open - hardware, software, applications. A key question regarding this approach is the following: Is inexpensive hardware and open software sufficiently reliable to be a data centre?

At a time of turbulent technological changes, the construction of a modern data centre represents a serious undertaking and any chosen solution may prove to be wrong, already after one or two years. The architecture of data centres today is largely conditioned by the rapid expansion of Internet traffic and the different cloud concepts, however, anyone can hardly predict that such

architecture will be sufficient for the next 10 to 15 years, which is the normal life of a data centre.

Data centre managers are in a situation with a number of contradictory requirements placed upon them. On the one hand, the infrastructure (cables, air conditioners, power supplies) must be able to adapt to the growing needs for capacity, and on the other, budgets are limited and cost reduction is always the main goal. The greatest challenge is how to build a data centre with a limited budget that will be modular and flexible to meet current technological requirements, which will be energy efficient, and to ensure business continuity in all possible conditions, but also be ready for the new technology.

Existing data centres are characterized by complex and heterogeneous architecture, starting with the server architecture (blade systems, integrated systems), network equipment (switches, routers) to the cables (copper, optical) and racks (server, open frame, telecom-providers). In the centre of it, all is the hardware and physical infrastructure, and the size of the data centre is practically limited by the size of the space where the infrastructure is located. The applied standards for the greenfield data centre define minimum requirements relating to the infrastructure components (TIA-942 standard defined by ANSI), while for the technology in data centres, there are no such standards, except to form technological trends [6].

Resource sharing and virtualization have long been established concepts when configuring servers and storages proved to be successful both in terms of saving money, and also from the standpoint of increasing the agility and availability of the system. Virtualization allows data centres to exceed the physical limits of server rooms, but for one virtual server to really use the hardware resources that are located in different locations, there must be a support of network.

In addition, the amount of data that is nowadays exchanged between the servers in the data centre, is often greater than the amount exchanged between servers and clients outside the data centres, whereby the data is transmitted from the virtual to the physical environment and vice versa. SDN and the overlay concept were designed in order to adapt to every form of virtualization, but it is not all that will be seen of advanced technology in data centres.

2.1 Megadata Centres

Judging by what large IT companies do, significant changes are expected in the infrastructure of data centres. Specifically, these companies have started their migration process of megadata centres on software-led infrastructure (SLI), which includes a complete virtualization and automatic allocation of resources within and between data centres. In the background of all is the new infrastructure that will be based on open hardware, open software and the platform that will orchestrate all the resources and optimize them according to current needs. Openness of hardware and software, significantly reduces the cost of IT infrastructure (up to 50%), due to the fact that the purchase of general purpose hardware is cheaper, and the equipment in data centres is more homogeneous, which, in turn, reduces maintenance costs. Instead of expensive branded switches, megadata centres rely on the so-called white box switches, which have the same hardware as the switches of world famous manufacturers, but they do not have any software. Therefore, large companies buy equipment and make their own switches based on an open software stack [6].

The open software stack enabled virtualizing network functions such as firewall, routing or rate limiting without purchasing additional hardware. For example, Google implemented some of these functions as pipelining processes in the software, so packets sent by the virtual server go through these software processes before they reach the network adapter on the physical server. It is important to emphasize that the configuration and management of virtual machines, software switches, and even the hardware works with a central controller in one place, which can be programmed to all of these components.

High-tech data centres are expected to automatically add or release resources (scale-out and scale-in) such as CPU, memory, or disk drives, according to current needs and load of machines. In order for this to be possible in practice, all the components participating in the process (virtual servers, storages, network, applications) must be in some way identified with each other. Therefore, each SLI component in megadata centres must have a specific set of metadata to be exchanged with other components using known APIs. When, for example, for the newly created virtual server or a new application server it is needed to reserve some capacity in the network, the metadata will be used to identify the server or the application, which reserve capacity. For such a complex interaction, the open software stack is an ideal testing polygon for the work.

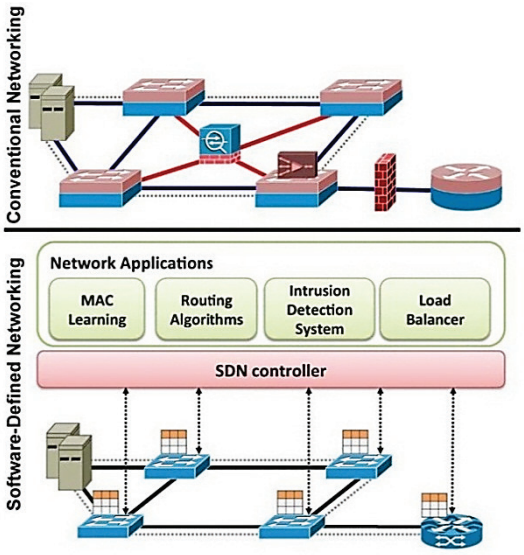

Figure 1 Conventional networking compared to the SDN [7]

3 SOFTWARE-DEFINED NETWORKS

While companies move to cloud environments, outdated network architectures stumble under the pressure of immediate access to applications and services that offer a high quality user experience. At the same time, organizations struggle with the complexity of the network settings in the data centre, which require manual adjustment of all devices. SDN represents the physical separation of the control plane in a network from data

plane, where the control plane controls the multiple devices. It presents the architecture in the development stage, which is dynamic, manageable, cost-effective and flexible, thus making it ideal for today's dynamic applications that require wide bandwidth.

