Software Defined Networks
and Internet of Things

Internet of Things
for Industry and Human Applications

ALIoT

Internet of Things
for Industry and Human Applications

Software Defined Networks
and Internet of Things

PRACTICUM

Funded by the
Erasmus+ Programme
of the European Union

Newcastle University

LEEDS BECKETT UNIVERSITY

KTH VETENSKAP OCH KONST

UNIVERSIDADE D COIMBRA

IT Альянс

SmartMe UNIVERSITY

Ministry of Education and Science of Ukraine
Zaporizhzhia National Technical University
Volodymyr Dahl East Ukrainian National University
National Aerospace University "Kharkiv Aviation Institute"

V.V. Shkarupylo, R.K. Kudermetov, I.S. Skarga-Bandurova,
A.Yu. Velykzhanin, L.O. Shumova, D.S. Mazur,
V.S. Kharchenko, D.D. Uzun, Y.O. Uzun, P.A. Hodovaniuk

Internet of Things for Industry and Human Applications

Software defined networks and Internet of Things

Practicum

Edited by R.K. Kudermetov

Project
ERASMUS+ ALIOT "Internet of Things: Emerging Curriculum for Industry and Human Applications" (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP)

2019

UDC 004.7:004.6+004.415/.416](076.5)=111
П78

Reviewers:
Prof., Dr. Felicita Di Giandomenico, ISTI-CNR, Pisa, Italy
Prof., DrS. Volodymyr Opanasenko, State Prize Winner of Ukraine, V.M. Glushkov Institute of Cybernetics of NAS, Ukraine

П78 V.V. Shkarupylo, R.K. Kudermetov, I.S. Skarga-Bandurova, A.Yu. Velykzhanin, L.O. Shumova, D.S. Mazur, V.S. Kharchenko, D.D. Uzun, Y.O Uzun, P.A. Hodovaniuk. **Software defined networks and Internet of Things**: Practicum / Kudermetov R.K. (Ed.) – Ministry of Education and Science of Ukraine, Zaporizhzhia National Technical University, Volodymyr Dahl East Ukrainian National University, National Aerospace University “KhAI”, 2019. – 129 p.

ISBN 978-617-7361-93-9

The materials of the practical part of the study course “PC2. Software defined networks and IoT”, developed in the framework of the ERASMUS+ ALIOT project “Modernization Internet of Things: Emerging Curriculum for Industry and Human Applications Domains” (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP).

Practicum materials are supposed to be used by PhD-students in sphere of computer networking, software engineering etc., and aimed at delivering the essential knowledge and practical skills on the topic of Mininet emulator usage for the purpose of typical engineering tasks solving, covering, in particular, the aspects of programming – for the purpose of automation tasks solving. Practicum is devoted to development, implementation and testing of SDN-based IoT-solutions. Moreover, techniques and tools of DevOps application in context IoT and Big Data are described.

Practicum materials are intended to be used by the PhD-students on computer networking, software engineering, and engineers and researches involved in the development, implementation and testing of SDN-based IoT-solutions and DevOps techniques application.

Ref. – 39 items, figures – 65, tables - 2.

Approved by Academic Council of National Aerospace University “Kharkiv Aviation Institute” (record No 4, December 19, 2018).

ISBN 978-617-7361-93-9

Міністерство освіти і науки України
Запорізький національний технічний університет
Східноукраїнський національний університет ім. В. Даля
Національний аерокосмічний університет
ім. М. Є. Жуковського «Харківський авіаційний інститут”

В.В. Шкарупило, Р.В. Кудерметов, Д.С. Мазур, І.С. Скарга-Бандурова,
Л.О. Шумова, А.Ю. Великжанін, В.С. Харченко, Д.Д. Узун,
Ю.О. Узун, П.А. Годованюк

Інтернет речей
для
індустріальних і гуманітарних застосунків

ПРОГРАМНО-КОНФІГУРОВАНІ МЕРЕЖІ ТА ІНТЕРНЕТ РЕЧЕЙ

Практикум

Редактор Кудерметов Р.К.

Проект ERASMUS+ ALIOT
“Інтернет речей: нова освітня програма для потреб
промисловості та суспільства”
(573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP)

2019

УДК 004.7:004.6+004.415/.416](076.5)=111
П78

Рецензенти: Проф., д-р Фелічіта Ді Джандоменіко, ISTI-CNR, Піза, Італія

Д.т.н., проф. Володимир Опанасенко, Лауреат Державної премії України, Інститут кібернетики ім. В.М.Глушкова НАН України

П78 В.В. Шкарупило, Р.В. Кудерметов, Д.С. Мазур, І.С. Скарга-Бандурова, Л. О. Шумова, А. Ю. Великжанін, В.С. Харченко, Д.Д. Узун, Ю.О. Узун, П.А. Годованюк. **Програмно-конфігуровані мережі та Інтернет Речей: Практикум** / За ред. Кудерметова Р.К. – МОН України, Запорізький національний технічний університет, Східноукраїнський національний університет ім. В. Даля, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ». – 129 с.

ISBN 978-617-7361-93-9

Викладено матеріали практичної частини курсу PC4 “ Програмно-конфігуровані мережі та IoT ”, підготовленого в рамках проекту ERASMUS+ ALIOT “ Internet of Things: Emerging Curriculum for Industry and Human Applications” (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP).

Матеріали для практикуму призначені для докторантів у галузі комп'ютерних мереж, інженерії програмного забезпечення тощо та спрямовані на надання необхідних знань та практичних навичок щодо використання емулятора Mininet для вирішення типових інженерних завдань, що охоплюють, зокрема, аспекти програмування для вирішення завдань автоматизації. Крім того, практикум присвячено дослідженням, розробці, впровадженню та тестуванню IoT-рішень на основі SDN, а також впровадженню DevOps методології в контексті IoT і великих даних.

Призначено для інженерів, розробників та науковців, які займаються розробкою та впровадженням IoT систем, для аспірантів університетів, які навчаються за напрямами IoT, кібербезпеки в мережах, комп'ютерних наук, комп'ютерної та програмної інженерії, а також для викладачів відповідних курсів.

Бібл. – 39, рисунків – 65, таблиць -2.

Затверджено Вченою радою Національного аерокосмічного університету «Харківський авіаційний інститут» (протокол № 4, 19 грудня 2018).

ISBN 978-617-7361-93-9.

ABBREVIATIONS

API – Application Programming Interface
CI/CD – Continuous Integration / Continuous Delivery
CLI – Command Line Interface
DHCP – Dynamic Host Configuration Protocol
GUI – Graphical User Interface
HTML – HyperText Markup Language
ICMP – Internet Control Message Protocol
IP – Internet Protocol
LSA – Link State Advertisement
ONOS – Open Network Operating System
OS – Operating System
OSPF – Open Shortest Path First
PING – Packet Internet Group
POCO – Pareto-Optimal Controller
QoE – Quality of Experience
QoS – Quality of Service
RTP/TS – Real-time Transport Protocol
RTT – Round-trip Time
SDLC – Software Development Lifecycle
SDN – Software Defined Networking (Network)
SDP – Specification of Developed Project
SPF – Shortest Path First
SSH – Secure Shell
VM – Virtual Machine
VND – Visual Network Description

INTRODUCTION

The materials of the practical part of the study course “PC2. Software defined networks and IoT”, developed in the framework of the ERASMUS+ ALIOT project “Modernization Internet of Things: Emerging Curriculum for Industry and Human Applications Domains” (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP)[1].

