Implementing Dynamic Management Of Virtual Network Infrastructure Components*

Klimova A.S.
Ural Federal University
Ekaterinburg, Russia
alina_klimova1503@mail.ru

Kodolov S.D
Ural Federal University
Ekaterinburg, Russia
sergey.kodolov@urfu.ru

Aksyonov K.A.
Ural Federal University
Ekaterinburg, Russia
wiper99@mail.ru

Filimonov A.Yu.
Ural Federal University
Ekaterinburg, Russia
a.filimonov@urfu.ru

Abstract

The active using virtualization technologies which are applied in the construction of network infrastructure provides a possibility of applying new methods and tools for managing the components of such infrastructures. The paper discusses the use of such tools as the Ansible platform, the NCCLIENT and PyEZ libraries for the dynamic management of virtual components of a hardware-software workbench as part of the laboratory complex of software-defined network infrastructures of RTF Ural Federal University (UrFU).

1 Introduction

The basic virtualization principle is the sharing of a physical resource between independent virtual objects. In telecommunication systems, this principle is used for the formation of virtual overlay networks [1] to create virtual network layers (network slices) [2] on top of the underlay (real) network infrastructure. According to the Network Functions Virtualization (NFV) classification of the ETSI Industry Specification Group (ISG), this approach is called Virtual Partitioning [3]. The use of solutions based on the principles of virtualization of network functions and resources allows not only to significantly increase the efficiency of using the network infrastructure, but also provides the possibility of a radical improvement quality of the information services provided on it [4, 5]. The widespread introduction has led to the need for a crucial change in traditional approaches to monitoring and managing telecommunication systems and the emergence of the concept of programmable or software-configurable networks (Software-Defined Network, SDN) of such solutions in the infrastructure of data centers. As you know, this concept is based on the unification of processes and tools for the information interaction of information services with the communication infrastructure, on the basis of which they are provided [6]. The classic version of the implementation of the SDN concept involves the use of a controller as an intermediary in the interaction between the service or application and the communication infrastructure by using the OpenFlow configuration protocol - OF-CONFIG [7]. It should be noted that recently there has been a tendency to reducing quantity the implementations of the classic SDN scheme and, at the same time, growing popularity of alternative directions that are based on other protocols for managing the communication infrastructure or do not involve the using of controllers as separate pieces of system [8,9]. However, this does not mean SDN concept crisis, but, instead, a transition to a new level its development. The most fruitful idea that was proposed under this concept was the proposal to separate the monitoring and control of communication equipment on the control plane (Control Plane) and direct data processing (Data Plane) [10]. This is exactly feature that makes SDN a catalyst for a radical change in the process of managing network infrastructure components, which led to the emergence of a new family of universal configuration platforms (such as

Ansible) [11] and led to model-driven automating of this process [12]. Today, in the field of IT, a wide range of various software tools and information resources is actively used which are aimed at supporting the development or ensuring the implementation of technical solutions of SDN. Such tools and resources include:

- software configuration management platforms [13], such as Ansible, Saltstack, Chef, Puppet.
- special libraries for universal programming languages, such as, for example, the NETCONF Client (NCCLIENT) [14], which are designed to automate the management of network infrastructures.

In following sections, we will discuss the use of tools such as the Ansible platform, the NCCLIENT and PyEZ libraries for dynamically managing the virtual components of the hardware-software laboratory bench of the software-defined network infrastructures of UrFU.

2 Infrastructure of UrFU's SDN workbench

UrFU's SDN workbench Infrastructure is based on hardware complex, which was previously used for laboratory VoIP classes providing. It includes both virtual and hardware components of the network infrastructure, using of which, as was shown earlier [15], lead to increasing the efficiency of using existing equipment, intensifying the learning process and at the same time help to avoid using of the nonessential components.

To organize serial connections between the routers of the laboratory complex, fractional digital streams are used, formed in the E1 channel, the parameters of which are defined in ITU-T G.704 [16]. In this case the distributed resource is the channel capacity (in this case 2048 Kbps), the resources allocation is performed according to the Time Division Multiplexing (TDM) scheme. This solution allows you to create up to 30 virtual channels in one physical channel and combine these channels into virtual groups (interfaces), the bandwidth of which can vary from 0 to 2048 Kbps with a 64 Kbps discrete. To organize virtual IEEE 802.3 (Ethernet) connections, the laboratory complex uses Virtual Local Area Network (VLAN) technology [17]. The organization of virtual IP (Internet Protocol) connections is carried out using Virtual Routing and Forwarding (VRF) technology [18] in the laboratory complex. The topology of the laboratory complex is shown in Figure 1. Black color on this topology denotes real, and gray one - virtual components, respectively.