This architecture separates the network control and forwarding functions, enabling the control layer to become directly programmable, and the basic architecture to be separated for the purpose of applications and network services. OpenFlow protocol is a basic element for the construction of SDN solutions.

4 SDN ARCHITECTURE

The basic characteristics of SDN architecture are: (a) directly programmable: the control part of the network is directly programmable, because it is separated from the forwarding function; (b) agility: The separation of the control part from the forwarding part allows administrators to dynamically adjust traffic flows in the network according to the needs of users for changes; (c) central management: logical part is centralized in software-based SDN controllers that manage the network, which applications "see" like one logical switch; (d) configured by programs: the SDN enables configuration, security, and optimization of network resources quickly by dynamic automated SDN programs that administrators can create themselves because these programs are not dependent on protected software; (e) based on open standards: When implemented through open standards, SDN simplifies the design and operation of the network, because the instructions are determined by SDN controllers instead of multiple devices and protocols that define network equipment manufacturers.

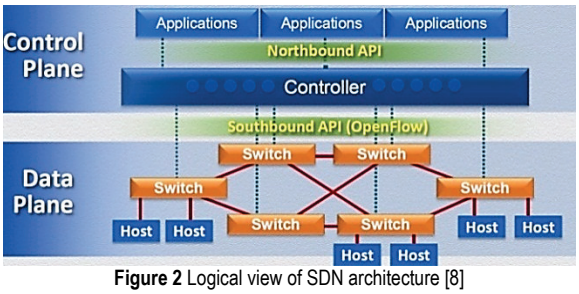

Figure 2 Logical view of SDN architecture [8]

SDN also significantly simplifies the performance of network devices because it is no longer necessary to understand and process thousands of protocols, but only to accept the instructions from SDN controller. Probably the most important characteristic is that the network operator and the administrator can programmatically configure a network simplified in this way, instead of manually entering a large number of command lines on different network devices. In addition, using a centralized intelligence of SDN controller, it is possible to change the behaviour of the network in real time and set up new applications and network services in the SDN controller, in a much shorter time. A part of forwarding open commands is defined as a protocol called OpenFlow. OpenFlow protocol allows overall control of software that can be centralized or distributed, to drive the network hardware in order to create easily programmable overlay of current IP core network. Apart from the aggregation of network, SDN architecture supports a set of APIs that enable the implementation of common network services including routing, multicast, security, access control, bandwidth management, service quality, optimization of CPU and storage, energy usage and all forms of management policies tailored to business objectives [9].

The aim of SDN is to ensure that all control plane logical decisions are made in one central place, compared to the conventional networking, where control plane decisions are made locally and the intelligence is distributed in each switch. This centralized approach reduces the need for a number of intelligent nodes.

Programmability of computer networks is achieved by applying not only low-level machine language, such as OpenFlow, but also high-level programming languages. There are a number of programming languages for SDN and the majority of them can be applied in OpenFlow networks. Pyretic is one of the most important high-level programming languages.

5 OpenFlow

OpenFlow is the first and only standardized interface between the control and the infrastructure layer that allows forwarding traffic in SDN. It uses the concept of data flows for identifying traffic based on predefined rules, which are statically or dynamically defined in the control software. Because of its flexibility in multi-vendor networks, it can be applied in existing physical and virtual networks. Unlike traditional IP networks, which use the same path of packets through the network, network managing with OpenFlow protocol is characterized by granularity at the level of applications, users and sessions. This protocol allows direct access to the infrastructure layer and the manipulation of network devices such as switches and routers (physical or virtual). It is implemented on both sides of the interface between the network infrastructure and SDN control software.

As the OpenFlow network can be programmed on the data flow level, it provides higher-granular control that enables fast adaptation to network applications and user requirements. Today's networks based on IP protocol do not provide this option, because in this case all traffic flows between the two end systems use the same communication way, no matter their requirements are completely different, such as in the case of transfer of video and regular web content. The absence of open interfaces to forwarding level led to the characterization of today's network devices as closed. Protocol as OpenFlow is required to move the control of network from the network switches to a logical centralized control software. The protocol specifies the basic functions that can be used via external software application for programming devices that are within the forwarding level.