Nowadays, the paradigm of Software-defined networking (SDN) is broadly considered to be the basis for the Internet of Things (IoT) solutions implementation. The defining features of Software-defined networks are programmability, easiness of network topology reconfiguration, centralized control, automation, etc. To test the soundness of resulting SDN-solutions, especially the functioning of the controller – centralized control unit, different emulators are brought to the table. One of those that is widely used is the Mininet environment, providing the reliable basis for diverse industrial scenarios and implementations.

Practicum materials are supposed to be used by PhD-students in sphere of computer networking, software engineering etc., and aimed at delivering the essential knowledge and practical skills on the topic of Mininet emulator usage for the purpose of typical engineering tasks solving, covering, in particular, the aspects of programming – for the purpose of automation tasks solving. Moreover, practicum is devoted to engineers and researches involved in the development, implementation and testing of SDN-based IoT-solutions.

Practicum covers four modules: PCM 2.1 “Software defined networks basics”, PCM 2.2 “SDN programming and simulation of SDN composing, configuring and scaling”, PCM 2.3 “Algorithms and applications for utilization of SDN technology to IoT” and PCM 2.4 "Development of project for SDN-DevOps using modern CI/CD tools".

The PCM2.1 “Software defined networks basics” module covers one laboratory work – “Installation and configuration of Mininet environment”. This laboratory work is devoted to covering the basics of Mininet emulator utilization – command line usage, network topology creation and testing aspects.

---

[1] *The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.*

The PCM2.2 “SDN programming and simulation of SDN composing, configuring and scaling” module covers one laboratory work – “Working in MiniEdit graphical environment”. This laboratory work is devoted to covering the peculiarities of MiniEdit graphical utility of Mininet emulator usage, aiming at clarification and simplification of engineer/developer work experience.

The PCM2.3 “Algorithms and applications for utilization of SDN technology to IoT” module covers three laboratory works and three tutorials aimed, in particular, at teaching SDN controllers, Quality of Services in SDN, IoT data streaming over SDN.

The PCM2.4 "Development of project for SDN-DevOps using modern CI/CD tools" module covers the principles of DevOps techniques and processes of software development lifecycle in context of IoT and Big Data.

Each laboratory work is structured as follows: brief theoretical information disclosing the specifics of work; execution of work as the sequence of steps to be done; recommendations on report creation; control questions to be briefly answered in the report.

Practicum is prepared by Assoc. Prof., Dr. V.V. Shkarupylo, Assoc. Prof., Dr. R.K. Kudermetov, MSc student D.S. Mazur, (Zaporizhzhia National Technical University), Prof., DrS. I.S. Skarga-Bandurova, PhD student A.Yu. Velykzhanin, Dr. L.O. Shumova (Volodymyr Dahl East Ukrainian National University), Prof., DrS. V.S. Kharchenko, Assoc. Prof., Dr. D.D. Uzun, Senior Lecturer Y.O. Uzun, PhD student P.A. Hodovaniuk (National Aerospace University “KhAI”).

General editing has been performed by Head of Computer Systems and Networks Department of Zaporizhzhia National Technical University, Assoc. Prof., Dr. R.K. Kudermetov.

The authors are grateful to the reviewers, project colleagues, fellows of the departments of academic organizations for valuable information, methodological assistance and constructive suggestions that have been made during practicum materials discussion and preparation.

PCM2.1. Software defined networks basics

Assoc. Prof., Dr. V.V. Shkarupylo, Assoc. Prof., Dr. R.K. Kudermetov, MSc student D.S. Mazur (ZNTU)

Laboratory work 1

INSTALLATION AND CONFIGURATION OF MININET ENVIRONMENT

Goal: get familiar with technical aspects of Software defined network functioning and usage; obtain practical skills in sphere of Mininet emulator installation, configuration and utilization.

Laboratory work participants: lecturers, scientists, technical staff, students and post-graduate students of the department (faculty, institute) of the university; developers, engineers, trainees.

1.1. Theoretical information

Current level of global networking evolution can be characterized as follows: there is a vital need for such system control, monitoring and configuring aspects improvement. The solution can be found with Software Defined Networking (SDN) principles in mind. Here is the brief list [1]: differentiation between control and date planes [2], unified switches usage to maintain data forwarding, utilization of controller – to coordinate such switches in a centralized manner, stick to OpenFlow protocol [3, 4]. OpenFlow specification provides open interface for SDN-network components communication. The key components are controller, switches and hosts.

Because of the fact that SDN technology is relatively new, it is commonly relatively difficult to work with such network directly. The solution can be in different emulators usage. The emulator typically is a set of software and hardware means to represent SDN network within virtual environment. SDN software is based on Linux platform. Here are some examples of such emulators: Mininet [5, 6], EstiNet [7], OpenNet [8], ns-3 [9]. Each of these solutions has its advantages and drawbacks. The Mininet emulator though is being frequently considered to be an exemplar to be compared to. That's why this solution will be used in our laboratory works.

Mininet environment is devoted to be the mean for SDN-network emulation, particularly by creating virtual hosts, switches, controllers and connections between these components. Named components and connections between them form the topology of network.

The architecture of SDN is given in Fig. 1 [10].

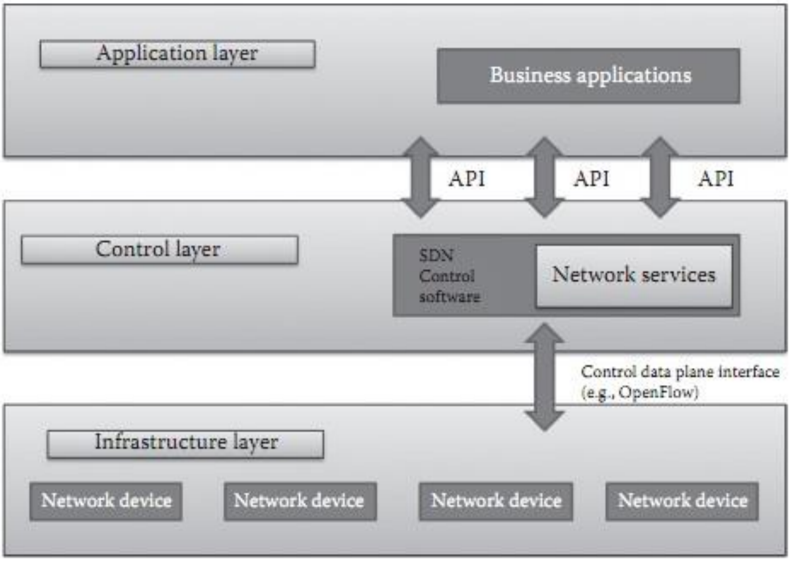


Fig. 1 – Layered architecture of SDN

Mininet environment provides the means to conduct the development, investigation, testing and software configuring of SDN systems, etc.