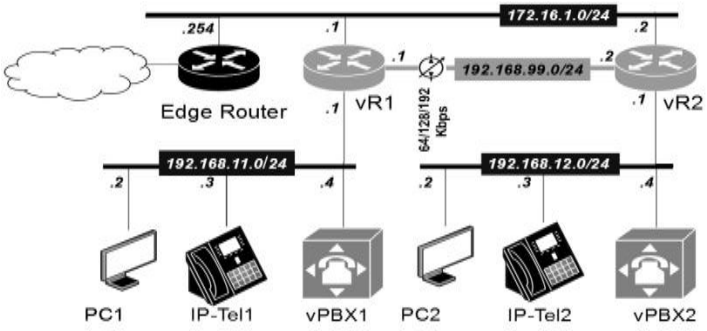

Figure 1: The infrastructure of the laboratory complex

The complex described above was used for laboratory work as part of the testing of the optional course “Fundamentals of IP-telephony”. During the performance of these works, 2-4 teams of students worked simultaneously, whose task was to configure virtual PBXs and other infrastructure components in order to ensure incoming and outgoing calls using standard (Cisco Unified IP Phone 7912G / 7941G / 7942G) or video (Cisco Unified IP Phone 9971) Phones.

Performed tests showed the full performance of the laboratory complex and confirmed the correctness of the solutions that were chosen for its construction. To assess the feasibility of implementing a system that could provide dynamic control of the virtual channel capacity during an audio-video call, it was necessary to determine the characteristics of the information stream devoted to provide it. The measurements were carried out in one of the virtual layers of the topology shown in Fig. 1 using Cisco Unified IP Phone 9971 phones, in which the H.264 / AVC codec is used for encoding / decoding an audio-video stream [19]. Since the encoding algorithm of this codec uses a variety of techniques for compressing video images and motion compensation, the information flow characteristics generated by it strongly depend on the dynamic characteristics of the transmitted image (scene). In order to take these features into account during

measurements and tests, two types of scenes were used: quasi-static, when moving objects practically did not get into the camera field during the call; dynamic, when there was a working office fan in the camera field.

The measurements results are shown in Fig. 2.a and 2.b of the information flow characteristics during the transmission of quasi-static and dynamic VoIP images, respectively. The presented histograms were generated according to the results of 500 measurements with three values of the throughput of the virtual channel that connects vR1 and vR2: 128, 192 and 256 Kbps (vertical hatching, inclined hatching and uniform filling, respectively).

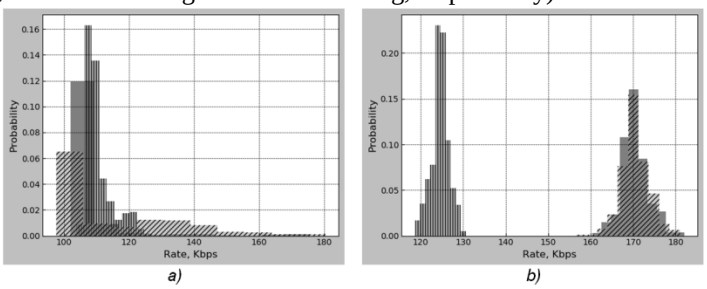

Figure 2: Measurement results of VoIP information flow characteristics

The obtained results made it possible to establish that during a quasi-static images exposing for VoIP phones, they formed digital stream, by average data rate of (110 ± 10) Kbps. To transmit such digital stream 128 Kbps throughput channel was sufficient.

It was also found that during a dynamic images exposing for VoIP phones, they formed digital stream by average data rate of (170 ± 10) Kbps. Such stream was transmitted without loss through channels with a bandwidth of 192 and 256 Kbps. Examples has shown in Fig. 3.a-3.c of such call's by using a virtual channels with a bandwidth of 64, 128 and 192 Kbps, respectively.