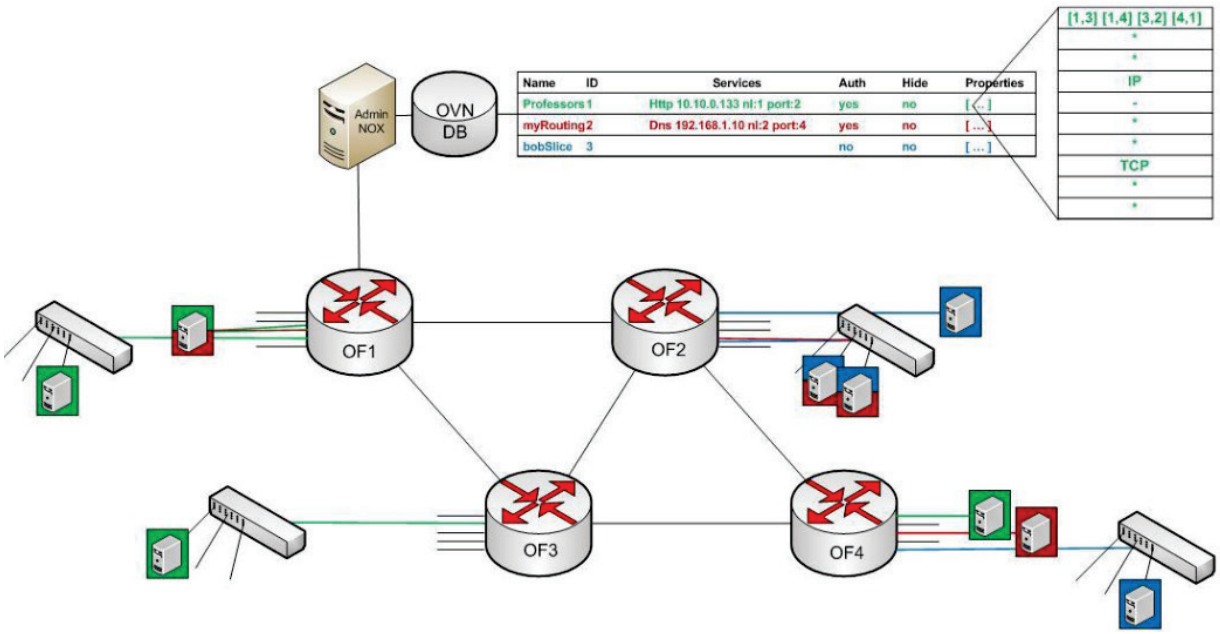

Figure 3 Example of OpenFlow network after defining the table of flows at the database of the controller [10]

5.1 OpenFlow Logical Switch

The OpenFlow logical switch consists of: (1) one or more flow tables and the group table on the basis of which the traffic is forwarded; (2) one or more OpenFlow channels to an external controller.

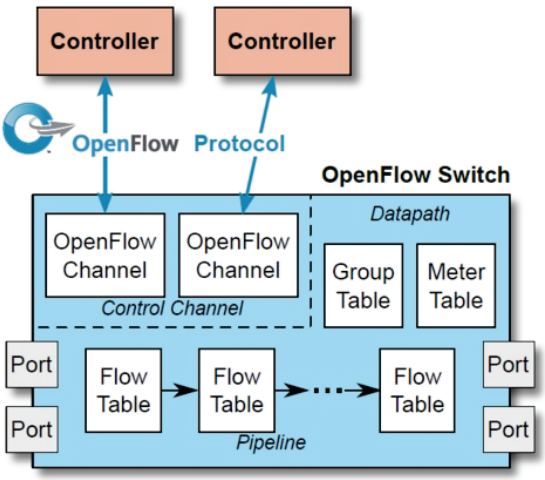

Figure 4 Architecture of the OpenFlow 1.5 switch [11]

Using the OpenFlow protocol controller changes the records of dataflow in flow tables. Each table contains a set of instructions on the basis of which the traffic is forwarded. If there is no record in the flow table, the controller programs flow table of the switch. The forwarding engine performs packet forwarding.

Fig. 5 shows the architecture of the OpenFlow network, whose main elements are switches that support the OpenFlow protocol and controller that contains centralized network intelligence. The OpenFlow switch represents a generalization of the Ethernet switch in which the data plane is separated from the control plane and abstracted by table of flows. External control is done through the secure channel.