Mininet provides in particular the following abilities:

– can be used as testbed for SDN applications development;

– brings to the table the ability of different developers to jointly work on network topology;

– includes the means of complex topology testing;

– provides specialized Application Programming Interface (API), oriented on Python programming language usage;

Comparing to typical approaches to virtualization, Mininet provides the following advantages:

– easiness of installation;

– quick boot time;

– easiness of system reconfiguration.

As a drawback the difficulties during the work with graphical environment of Mininet on Windows platform and also the limitation of network configuration by hardware resources available for virtual machine can be pointed out [6].

The tasks to be accomplished during the laboratory work:

– Mininet Linux-environment installation on Windows platform by way of VirtualBox usage;

– virtual machine network interfaces configuration;

– get in touch with basic console commands of Mininet emulator, particularly to create the networks with different topologies.

The presentation of accomplished tasks has to be conducted by one of two ways:

– one-by-one;

– after all the tasks have been accomplished.

Obtained results have to be properly represented in the report to be defended.

1.2. Example of work execution

Step 1. Installation of Mininet environment.

To install Mininet emulator the following software has to be used:

– VirtualBox-4.3.10 has been chosen to be a tool for software virtualization. File VirtualBox-4.3.10-93012-Win.exe is intended to be installed on 32-bit Windows-platform;

– mininet-2.2.1-150420-ubuntu-14.04-server-i386.zip archive has been chosen as Mininet emulator build. This archive has to be unpacked. It can be seen from file name that the content of archive is the emulation of 32-bit ubuntu-14.04-server Linux environment.

After installation of VirtualBox-4.3.10 tool the content of archive – mininet-vm-i386.vmdk file – then has to be used as a hard drive of virtual system to be created.

The snapshot of successfully created virtual machine is given in Fig. 2.

It has to be noted that for the needs of virtual machine it has to be allocated not less than 512 MB of random access memory.

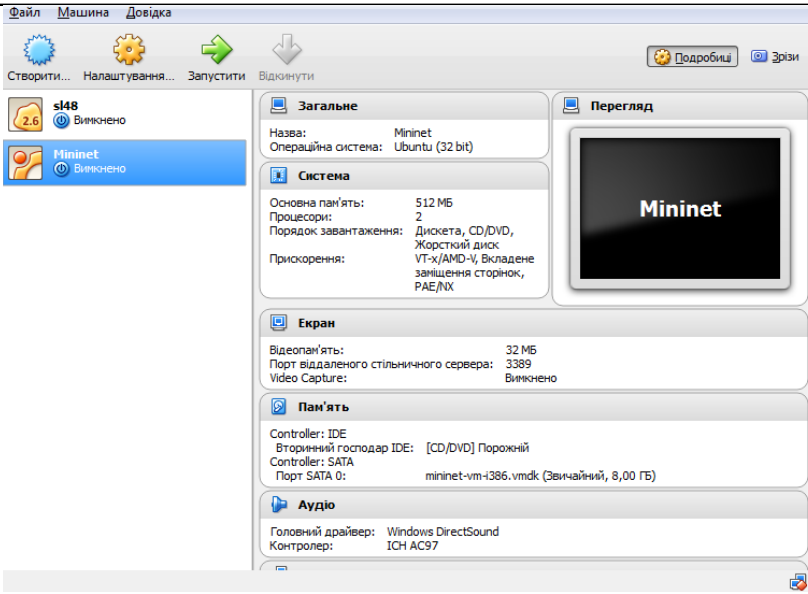


Fig. 2 – The information about virtual machine configuration

Step 2. Configuration of network interfaces.

To configure network interfaces of virtual machine the option "Settings" has to be chosen first, then "Adapter 1" and "Adapter 2" configurations have to be accessed via "Network" option (Fig. 3, Fig. 4).

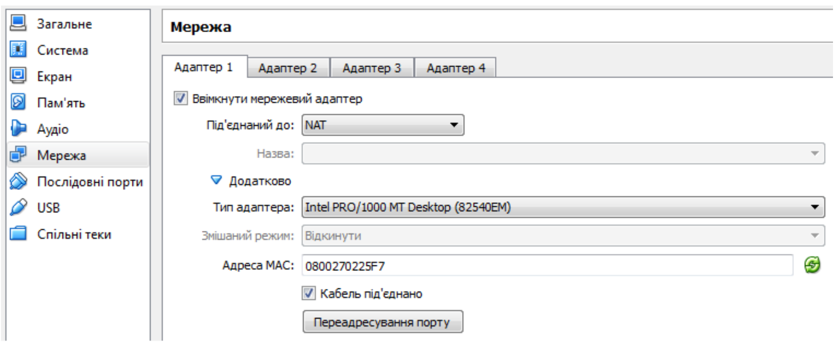


Fig. 3 – "Adapter 1" network configuration

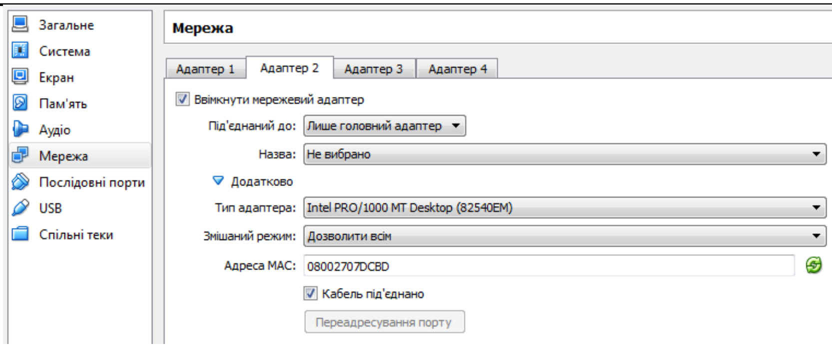


Fig. 4 – "Adapter 2" network configuration

In order to create your own information model, you first need to create a project in which you will work in the future.

Step 3. Mininet usage basics.

Launch virtual machine by pushing the "*Launch*" button.

After a while there will be proposed to enter the login and password in console:

– in the mininet-vm login: field the mininet login has to be entered;

– in the password: field the mininet password has to be entered.

As confirmation of success the following line is going to appear:

```
mininet@mininet-vm:~$
```

1. View the information about network interfaces configuration.

For this purpose the following command has to be executed:

```
> sudo ifconfig
```

As a result, we can check, for instance, the IP-address of *eth0* interface. It's going to be *10.0.2.15* or something like that.

2. Create network topology with minimal configuration – single controller (*c0*), single switch (*s1*) and pair of hosts (*h1*, *h2*):

```
> sudo mn
```

After command execution the information about newly created network with minimal topology will be given, and then the Mininet console will be provided.