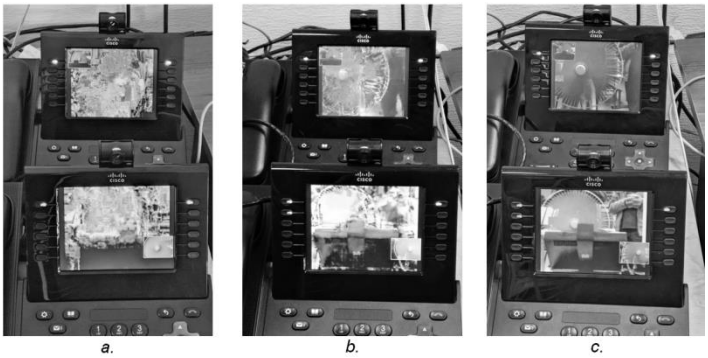

Figure 3: The effect of channel bandwidth on image quality

During those tests, it was found that the transmission led to losses and deterioration of image quality due to overloading of this channel of the information flow of an audio-video call with a moving image through a channel with a bandwidth of 128 Kbps. It should also be noted that a channel provides almost error-free transmission of such a stream with a bandwidth of 192 Kbps. The results, obtained by those tests were used to develop a program that is able to control the bandwidth of a virtual channel by redistributing the resource of the corresponding component of the underlay network (in this case, the physical channel) between the virtual layer's components.

3 Using Ansible to configure UrFU's SDN workbench

Among configuration management platforms such as Ansible, Saltstack, Puppet, or Chef, Ansible has been the most popular over the past 5 years [20]. Ansible is commonly used to provide, deploy, and manage a computing infrastructure in the cloud, virtual, or physical environment [21]. Among the advantages of Ansible, it is worth noting that there is no

need to deploy an agent on configurable devices (agentless). Other advantages are low cost compared to competitors, and ease of installation and administration [21]. Automation of building a network infrastructure using Ansible does not require deep knowledge in programming: the programming required for many common operations has already been completed and is available as modules. In the process of automation, a “playbook” is created that describes the necessary operations, combining a number of modules. Ansible configuration file (ansible.cfg); File or directory of the workspace (inventory); A playbook file in YAML format that contains operations to execute.When a playbook is executed, the parameters are imposed on the template for each device, forming separate configuration files, which are then applied on the device.

In the event of problems associated with a managed device, and not with syntax or other errors in the playbook, Ansible monitors errors separately for each device. If an error occurs with a specific device, Ansible stops processing tasks for that device. However, if there are no errors on other devices, the tasks for these devices will be completed [22]. The main advantage of automation platforms is the ability to automate configuration management of both real and virtual equipment using popular communication protocols in general and Ansible in particular. Depending on the platform and module destination, XML over SSH (NETCONF) [23], CLI over SSH, or API over HTTPS (RESTCONF) [24] can be used as communication protocols.

To manage the constructed UrFU's SDN workbench, playbook and template files were developed, which allow to change the bandwidth of the communication channel of two virtual IP PBX.

As a result, changing the bandwidth of the communication channel of two virtual routers is reduced to the execution of one command: ansible-playbook slot-settings.yaml -e 'slot = 1-4', where the -e parameter passes the value of the slot variable, which determines the number of time-slots used.

Automation platforms, considered on the example of Ansible, are convenient for efficient configuration of many devices and are used, as a rule, in cases when the required state of the system after configuration is determined in advance. To solve the problem posed in this article, it is important that the system itself can respond to changes and move from one state to another depending on certain factors. Therefore, the use of these tools is impractical in solving this problem.

4 Using libraries based on NETCONF Protocol to provide UrFU’s SDN workbench management

The NETCONF and RESTCONF protocols are actively used today to automate the management of network equipment [25]. The use of these protocols allows us to translate the solution of network equipment control problems to a qualitatively new level, making possible dynamic reconfiguration of the network infrastructure in real time [12]. The advanced libraries have been developed [26] for many of the popular programming languages that make the automation process of managing network equipment accessible to a wide range of developers in order to ensure the development and promotion of such applications.

Initially, the scripts were prepared using the Python library - NCCLIENT, working on the protocol NETCONF - a network protocol for managing devices. Using the library simplifies equipment setup and management. The library is exported to a Python script and connection parameters via SSH are set to connect to devices.