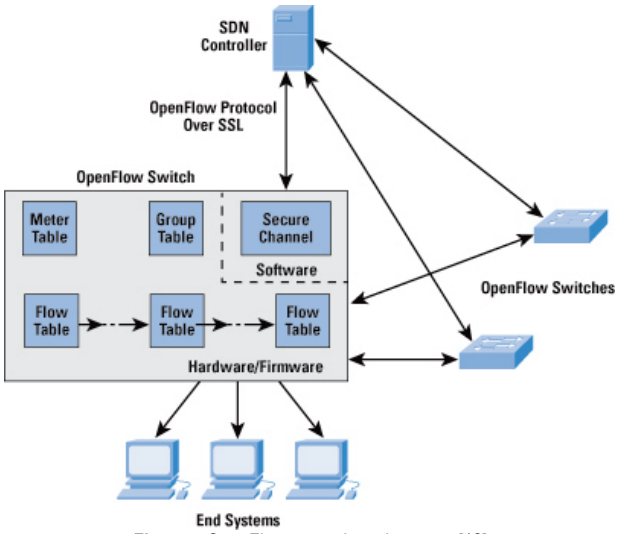

Figure 5 OpenFlow network environment [12]

5.2 Secure Channel

The security channel is an interface through which the OpenFlow switch exchanges messages with the controller. These messages must comply with the format defined by the OpenFlow protocol and are transmitted in binary format via the TCP connection. Within the OpenFlow protocol three types of control messages are defined (Fig. 6): controller-switch messages, asynchronous and symmetrical messages.

Controller-switch messages are generated by the controller and do not always require a response of the switch. They are used to configure the switch, to manage the table of flows, obtaining information about its contents and characteristics of the switch.

The switch sends asynchronous messages to the controller to inform it about an event in the network. One such situation is when the switch receives a packet that does not belong to any of the flows defined in the table. Then by a special type of asynchronous message (PACKET_IN message), a packet (or a part of it) is sent to controller for examination. Sending asynchronous

messages can be also initiated with the expiration of the records in the table of flows, changing the state of one of the interfaces or some other problem in the network.

Symmetrical messages are sent in both directions and are used to detect connection problems between the OpenFlow switch and controller.

Messages	Description
Controller –To - Device	
Features	Request the capabilities of a switch. Switch responds with a features reply that specifies its capabilities.
Configuration	Set and query configuration parameters. Switch responds with parameter settings.
Modify-State	Add, delete, and modify flow/group entries and set switch port properties.
Read - State	Collect information from switch, such as current configuration, statistic, and capabilities.
Packet- out	Direct packet to a specified port on the device.
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.
Role – Request	Set or query role of the Openflow channel. Useful when switch connects to multiple controllers.
Asynchronous – Configuration	Set filter on asynchronous messages or query that filter. Useful when switch connects to multiple controllers.
Asynchronous	
Pocket-in	Transfer packet to controller.
Flow-Removed	Inform the controller about the removal of a flow entry from a flow table.
Port - Status	Inform the controller of a change on a port.
Error	Notify controller of error or problem condition.
Symmetric	
Hello	Exchanged between the switch and controller upon connection startup.
Echo	Echo request reply messages can be sent from either the switch or the controller, and they must return an echo reply.
Experimenter	For additional functions.

Figure 6 OpenFlow messages [13]

5.3 OpenFlow Controller

The controller is a key component of OpenFlow architecture. It conducts control policy by distributing appropriate instructions to network devices. It is responsible for making decisions on the manner of processing packets, and manages the flow table by adding or deleting records in it through the security channel. The controller centralizes the network intelligence, while network maintains distributed data plane over OpenFlow switches. Accordingly, the controller provides the interface which is necessary for the management, control and administration of table flows.

Typically, the controller is installed on a server connected to a network, with two possible configurations of control: centralized and distributed. In the first, one controller manages all switches in the network, while in the second, two or more controllers are responsible for controlling two or more groups of OpenFlow switches. Centralized configuration is problematic because in the case of controller failure, all operations in a network are interrupted.

In a distributed configuration, each controller must have a map of network in real time to avoid packet loss. The map contains information about the topology, the location of users, hosts, firewall and other devices and services. Moreover, it includes all the relations between the names of end systems and addresses. Currently, a number of different open-source implementations of controllers are available that are based on Python, C, C++ or Java programming language.

6 OpenStack

The cloud is a computing model in which a complete IT infrastructure of an organization including computers, network resources, storage capacities and software is abstracted and is available to users in the form of internet service. Given the types of services offered to customers, there are three basic categories of the cloud: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

OpenStack is an IaaS cloud project initiated by Rackspace Cloud and NASA. Currently, over 150 companies are associated with this project (the most important of them are: AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Citrix, Dell, HP, IBM, Yahoo). The project represents a Linux open source solution. OpenStack is written in Python and is developed under the Apache license.

The technology consists of a series of connected projects that control large sets of resources for computing, storage and networking resources in whole data center. OpenStack is controlled via a control panel that gives administrators control, in order to provide resources to users via the network interface.