3. View the information about Mininet commands:

```
> help
```

4. View the information about all network nodes (there are should be four nodes in total – controller (*c0*), switch (*s1*) and pair of hosts (*h1*, *h2*)):

```
> nodes
```

5. Check the connections between nodes:

```
> links
```

6. Get the information about the IP-address and corresponding subnetwork mask for a particular interface of certain host. For instance, for *eth0* interface of *h1* host, the command will be as follows:

```
> h1 ip addr show | grep eth0
```

7. Test the throughput of communication channel between the specified hosts. For instance, with respect to *h0* and *h1* hosts, the command should be as follows:

```
> iperf h1 h2
```

It takes a while to accomplish the command. Each new command execution will provide slightly different result.

8. View the information about nodes' interfaces:

```
> net
```

9. View the information about nodes configuration:

```
> dump
```

10. View the information about network interfaces of specified node. For instance, for *h1* node the following command should be executed:

```
> h1 ifconfig -a
```

11. Check the information about the processes executed on nodes. For instance, for *s1* node the following command should be executed:

```
> s1 ps -a
```

12. Check the connections between hosts.
Between specified hosts:

```
> h1 ping -c 1 h2
```

Between all pairs of hosts:

```
> pingall
```

13. Launch web server and appropriate client on *h1* and *h2* hosts respectively:

```
> h1 python -m SimpleHTTPServer 80 &
> h2 wget -O - h1
```

As a result of command execution, the HTML-code of web-page, obtained by client, will be shown in the console.

14. Finalize the functioning of web server:

```
> h1 kill %python
```

15. Exit from Mininet console environment back to Linux console:

```
> exit
```

16. Clear topology-related data:

```
> sudo mn -c
```

As a result, all topology-related configuration data will be removed.

1.3. Tasks for individual execution

1. Create Software-defined network with specified parameters.

Solve the task with respect to a given variant. The topology should be created with specified *bandwidth* and *delay* parameters.

Variant 1. Throughput – 10 Mb/s; communication delay: 20 ms:

```
> sudo mn --link tc,bw=10,delay=20ms
```

Variant 2. Throughput – 100 Mb/s; communication delay: 40 ms:

```
> sudo mn --link tc,bw=100,delay=40ms
```

For each variant the following should be done:

– measure the bandwidth of communication channel between hosts 5 times. For this purpose, the *iperf* command should be used. The average should be placed to report;

– find the minimal value of *rtt* (round trip time) parameter with *ping* command.

Remarks:

– *rtt* parameter encompass the time, spent on package transfer from source host to destination host, plus the time on package retrieval notification;

– for each variant, the student should be able to explain the obtained results, e.g., why measured values of bandwidth are lower than specified value of *bw* parameter, minimal value of *rtt* parameter is about 4 times above the specified *delay* value.

2. Create the network with linear topology, encompassing 3 hosts:

```
> sudo mn --test pingall --topo single,3
```

3. Redo step 3 with respect to newly created topology.

4. Create linear topology with four switches and four hosts:

```
> sudo mn --test pingall --topo linear,4
```

This topology will include seven connections.

5. Create a tree-topology network with minimal configuration (one controller, one switch and pair of hosts) and test it with *pingall* command:

```
> sudo mn --topo tree,depth=1,fanout=2 --test pingall
```

The *--topo tree* parameter sets tree topology itself. The depth attribute sets the amount of switch layers (one layer in our case, represented with single element (top) of switches tree): on the potential second layer there will be a pair of switches, on the third – four, and so on. The *fanout* attribute defines the number of connections to each switch. In our case *fanout=2*. This means that, taking into consideration that *depth=1* (there are no other layers with switches and there are no other switches at all), both connections are the direct connections to hosts.

For instance, if we had *depth=2*, there would be one switch from the first layer connected to a pair of switches from the second layer, and those switches from the second layer would be directly connected to a pair of hosts each. That means that there would be three switches and four hosts in total.

The *--test pingall* parameter means that, after creation of network with specified topology, each host should ping all other hosts to test network consistency.

The procedure of such network creation and testing is a time consuming process which will take place about 5 sec and will be shown in console log.

6. Experiment on networks creation and testing with different values of depth and fanout parameters.

1.4. Report content

The report should contain:

– title page with the name of the laboratory work;
– aim of the work; problem statement according to the task;
– work progress and the results of tasks execution;
– analysis of the results and conclusions;
– brief answers to the control questions.

1.5. Control questions:

1. Software Defined Networking. The aim, advantages and drawbacks.
2. Give the definition of 'emulation' notion. Name the examples of SDN emulators.
3. Mininet emulator. Usage and peculiarities.
4. Mininet installation and configuration. Brief description of steps performed.
5. Stages of SDN network with minimal topology creation (by default) – one switch and pair of hosts.
6. Commands to communicate with hosts and switches.
7. Commands to check connections between hosts.
8. Commands to launch web server and appropriate client.
9. Commands to set the delays on communicational channels.
10. Commands to change network configuration.
11. The use of depth and fanout parameters during the creation of network with tree topology. Characterize the impact of these parameters values on total number of network nodes.

1.6. Recommended literature:

1. N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.
2. T. D. Nadeau and K. Gray, *SDN: Software Defined Networks*. Sebastopol, CA : O'Reilly Media, 2013, 384 p.
3. OpenFlow Switch Specification, March 26, 2015 [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf. [Accessed: 8 Jun. 2019].
4. P. Goransson, C. Black, and T. Culver, *Software Defined Networks: A Comprehensive Approach*, 2nd ed. Waltham, MA: Elsevier, 2016, 436 p.
5. Mininet: An Instant Virtual Network on your Laptop (or other PC) [Online]. Available: http://mininet.org. [Accessed: 8 Jun. 2019].

6. F. Keti and S. Askar, “Emulation of Software Defined Networks Using Mininet in Different Simulation Environments,” in *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, Kuala Lumpur, Malaysia, 9-12 February 2015, pp. 205-210.

7. S-Y. Wang, “Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet,” in *2014 IEEE Symposium on Computers and Communication*, Funchal, Portugal, 23-26 Jun. 2014.

8. M-C. Chan et al., “A simulator for software-defined wireless local area network,” in *2014 IEEE Wireless Communications and Networking Conference*, Istanbul, Turkey, 6-9 April 2014.

9. J. Ivey, H. Yang, C. Zhang and G. Riley, “Comparing a Scalable SDN Simulation Framework Built on ns-3 and DCE with Existing SDN Simulators and Emulators,” in *2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*, Banff, Alberta, Canada, 15-18 May 2016, pp. 153-164.

10. F. Hu, *Network Innovation through OpenFlow and SDN: Principles and Design*. Boca Raton, FL : CRC Press, 2014, 520 p.

PCM2.2. SDN programming and simulation of SDN composing, configuring and scaling

Assoc. Prof., Dr. V.V. Shkarupylo, Assoc. Prof., Dr. R.K. Kudermetov, MSc student D.S. Mazur (ZNTU)

Laboratory work 1

WORKING IN MINIEDIT GRAPHICAL ENVIRONMENT

Goal: obtain the skills of MiniEdit graphical environment of Mininet emulator usage. Get familiar with Software defined network topology graphical constructor, network hosts configuring, network topology testing.

Training participants: lecturers, scientists, technical staff, students and post-graduate students of the department (faculty, institute) of the university; developers, engineers, trainees.

1.1. Theoretical information

To foster the convenience of Mininet environment usage, the MiniEdit graphical interface is taking place.