The developed technological scripts allow reading, configuring and monitoring routers without using the command line. The use of these scripts greatly simplified the equipment configuration processes, and made it possible to get operational access to such parameters of the virtual port of the router as the current value of the transmission speed on this port and the number of packets that were dropped due to an overload.

The connect function is used, in the parameters of which the necessary data is connected to establish a connection with the device. The current device configuration is written to the xml Dom using the get_config function. The full list is given in the repository of the prepared technological scripts [27]. Another tool is the PyEZ library [28] for automating the management of network equipment. The PyEZ library is similar to NCCLIENT one except it supports only Juniper devices. The narrow focus of this library allows you to form a more compact program code.

In the process of comparing the tools the development of a technological script is the most suitable in Python using libraries based on the NETCONF Protocol to solve the problem 'of dynamic management of virtual components.

The next stage of the work was the preparation of the controller program which allows to dynamically adapt the bandwidth of the virtual channel to the requirements of the transmitted signal.

5 Programming hardware and software UrFU's SDN workbench

The controller program is a Python script that uses the procedures of the PyEZ library described above. The script runs on the workstation which is located in the equipment control network.

The controller operation algorithm includes the following steps: channel status monitoring; compute of parameters; accepting decision on the need to change the channel capacity; bandwidth change if necessary.

At the monitoring stage, using the methods of the PyEZ library and the NETCONF protocol, the controller requests the following parameters from the router: serial port speed, interface transfer rate and number of lost packets. The measured data rate values are recorded in two parameters: one contains the current speed value, the other contains the maximum transmission rate since the channel bandwidth changes. The values of the lost packets are written to a buffer, which is organized as a queue with a fixed length. Thus, at each moment of time, the average value is calculated of discarded packets for a certain period of time. As a result of monitoring and calculation of the parameters, the following values are determined:

- Average number of dropped packets;
- Difference between maximum and current data rates.

At the next stage, a decision is made on the need for bandwidth changes. The controller increases the transmission channel if the average number of dropped packets exceeds the value X. In the event that the difference between the maximum and current data transfer rate exceeds the value Y, the controller reduces the transmission channel. The block diagram is shown in Figure 4 of the controller program algorithm.

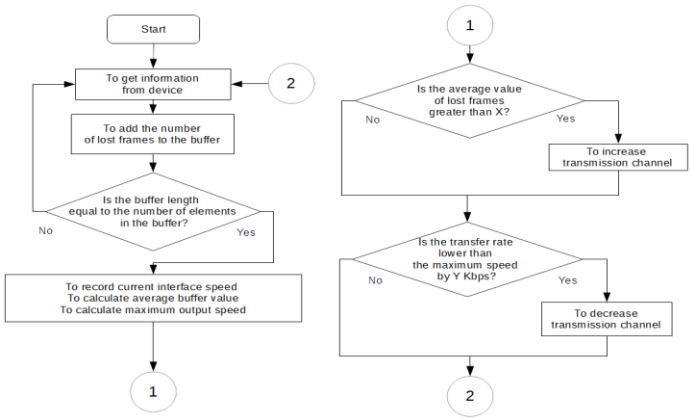

Figure 4: Block diagram of the algorithm of the controller program

The values of X and Y were determined experimentally, and for the conditions under which the tests were carried out, these values were 20 packets and 30 Kbps, respectively. Bandwidth change is also implemented using the NETCONF protocol: an XML configuration is downloaded and applied to devices with the number of time slots defined by the controller.

At the final stage of works, dynamic tests were carried out of the program controller as a part of the hardware and software workbench. To perform such tests in the topology shown in figure 1, an audio-video call was initiated between vPBX1 and vPBX1. During this call, quasi-static and dynamic images were alternately used and values were fixed:

- the maximum possible information rate (Limit Rate);
- the current speed information (Current Rate);
- rates of change in the number of packets dropped due to congestion (Drop Rate).

The timing diagram is shown in Figure 5 of one of the dynamic tests of the controller program. In this diagram the ordinate axis depicts the values of information speeds expressed in Kbps and the number of packets dropped over the current time interval.