From the server point of view, this includes a standardized hardware, operating system and their configuration. As part of the infrastructure, data storage can also be defined, as well as VLAN configuration for network isolation. Furthermore, the management of these resources should be provided to the user in the form of a standard logical interface for configuration changes and collecting infrastructural information. Within the private cloud model, IT departments can quickly provide requirements for LOB (Line-of-business) applications, much faster and in more economical way than companies using their own infrastructure.

These components are primarily available through shared infrastructure and virtualization technology. Instead of each application devoted to special hardware, virtualization allows hardware resources to be shared and allocated to a larger number of applications, resulting in greater efficiency, utilization, and lower costs. In addition, the possibility of expanding or downsizing, moving the application by cloning, or live migration of virtual machines is also possible with virtualization.

The basic concepts of OpenStack architecture are described in the works [15] and [16].

6.1 OpenStack Services

The basic services of OpenStack are [17]: (a) OpenStack Compute (Nova) - manages the lifecycle of instances of virtual machines from creation and configuration to definition of security policies. It supports a variety of hypervisors such as KVM, VMware, Hyper-V and Xen; (b) OpenStack Storage - allows to store large amounts of data using a simple API interface. There are two projects: Swift (object storage) and Cinder (block storage); (c) OpenStack Networking (Neutron) - enables management of network resources and IP addressing via the API interface; (d) OpenStack Dashboard (Horizon) - represents a web interface for management of other services; (e) OpenStack Identity (Keystone) - service for authentication and authorization. It also enables access to the register of implemented OpenStack services.

7 IMPLEMENTATION AND ANALYSIS OF SDN IN OpenStack ENVIRONMENT

The aim of this research is to examine in practice the functionality and features of SDN, the way of creation of new SDN and the possibilities of their implementation in existing conventional networks. The real network that was used is the computer network of the Academy of Criminalistic and Police Studies in Belgrade, with the existing typical instances (professors and students).

Since the installation of the VMware hypervisor determined that the necessary hardware is quite demanding, ie. three physical VMware ESXi servers in the cluster are needed for integration and also VMware vCenter Server, for the purpose of this research the Microsoft Hyper-V for Windows Server 2012 R2 is used.

Firstly, the Windows Server 2012 R2 was installed, and then Hyper-V roll added. Hyper-V allows administrators to create multiple virtual machines. The virtual machine is a separate, isolated environment that has its own operating system and applications. This operating system is designed to be maximally secured. The hypervisor in the Windows operating system is the basis of Hyper-V. In order to add this functionality to the operating system, the Server Manager is used, which is the central console for a lot of administrative work in host computer.

7.1 Creating Networks and Routers in OpenStack Environment

On the left side of the OpenStack Network Topology window under the Project group that is the target project, subgroups can be found with the most common settings (for instances and networks). Choosing the Network Topology button, a graphical representation of created network can be obtained. From this window, in addition to the current view, it is possible to create new networks, routers and run instances.

After creating the necessary network, it is possible to graphically see the table, their subnets and whether they are external or internal. It is very easy to add new networks, to reconfigure the existing as well as to delete unnecessary elements.

When a router is created, if the mouse cursor is moved over it, a small window opens where it is possible to check whether the router is active and IP address of the interface that is connected to the external network. By clicking the 'Add Interface' button, the subnets of other networks can be added, where one wishes to route or NAT traffic.

7.2 Running Instances in OpenStack Environment

Running instances can be performed from a disk image or its snapshot. For now, OpenStack supports the following image formats: AKI, AMI, ARI, ISO, OVA, QCOW2, RAW, VDI, VHD and VMDK. When an image is imported into OpenStack, it can run a number of instances. Since it was decided to use Microsoft Hyper-V and its virtual machines run from image in VHDX format, which OpenStack does not support yet, it has to be converted into VHD. The existing images can be converted into Windows PowerShell interface, which is a command-line shell and scripting language designed for administrators to facilitate system administration, and improving automation. One further needs a QEMU tool for disk images in Windows environment used for creating, converting and validating different formats of virtual disks. It is compatible with Hyper-V, KVM, VMware, VirtualBox and Xen hypervisors. QEMU is an open source tool and can be downloaded on the website http://wiki.qemu.org/Main_Page.

Conversion from VHDX to VHD format with QEMU tool is performed by the command: qemu-img.exe convert win10.vhdx -o vpc -o subformat=dynamic win.vhd.

Validation of the image is done with the command: qemu-img check -r all win.vhd.

Once the virtual disk is obtained in VHD format or another format supported by OpenStack, it is possible to import it through a window that is obtained by clicking on Images. In this window, one can specify the image name, its location, format and minimal resources [18].

The created images in the OpenStack table can be edited, deleted and instances of them created. These images can be used in all projects for creating instances.