To get started with it, the following toolset is required:

– *VirtualBox software* – the 4.3.10-93012-Win.exe built;

– the deployed *Mininet* virtual machine – the content of 2.2.1-150420-ubuntu-14.04-server-i386.zip archive should be used as a hard drive of virtual machine;

– *PuTTY utility* – the putty.exe executable – openly accessible client software with different protocols support, Telnet and SSH (Secure Shell) in particular. SSH – application layer network protocol that allows to perform remote control of operating system. In contrast with Telnet, all the transmitted data is encrypted;

– *Xming Server* – the xming-x-server-6.9.0.38.exe built – the implementation of X Window System to provide standard instruments and protocols to build graphical interface for user. This server will be used in our work in order to compensate the absence of graphical interface in virtual machine the Mininet environment is installed in. It will provide the opportunity to visualize MiniEdit interface of Mininet that is implemented in Linux-environment of virtual machine within

Windows-environment [1].

First and second tools mentioned have already been covered in previous work.

Current work is all about the following steps:

– Xming Server installation – to work with MiniEdit graphical interface of Mininet in Windows-environment;

– Mininet environment launching and configuration;

– PuTTY SSH-client set-up – to connect to virtual machine's Mininet Linux-environment in encrypted manner;

– connect to virtual machine's Mininet Linux-environment via PuTTY interface.

1.2. Example of work execution

Step 1. Install Xming Server.

Perform the installation of Xming Server on E logical drive.

During this, the "*Full installation*" option should be chosen, the checkbox in front of "*Normal PuTTY Link SSH client*" position should also be set up.

As a result of successful installation, Xming Server will be launched automatically (Fig. 1).

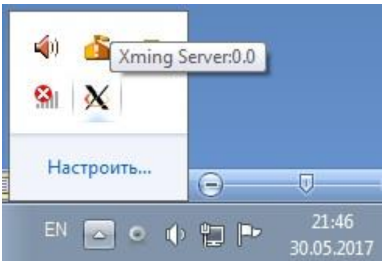


Fig. 1 – An icon indicating that Xming Server has been launched

To launch Xming Server manually, the *XLaunch.exe* utility has to be used. The utility is located in Xming installation directory.

Step 2. Configure Mininet virtual environment.

Launch Mininet emulator via VirtualBox. Enter login and password (mininet, mininet) – as it was for previous work.

Then the following substeps should be performed:

– (1-st substep) – try to launch MiniEdit interface directly in Mininet virtual environment by executing the following console command:

```
> sudo python ./mininet/examples/miniedit.py
```

As a result, the error message will appear: “no display name and no $DISPLAY environment variable”. To get rid of it, the succeeding steps should be done;

– (2-nd substep) – check the configuration of network interfaces:

```
> ifconfig -a
```

As a result of command execution, the IP-address of eth1 interface should be remembered – it should be like *192.168.56.101*. This address will be used further to externally connect to Mininet virtual Linux-environment from Windows-enfvironment by way of PuTTY client usage.

– (3-rd step) – manually configure eth1 network interface as DHCP-client:

```
> sudo dhclient eth1
```

Step 3. Configure PuTTY SSH-client.

To make it possible for PuTTY SSH-client to get access to Xming Server launched, the following steps should be accomplished:

– mark with a checkbox the “*Enable X11 forwarding*” option in “*Connection -> SSH -> X11*” category of PuTTY settings (Fig. 2);

– connect to Mininet virtual Linux-environment in “*Session*” category, entering previously remembered *192.168.56.101* IP-address. Then press “*Open*” to open the session.

Notice: connection port number should be left by default (22). As a connection type, the “*SSH*” option should be marked.

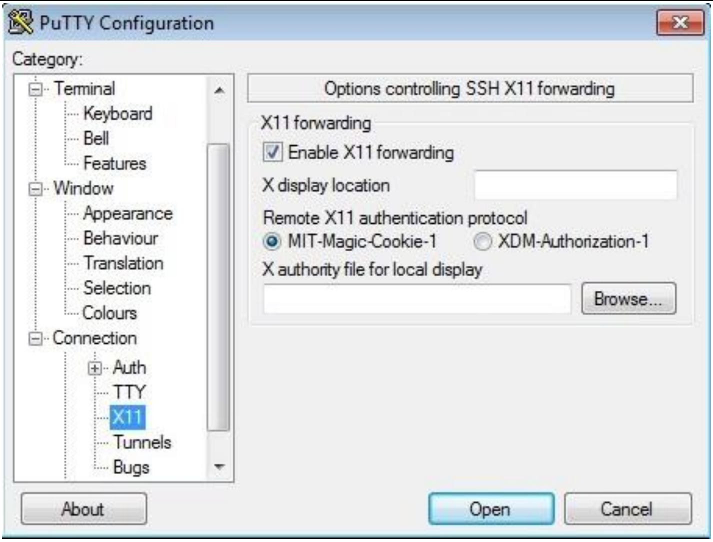


Fig. 2 – SSH-client configuration

Step 4. Connect to Mininet-console via PuTTY-client.

The following steps should be done:

– in opened PuTTY-console the login and password for Mininet (mininet, mininet) need to be entered;

– set up SSH-connection by assigning eth1 interface IP-address:

```
> ssh -Y mininet@192.168.56.101
```

– enter the password upon request – mininet;

– perform 1-st substep of step 2 again by entering the appropriate command via PuTTY-console.

As a result of rightly accomplished steps, the graphical MiniEdit environment should appear in a separate window (Fig. 3). It's possible due to Xming Server functioning.

The main advantage from Xming Server usage is that there is no need to utilize the additional libraries, comparing to Cygwin server [2], when working in Windows-environment.

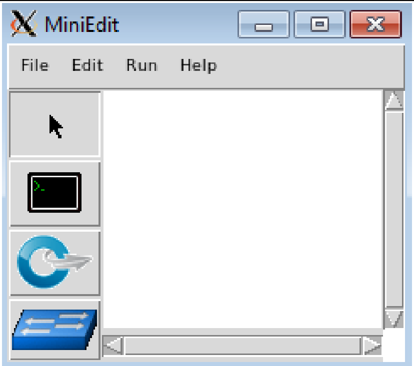


Fig. 3 – MiniEdit workspace

Step 5. Create SDN-network with minimal topology in MiniEdit graphical environment.

The network to be created is depicted in Fig. 4.

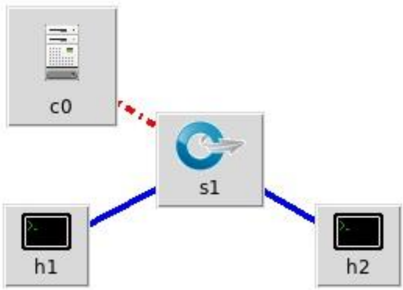


Fig. 4 – Network representation in MiniEdit workspace

Each of the nodes can be configured appropriately (by changing the name, setting IP-address, etc.) by pressing with right mouse button on Properties option. For instance, in case of *c0* node, the properties set will be as given in Fig. 5.

In Fig. 21.5, port 6633 is set by default.