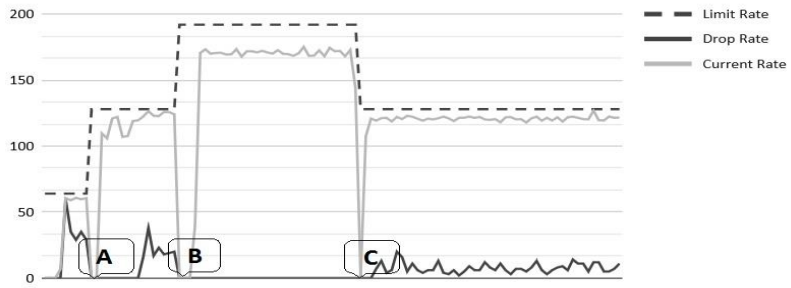

Figure 5: Controller dynamic test chart

The virtual channel bandwidth is set to 64 Kbps and a static image is used in the initial phase of the test. Since this bandwidth is not enough to transmit a static image, the Drop Rate exceeds the threshold value and the controller decides to increase the information speed limit to 128 Kbps (point " A" in the diagram). Then the data transmission process becomes stationary 110 Kbps. The presence of such fluctuations is explained by the fact that the tests were carried out for a sufficiently long time interval (from 3 to 10 minutes) and video telephones recorded the movements of test participants in the classroom. Approximately one minute after the start of the test, a moving object (a floor fan) was placed in the field of the video phone cameras, which, as shown above, increased the average information speed to 170 Kbps. In this case, as in the previous one, the Drop Rate exceeds the threshold value and the controller decides to increase the maximum information speed, this time to the value of 192 Kbps (point "B" on the diagram). After another minute, the moving object was removed from the field of video phone cameras, and the average value decreased again to 110 Kbps of the information speed. In this case, the discrepancy between the limiting and current speeds exceeds the threshold value and the controller decides to reduce the limiting information speed to 128 Kbps (point "C" in the diagram).

Multiple repeated tests showed similar results, which confirm the performance of hardware and software solutions that were the basis for the described workbench. The process did not require additional resources for the developing a hardware and software workbench alone; its programming also did not cause much difficulty due to the presence of good documentation of the NCCLIENT and PyEZ libraries and a large number of examples of their use. All this made it possible to carry out work on the creation and commissioning of the workbench in a fairly short time (May-October 2019).

6 Conclusion

The active use of virtualization technologies which are applied in the construction of modern network infrastructures makes it necessary to use new methods for managing the components of these structures. The introduction of these methods allows you to automate the component management process, which helps to increase the efficiency of using the network infrastructure's resources and make it possible dynamic changing infrastructure's configuration. Today, these features can be realized through the use of software platforms designed for configuration management or applications based on special libraries that are designed for many of the popular programming languages. The paper discusses the experience of using such tools as the Ansible platform, the Python libraries of the NCCLIENT and PyEZ for the dynamic management of virtual components of the laboratory workbench in the complex of software-configured network infrastructures of RTF UrFU. Due to the availability and maturity of these tools, the work was fully completed in a short time on creating and programming the laboratory workbench. The experience gained during the of these works fulfilling allows us to conclude that the future use of such tools is promising and fruitful for automation of network equipment management.

Acknowledgements

This research paper is supported by 211 acts of the Government of the Russian Federation, agreement No. 02.A03.21.0006.