Running instances can be performed from the window Network Topology or Instances. In this window one can give the name of the instance, choose some already defined or created according to one's specific needs (using option 'Flavor'), input the number of instances that will be launched, choose the source, and, at the end, select the name of the image.

Using the 'Networking' button a window opens that allows one to select the networks that will be connected to this particular instance. In fact, network adapters are added that are connected to the selected networks, while their configuration depends on DHCP of these networks or the specific adjustments.

Working in OpenStack instances is more comfortable from the Hyper-V Manager than from the OpenStack console, but there is a problem with the names of instances. To be exact, real names were given for instances in OpenStack (for example, 'KPA Ruter' and 'Profesor') while in the Hyper-V manager the names of the virtual machines were defined (for example, instance-00000026 and instance-0000001c). In order to facilitate the review, the authors wished to rename the virtual machines, but in that case, OpenStack would not see them anymore.

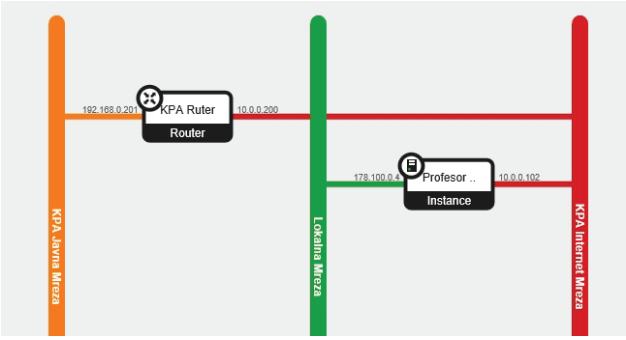

Figure 7 Final OpenStack topology with networks and subnets [19]

After starting all virtual machines, one can state that they see each other on the network and can access the assigned folders and a professor can use the Internet from his instance, while students are not allowed. If necessary, it is very easy to enable students to use the Internet by connecting their instances to the Internet network or connecting the entire local network of the Academy to the router. It is also possible to create a group of instances in the window Access & Security and define the rights to use the network resources according to IP addresses and protocols.

7.3 Monitoring of OpenFlow SDN Traffic Using Wireshark Program

In the SDN network created in OpenStack environment one can perform monitoring, collection and analysis of software package OpenFlow by Wireshark program. Wireshark [20] is a free open-source packet analyzer. By the end of 2014, the authors had to install an addition to the Wireshark program called OpenFlow Dissector to be able to keep track of OpenFlow packets. In later versions it is already supported. OpenFlow uses TCP as its transport protocol at ports 6633 and 6653. Wireshark supports all OpenFlow versions: 1.1, 1.2, 1.3, 1.4 and 1.5. To monitor the OpenFlow packets, the adapter was selected and the filter for TCP ports was activated, or conversely, the already created filter can be used that includes all versions of OpenFlow by entering the name of the filter "of".

7.4 Performance Comparison of Traditional and SDN Networking

Traditional and SDN networking show significant differences when compared. Those differences can be represented in the following table (Tab. 1).

Table 1 Differences between traditional and SDN networking

Traditional Networking	Software Defined Networking
Networks are static and inflexible. New business ventures do not find them useful. They possess little agility and flexibility.	They are programmable networks during deployment time, as well as at a later stage based on changes in the requirements. They help new business ventures through flexibility, agility and virtualization (Network Function Virtualization - NFV).
They are (dedicated) hardware appliances - (referring to one or multiple switches, routers and/or application delivery controllers).	Their configuration is based on using open software - OpenFlow (the industry accepted implementation and standard of SDN) acts like an instruction set for addressing.
They have a distributed control plane.	They have a logically centralized control plane - centralized traffic management.
Custom ASICs (Application Specific Integrated Circuits) and FPGAs (Field Programmable Gate Arrays) are used.	They use merchant silicon.
They function by using protocols.	APIs are used to configure as per need - Each application adopts an SDN specific API.

There are three most commonly propounded advantages of SDN [14]:

- efficiency (due to SDN's holistic view of the network and deeper understanding of inter-application requirements allowing the SDN controllers to perform far smarter traffic engineering, route determination and load balancing than traditional QoS implementations),
- agility (within the datacenter, the SDN can considerably help with automation of network reconfiguration and enhance virtualization agility), and
- security (this aspect will be elaborated in the later text).

Disadvantages regarding SDN are mostly related to its organization and finance. The most common reasons why SDN is not brought into effective action are staff (since it requires certain re-training or recruitments), reorganization (since knowledge between business requirements, application, servers and networking teams need to be shared) and security (explained later in the text that follows).

However, there are a number of benefits regarding SDN. Firstly its configuration accuracy which enables an administrator to manage the whole network as if it were a single device. Consistency and flexibility are also the positive sides of SDN. Secondly, multiple paths can be identified by a SDN controller and not only a single one when it flows from the source of configuration to its destination.