In case of many controllers, their port numbers should be different. For instance, if we have a pair of controllers, port numbers can be set as 6633 and 6634.

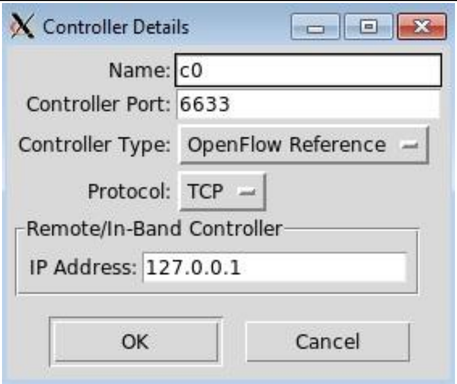


Fig. 5 – Controller configuration

To check and set up network preferences in MiniEdit environment, the option "*Edit -> Preferences*" can be used (Fig. 6) [3].

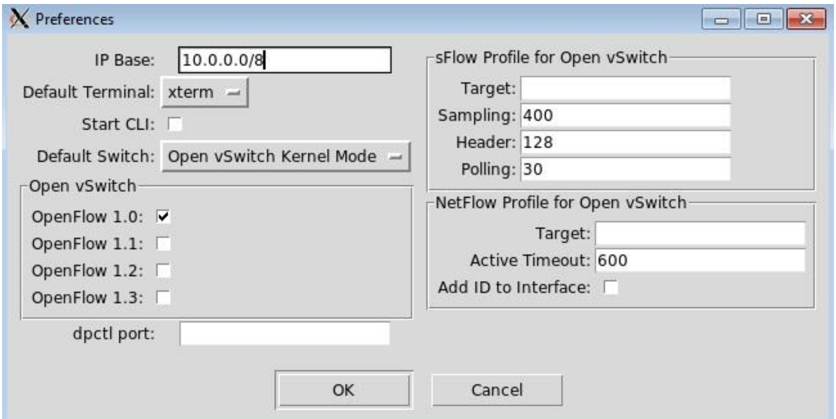


Fig. 6 – Network preferences

In Fig. 6, it can be seen that 1.0 version of OpenFlow protocol is being used by default. It's a typical picture for current level of SDN networks implementation. These preferences are legit only for created topology (Fig. 4).

Step 6. Save created topology in *topo1.mn* file by the default address /home/mininet (the topology file should necessarily have the *.mn extension). Then created topology can be loaded again in MiniEdit environment.

To save the topology, press “*File -> Save*”, to load – “*File -> Open*”.

Step 7. Checking the created topology.

1. Choose “*Edit -> Preferences*” on the panel of instruments. Set the flag in front of “Start CLI”. This will allow work with command line during topology checking by way of simulation.

2. Assign IP-addresses to network nodes.

The default subnetwork mask (in “*Edit -> Preferences*” section) is *10.0.0.0/8*.

3. Check the connection between hosts by launching the simulation process: “*Run -> Run*”. By opening the terminal of *h1* host (press the right mouse button -> “*Terminal*”), execute the *ping* command, assigning the IP-address of node:

```
> ping 10.0.0.4
```

Notice: before closing the terminal, enter the *exit* command.

4) Acknowledge that simulation is running: “*Run -> Show OVS Summary*”.

Step 8. Experiment with created network.

1. While the simulation is running, open the terminal of *h1* host and execute the following command:

```
> wireshark &
```

This will launch the *wireshark* utility (traffic analyzer) on the specified host.

2. In the main window – in *Capture* field (in the bottom left) – mark out the *h1-eth0* interface and press the *Start* control element. This will allow to associate the monitoring software with *eth0* interface of *h1* host.

3. In the terminal of *h1* host once again execute the *ping* command:

```
> ping 10.0.0.4
```

As a result of command execution, in the work area of wireshark software, the information about the packages transmitted through the *h1-eth0* interface will be shown.

1.3. Tasks for individual execution

1. Create SDN-network with tree-like topology. Choose configuration data with respect to the variant.

Notice:

– variant should be chosen with respect to the list number of the student. The odd numbers are associated with the first variant, the even ones – with the second variant.

Variant 1. The network must encompass 2 controllers, 7 switches and 8 hosts. The network should be configured with respect to the recommendations given below.

Variant 2. The network must encompass 3 controllers, 8 switches and 10 hosts. The network should be configured with respect to the recommendations given below.

Recommendations to network configuration:

– assign the unique port numbers to the controllers;

– in “*Edit -> Preferences*” section of instruments panel, set the “*Start CLI*” flag. This will allow use the command line;

– save the created network in “*.mn” file: “*File -> Save*”;

– save corresponding *Python*-script in “*.py” file: “*File -> Export Level 2 Script*”.

Notice:

– before stopping the simulation process, the *exit* command should be entered first. After that, the “*Stop*” control element on the control panel should be pressed.

2. Execute steps 7 and 8 with respect to the network created within the previous task.

1.4. Report content

The report should contain:

– title page with the name of the laboratory work;

– aim of the work; problem statement according to the task;

– work progress and the results of tasks execution;

– analysis of the results and conclusions;

– brief answers to the control questions.

1.5. Control questions:

1. Sequence of steps to get to work in graphical MiniEdit environment on Windows platform.
2. The use of PuTTY utility.
3. The use of Xming Server.
4. The advantages of Xming Server usage.
5. Describe advantages and disadvantages of command line and/or graphical MiniEdit environment usage for the purpose of SDN-network with specified topology creation.
6. Describe the use of *wireshark* utility.
7. Describe the facilities for network testing in graphical *MiniEdit* environment.
8. The use of *Show OVS Summary* command.
9. With respect to tree-like topology of SDN-network, describe the dependencies between the numbers of controllers, switches and hosts.
10. Describe the effect of subnetwork mask format on the potential number of hosts.

1.6. Recommended literature:

1. Xming X Server for Windows [Online]. Available: https://sourceforge.net/projects/xming/. [Accessed: 8 Jun. 2019].

2. Cygwin [Online]. Available: http://cygwin.com. [Accessed: 8 Jun. 2019].

3. How to use MiniEdit, Mininet’s graphical user interface [Online]. Available: http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/. [Accessed: 8 Jun. 2019].

PCM2.3. Algorithms and applications for utilization of SDN technology to IoT

Prof., Dr. Sc. I.S. Skarga-Bandurova, PhD student A.Yu. Velykzhanin, Dr. L.O. Shumova (V. Dahl EUNU)

Laboratory work 1

APPLICATION OF ONOS SDN CONTROLLER PLATFORM FOR IOT NETWORKS MANAGEMENT

Goal and objectives: This laboratory work is an introduction to the managing a software-defined networks (*SDN*). We'll discover the open network operating system (*ONOS*); set up a network OS, and practice in working with *ONOS*.

Learning objectives:

– to study the principles of the *ONOS*;

– to study the possibilities of managing a software-defined network using ONOS.

Practical tasks:

– acquire practical skills in working with *ONOS*.

Exploring tasks:

– discover *ONOS* communication tools to message exchange in the network;

– investigate how to perform basic management operations with *ONOS*.