References

1. Jaime Galán-Jiménez and Alfonso Gazo-Cervero OVERVIEW AND CHALLENGES OF OVERLAY NETWORKS. A SURVEY // International Journal of Computer Science &Engineering Survey (IJCSES) Vol.2, No.1, Feb 2011 DOI : 10.5121/ijcses.2011.2102.
2. S Gutz, A Story, C Schlesinger, N Foster Splendid isolation: A slice abstraction for software-defined networks. // Proceedings of the first workshop on Hot topics in software defined networks ACM 2012. pp 79-84.
3. ETSI GS NFV-INF 005 V1.1.1 (2014-12) - Network Functions Virtualisation (NFV); Infrastructure; Network Domain <http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/010/01.01.01_60/gs_NFV-INF010v010101p.pdf>.
4. NFV Management and Orchestration - An Overview, GS NFV-MAN 001 V1.1.1, European Telecommunications Standards Institute (ETSI), 2014.
5. Pedreno-Manresa, Jose-Juan & Sayyad Khodashenas, Pouria & Siddiqui, Muhammad Shuaib & Pavon-Marino, Pablo. (2017). Dynamic QoS/QoE assurance in realistic NFV-enabled 5G Access Networks. 10.1109/ICTON.2017.8025149.
6. H. Kim, N. Feamster, Improving network management with software defined networking, Communications Magazine, IEEE 51 (2) (2013) 114–119.
7. OF-CONFIG 1.2 OpenFlow Management and-Configuration Protocol https://www.opennetworking.org/images/stories/downloads/ sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf
8. Perspective: Controller-based vs Controllerless-based SDN Solutions https://www.pluribusnetworks.com/blog/ perspective-controller-based-vs-controllerless-based-sdn-solutions/
9. SDN: Where is it now and what is the future? https://www.computerweekly.com/feature/SDN-Where-is-it-now-and-what-is-the-future
10. Hakiri, Akram & Gokhale, Aniruddha & Berthou, Pascal & Schmidt, Douglas & Gayraud, Thierry. (2014). Software-Defined Networking: Challenges and research opportunities for Future Internet. Computer Networks. 75. 10.1016/j.comnet.2014.10.015.
11. 10 BEST Software Configuration Management Tools (SCM Tools in 2019) https://www.softwaretestinghelp.com/ top-5-software-configuration-management-tools/
12. Kundrát, Jan & Vojtech, Josef & Skoda, Pavel & Vohnout, Rudolf & Radil, Jan & Havlis, Ondrej. (2018). YANG/NETCONF ROADM: Evolving Open DWDM towards SDN Applications. Journal of Lightwave Technology. PP. 1-1. 10.1109/JLT.2018.2822268.
13. Chef vs Puppet vs Ansible vs Saltstack: Which Works Best For You? https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/
14. NCCLIENT: Python library for NETCONF clients https://github.com/ncclient/ncclient
15. Filimonov A., Medvedev D., Klimova A., Muraviev A. Application of virtual infrastructure components in construction of a laboratory complex in an educational institution.// Journal of Instrument Engineering (Izvestiya vysshikh uchebnykh zavedeniy. Priborostroenie) vol.61, December 2018, p. 1092-1099.
16. ITU-T Rec.G.704 (10/98) Synchronous frame structures used at 1544, 6312, 2048, 8448 and 44 736 kbit/s hierarchical levels <https://www.itu.int/rec/T-REC-G.704-199810-I/en>/.
17. IEEE Std 802.1Q (2012): "IEEE Standard for Local and metropolitan area networks Media Access Control (MAC) Bridges and Virtual Bridges" <https://ieeexplore.ieee.org/document/6606799/>.
18. J.Sonderegger, O.Blomberg, K.Milne, and S.Palislamovic, “Virtualization for high availability,” in Junos High Availability: Best Practices for High Network Uptime (Animal Guide)ch. 5, p. 119, O’Reilly Media,1 ed., August 2009.
19. ITU-T Rec.H.264 Advanced video coding for generic audiovisual services https://www.itu.int/rec/T-REC-H.264-201906-I/en
20. Chef vs Puppet vs Ansible vs Saltstack: Which Works Best For You? https://www.edureka.co/blog/chef-vs-puppet-vs-ansible-vs-saltstack/
21. ANSIBLE ESSENTIALS. https://www.ansible.com/webinars-training/introduction-to-ansible
22. Sawtell S. DAY ONE: Automating JUNOS with Ansible, 2nd Edition. – Juniper Networks Books, 2018. – 398 p.
23. IETF RFC6421 Network Configuration Protocol (NETCONF) https://tools.ietf.org/html/rfc6241

24. IETF RFC8040 RESTCONF Protocol https://tools.ietf.org/html/rfc8040
25. Wallin, S., & Wikström, C. Automating network and service configuration using NETCONF and YANG". In LISA'11 Proceedings of the 25th International Conference on Large Installation System Administration (2011)(p. 22).
26. Netconf Central http://www.netconfcentral.org/
27. RTF PKS repository https://github.com/KlimovaAlina/SDN
28. Junos PyEZ Documentation (https://www.juniper.net/documentation/product/en_US/junos-pyez)