Besides, this approach allows the flow's traffic to be split across multiple nodes. Network performance and scalability is enhanced by optimizing the network path for a particular data flow based on the source and destination nodes [21].

Replacing a plethora of legacy network systems by a single network operating system with open interfaces between hardware, software and applications, cutting down operational complexities makes networks simpler for operator to manage. This actually makes SDN technology simpler than the traditional one. Operators can build virtual networks over a common physical network by centralising control across multiple network elements. To a large extent, software rather than hardware will do the

configuration of the networks. Moreover, expensive, proprietary hardware platforms will not be required.

The limitations that are related to traditional hardware-centric network approach confront organizations of SDN.

Traditional configuration takes more time and it is error-prone. There are many steps to be done in order to add or remove a single device. The first step is to manually configure multiple devices (switches, routers, firewalls). The next step is using device-level management tools to update numerous configuration settings [21].

Multi-vendor environments require a high level of expertise–it demands high knowledge of all present device types from an administrator in order to successfully complete a configuration of a variety of equipment of different vendors.

Traditional architectures complicate network segmentation. In addition to tablets, PCs and smartphones, other devices such as alarm systems and security cameras are also linked to the internet. The explosion of smart devices is accompanied by a new challenge for organizations: how to incorporate all these devices of different vendors within their network in a safe and structured manner. In the case of a compromised device, this design risks by giving external parties access to the entire network [21].

7.5 Security Aspect of SDN

There are certain advantages and disadvantages when the security aspect of SDN is evaluated.

Advantages [14], [22]:

- It will be much simpler to manage centrally the network traffic and device configuration due to rapid configuration implementation.
- The new central control point for the control plane aspects of enterprise networking: Routing and switching controls for data will be simple to implement within controllers, too.
- Traffic shaping and quality of service (QoS) may be more flexible, with improved denial of service (DoS) and distributed denial of service (DDoS) detection/prevention: Software-based packet analysis and traffic control could potentially help with QoS and DDoS detection/prevention. SDN can improve network security by providing basic (typically layer 2 to 4) packet filtering at the network ingress and throughout the network, thus reducing the amount of undesirable traffic entering and traversing the network. Similar to the ability to dynamically modify service chains and network connectivity, it is easier to insert a physical or virtual firewall/IDS/IPS into a network path or to orchestrate packet captures and flow analyses. With more dynamic (and therefore more up-to-date) security policies and RBAC, there will be less possibility for security and resource allocation loopholes to occur.

Disadvantages [22]:

- A new weak point to administer and audit (and attack): SDN controllers now represent a very valuable asset and thereby a new aspect to the organization's threat surface.
- The need to define policies and encryption controls for SDN: Restricting and protecting control traffic will be paramount, and new policies and encryption segments will be needed. Open Flow - relies heavily on TLS. Mismanaged TLS keys lead to compromised network.
- Potential false positives for log management and SIEM (Security Information and Event Management) in control traffic (and new log types): If the control traffic is monitored, and controllers produce logs for monitoring (which they should), then some adaptation and learning will need to take place for traditional security monitoring platforms.
- Availability: Redundancy for controllers will be of prime importance, since there will be a point of blockage in the network.

SDN is considered to be a relatively new, rapidly evolving technology. It has new protocols that are followed by certain weaknesses and vulnerabilities. Hackers, industrial espionage and others can find SDN an interesting target since it still misses maturity and the possibilities for compromising the network control layers.

8 CONCLUSIONS

Trends such as user mobility, server virtualization, IT-as-a-Service and the need to respond quickly to different business conditions make substantial demands to networks-requirements that today's conventional network architecture cannot satisfy. Software-defined networks give a new, dynamic network architecture that transforms the traditional backbone networks into rich service-delivery platform.

SDN will change today's static network into a flexible, programmable platform with intelligence to support virtualization, highly automated and secure cloud environment. With the many benefits and significant industrial moment, SDN is on the way to becoming the new standard for the entire network system. A software defined network includes architecture that is dynamic, cost-effective and flexible, so it is easy to operate, making it ideal for the dynamic nature of today's applications. The separation of the control and the infrastructure layer enables the control of the network to become equally programmable and lower layers infrastructure to be separated for applications and network services. SDN offers a centralized view of the network, giving controller the ability to act as the "brain" of the network, strategic control point in SDN network that communicates with switches/routers via the southbound API, and with applications via northbound API. One of the most popular protocols used for communication between controller and network devices is OpenFlow.

The practical part of this research confirmed that the implementation of a software-defined network into an existing standard network is relatively easy to realize, thereby remaining fully in open-source framework. All the functionalities of the existing network were held, and providing of network monitoring is also simple in completely open-source environment.

9 REFERENCES

[1] Sallai, G. (2014). Future Internet Visions and Research Clusters. *Acta Polytechnica Hungarica, 11*(7), 5-24.