Setting up.

In preparation for laboratory work it is necessary:

– to clear the goals and mission of the research;

– to study theoretical material contained in this manual, and in [1]-[3];

– to familiarize oneself with the main procedures and specify the exploration program according to defined task.

Recommended software and resources: *ONOS*, *Mininet*, OracleVM *VirtualBOX*, *SDNHub*.

1.1. Synopsis

In this laboratory work you will learn basic software-defined networking (SDN) concepts using the *ONOS* SDN controller, *ONOS* components, and the *Mininet* network simulator. Specifically you will use *ONOS* SDN controller. While you explored this tool using the Linux operating system, the same tool is available for Windows operating systems.

1.2. Brief theoretical information

IoT applications are fundamentally different from traditional ones. IoT intends to connect billions of devices from different manufacturers that can be deployed in an uncoordinated way. These problems can be solved by implementing a high level of automation of delivery and operation of IoT applications. IoT is the area where SDN can be extremely useful. Below are some issues in IoT and how SDN application can help with this:

– mass device connectivity: Adding routing/forwarding information related to the IoT devices will require the automatic detection of these devices and mechanisms for dynamically calculating the route over the network. The SDN controller can be used to interact with switches in the forwarding planes to configure traffic flows over the network for these devices.

– fast network changes: IoT devices are limited in terms of power consumption and processor utilization. They can often be removed from the network due to low battery or processor overload. They also work on a variety of wireless technologies, which can have a significant failure rate. IoT network infrastructure may need to handle fast changes, which requires changing routing/flow information in the network elements. The SDN can optimally handle such scenarios with dynamic topology maintenance.

– network scalability processing. Recent applications and services are developed based on the principles of NFV, when network objects are created or completed on the fly. In IoT, this can mean pruning and grafting IoT controllers (gateways) as needed. SDN can be used to intelligently locate these controllers and update the streams of connected devices accordingly.

– low power sensors: IoT sensors have the very low processing power and need frequent battery replacement. Therefore, they cannot implement complex routing or network management protocols.

One of the most common SDN controllers for this purpose is *ONOS*. *ONOS* is an SDN network operating system with powerful architectural features such as high availability, scalability, modularity, and complete separation of protocol-independent and protocol-specific device and channel representations.

The *ONOS* (Open Network Operating System) project is an open source community supported by The Linux Foundation. The aim of the project is to create SDN operating system for communications service providers designed to provide scalability, high performance and high availability [1].

In this work we will develop and test an IoT network using *ONOS* SDN Controller with the Mininet Network Emulator. The example of network is shown in Fig. 1.

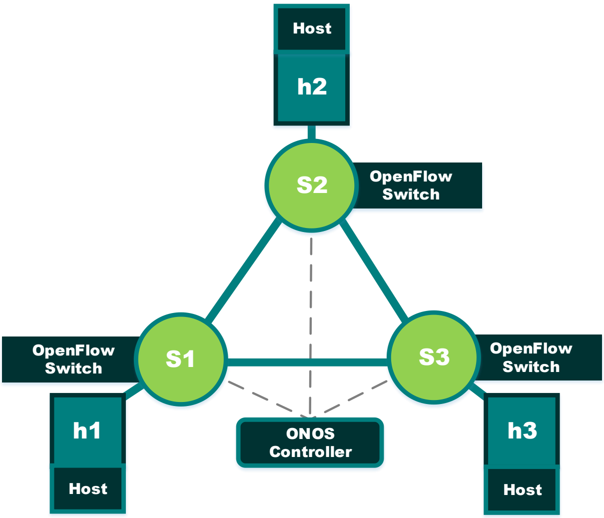


Fig. 1 – An example the network under the test

1.3. Execution order and discovery questions

Step 1. Install *OracleVM VirtualBOX*.

Use link [4] to install OracleVM.

Step 2. Deploy the *SDNHub* image [5].

Step 3. Create a network of switches and hosts.

1. Install *ONOS* [6] and *Mininet Network Emulator* [7].

In this laboratory work we assume you have already set up a *Mininet* VM in VirtualBox.

Start VirtualBox and then start the VM.

We will use the *Mininet* graphical user interface, to set up an emulated network made up of *OpenFlow* switches and Linux hosts. To start *Mininet*, run the following command on a terminal window connected to the *Mininet* VM:

```
> mininet@mininet-vm:~$ sudo ~/mininet/examples/miniedit.py
```

2. Double-click on the downloaded *ONOS* tutorial OVA file will open virtual box with an import dialog. Allocate 2-3 CPUs and 4-8GB of RAM for the VM.

3. Run the *ONOS*. Since we use a ready-made image of the system, almost all the necessary packages have been already installed but you have to conFig. them.

4. Run the command prompt. Then set up a network operating system (Fig. 2).

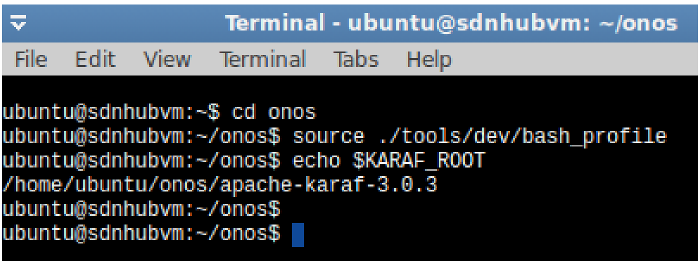


Fig. 2 – Setting the environment variables for *ONOS* and *karaf* execution

Optionally, you can compile the controller using the commands:

```
> mvn clean install -nsu -DskipIT -DskipTests
```

In the new terminal window, write a command:

```
> sudo mn --custom lab1.py --topo mytopo –mac –controller remote
```

Set up a network using standard *Mininet* topology commands. As an alternative for standard command you can use MiniEdit to create the network topology. MiniEdit provides a visual representation of the network. However, while MiniEdit is a good tool for creating topologies for the *Mininet* network simulator, in this lab we create a topology via standard *Mininet* commands.

```
> sudo mn --custom lab1.py --topo mytopo --mac --controller=remote
```

Step 4. Discover *ONOS* Basics.

Before start *ONOS* SDN controller, we need to determine which components we want to run when we start the controller.

1. Start *ONOS* controller with one of the following commands command:

```
> ok clean
> karaf clean
```

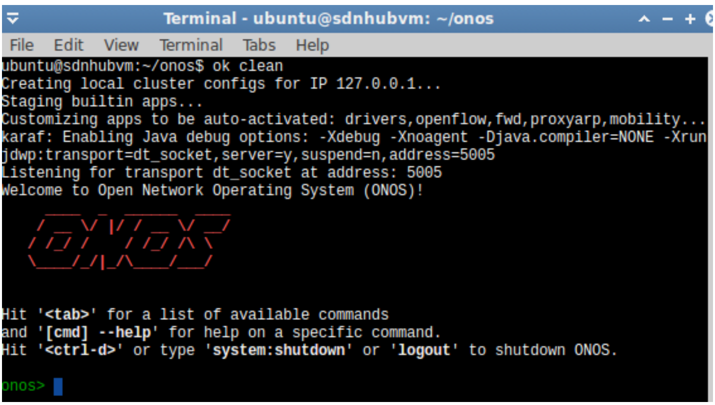


Fig. 3 – Start *ONOS* controller window

2. Look through the main commands. A full list of commands is

available at [8], or use basic *ONOS* tutorial for [9].