[2] Raghavan, B., Casado, M., Koponen, T., Ratnasamy, S., Ghodsi, A., & Shenker, S. (2012). Software-defined internet architecture: Decoupling architecture from infrastructure. *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, New York, USA, 43-48. https://doi.org/10.1145/2390231.2390239

[3] Tiwari, V. (2013). *SDN and OpenFlow for beginners with hands on labs*, M.M. D.D. Multimedia LLC., Kindle Edition, Northville.

[4] Sundararajan, R. K. (2013). *Software Defined Networking (SDN) - A definitive guide*. Kloudspunn Press, Kindle Edition.

[5] Goransson, P. & Black, C. (2014). *Software Defined Networks*, 1st Edition - A Comprehensive Approach.

[6] Cvjetić, A. (2014). Softverski definisani data centri – hir velikih ili realnost. *Connect*, 42, PC Press. https://pcpress.rs/softverski-definisani-data-centri-hir-velikih-ili-realnost/ (27.09.2016).

[7] Sotirov, B. (2016). Why Software Defined Networking (SDN)? http://www.slideshare.net/lz1dsb/why-sdn (27.09.2016).

[8] Hinden, R. M. (2014). SDN and Security: Why take over the hosts when you can take over the Network? *RSA Conference* / San Francisco. http://www.rsaconference.com/writable/presentations/file_upload/tech-r03-sdn-security-v3.pdf (27.09.2016).

[9] Hood, D. (2014). SDN architecture, TR-502 Open Networking Foundation, https://www.opennetworking.org/images/stories/downloads/sdn-resources/technicalreports/TR_SDN_ARCH_1.0_06062014.pdf (27.09.2016).

[10] Kontesidou, G. & Zarifis, K. (2009). OpenFlow Virtual Networking: A FlowBased Network Virtualization Architecture, Master of Science Thesis, Stockholm, Sweden. http://www.diva-portal.org/smash/get/diva2:302700/FULLTEXT01.pdf (27.09.2016).

[11] Bernstein, G. SDN Overview. Grotto Networking. https://www.grotto-networking.com/BBSDNOverview.html (27.09.2016).

[12] Stallings, W. (2013). Software-Defined Networks and OpenFlow. The Internet Protocol Journal, 16(1). https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html (27.09.2016).

[13] SDN Geeks, Understanding OpenFlow Communication Channel, 2014. https://sdngeeks.wordpress.com/2014/07/30/understanding-openflow-communication-channel/ (27.09.2016).

[14] Roper, J. Software Defined Networking: Should Do Now? Should Do Never? Simply Don't Know! https://entuity.com/resources/sdn-white-paper/. (27.09.2016).

[15] Bumgardner, V. K. C. (2016). *OpenStack in Action*. Manning Publications Company.

[16] Denton, J. (2014*). Learning OpenStack Networking (Neutron) - Architect and build a network infrastructure for your cloud using OpenStack Neutron network*ing, Packt Publishing Ltd., Birmingham, UK.

[17] OpenStack, Chapter 1 - Architecture, 2016. http://docs.openstack.org/juno/install-guide/install/apt/content/ch_overview.html (27.09.2016).

[18] Coballes, A. M. (2013). Blog de Alberto Molina Coballes, How to launch an instance on OpenStack (I): Horizon, https://albertomolina.wordpress.com/2013/11/20/how-to-launch-an-instance-on-openstack-i-horizon/ (27.09.2016).

[19] Erlenvajn, D. (2016). Implementation of software-defined network using open-source solutions, *Master of Science Thesis*, Academy of Criminalistic and Police Studies, Belgrade-Zemun.

[20] Wireshark, 2016. http://www.wireshark.org/ (27.09.2016).

[21] IPknowledge. Traditional vs Software Defined Networking– IPknowledge. https://www.ipknowledge.net/wp-content/uploads/2014/12/SDN.pdf (27.09.2016).

[22] Shackleford, D. Software-Define Networking: What Security Teams Need to Know Now? Free report. https://www.iansresearch.com/insights/reports/software-defined-networking-what-security-teams-need-to-know. (27.09.2016).

Contact information:

Petar ČISAR, Associate Professor, PhD
Academy of Criminalistic and Police Studies
Cara Dušana 196, 11070 Beograd-Zemun, Serbia
petar.cisar@kpa.edu.rs

Dragan ERLENVAJN, Master of Informatics
Security Information Agency
Ulica kraljice Ane bb, 11000 Beograd, Serbia
erlenvajn@gmail.com

Sanja MARAVIĆ ČISAR, College Professor, PhD
(Corresponding author)
Subotica Tech-College of Applied Sciences
Marka Oreškovića 16, 24000 Subotica, Serbia
sanjam@vts.su.ac.rs