2.1. Links. Similarly, the links command is used to list the links detected by *ONOS*.

At the *ONOS* prompt run the command (Fig. 4).

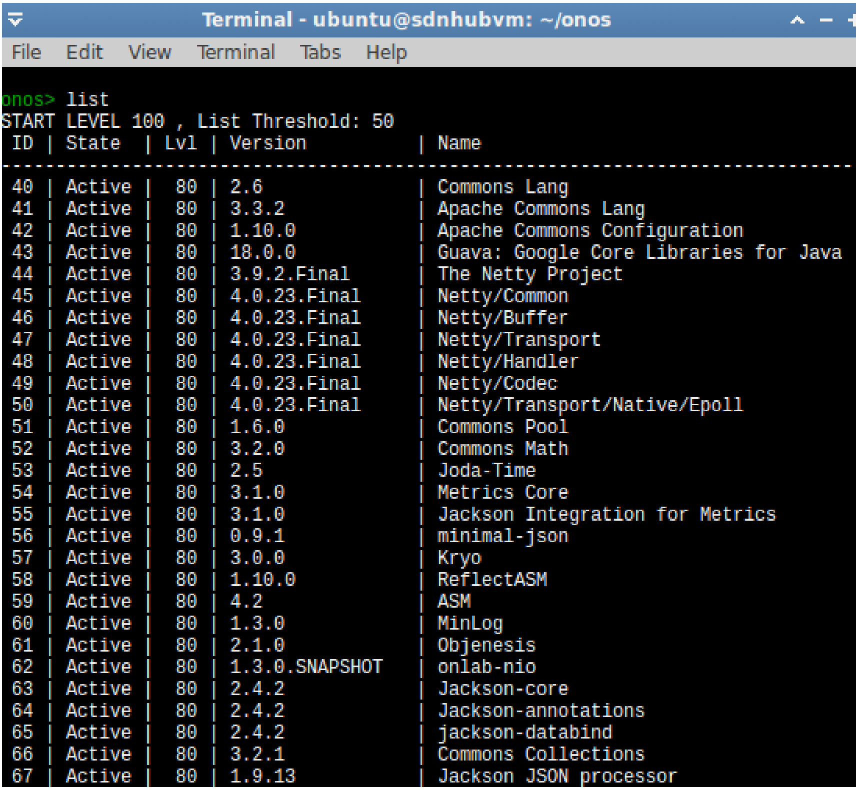


Fig. 4 – List of links detected by *ONOS*

With the help of the "*devices*" command, find out the list of all infrastructure devices (Fig. 5).

```
Terminal - ubuntu@sdnhubvm: ~/onos
File  Edit  View  Terminal  Tabs  Help
onos> devices
id=of:0000000000000001, available=true, role=MASTER, type=SWITCH, mfr=Nicira, In
c., hw=Open vSwitch, sw=2.3.90, serial=None, protocol=OF_13, channelId=127.0.0.1
:50121
id=of:0000000000000002, available=true, role=MASTER, type=SWITCH, mfr=Nicira, In
c., hw=Open vSwitch, sw=2.3.90, serial=None, protocol=OF_13, channelId=127.0.0.1
:50123
id=of:0000000000000003, available=true, role=MASTER, type=SWITCH, mfr=Nicira, In
c., hw=Open vSwitch, sw=2.3.90, serial=None, protocol=OF_13, channelId=127.0.0.1
:50122
onos>
```

Fig. 5 – The list of infrastructure devices

With the help of the "*links*" command, we can find out the list of all infrastructure links (Fig. 6).

```
Terminal - ubuntu@sdnhubvm: ~/onos
File  Edit  View  Terminal  Tabs  Help
onos> links
src=of:0000000000000001/2, dst=of:0000000000000003/2, type=DIRECT, state=ACTIVE
src=of:0000000000000002/2, dst=of:0000000000000003/1, type=DIRECT, state=ACTIVE
src=of:0000000000000003/2, dst=of:0000000000000001/2, type=DIRECT, state=ACTIVE
src=of:0000000000000003/1, dst=of:0000000000000002/2, type=DIRECT, state=ACTIVE
src=of:0000000000000002/1, dst=of:0000000000000001/1, type=DIRECT, state=ACTIVE
src=of:0000000000000001/1, dst=of:0000000000000002/1, type=DIRECT, state=ACTIVE
onos>
```

Fig. 6 – The list of infrastructure links

2.2. Flows list all currently-known flows.

Now all switches have 5 routing rules. Here, deviceID is a device identifier (switch) and id is a routing rule (Fig. 7).

2.3. Run the *Mininet pingall* command.

This command runs ping tests between each host in the emulated network. This generates traffic to the controller every time a switch receives a packet that has a destination MAC address that is not already in its flow table.

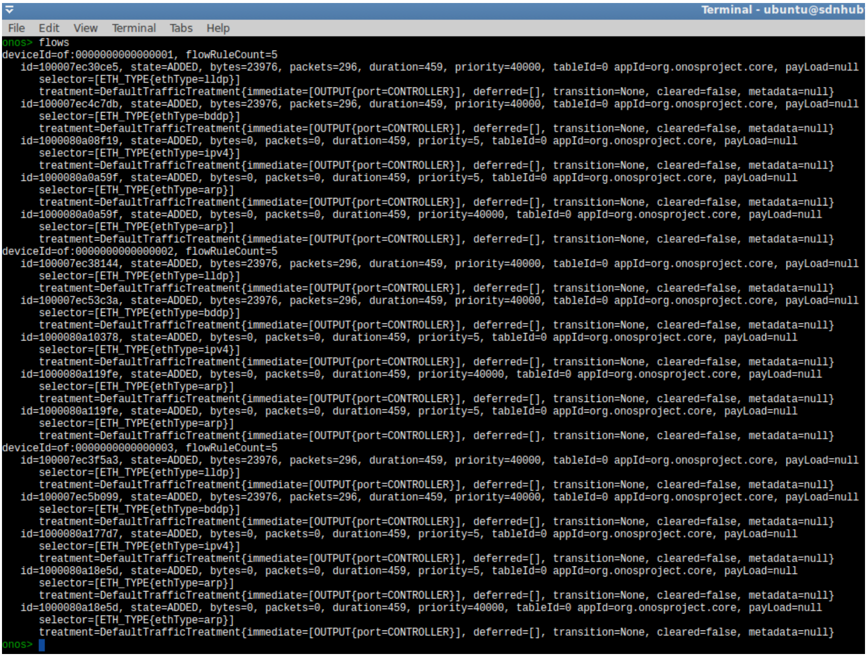


Fig. 7 – The list of flows

From the host h3 send the command "*ping*" to the host h1 in the *Mininet* window (Fig. 8).

To see the contents of the flow tables on all switches, execute the *Mininet* command:

```
mininet> dpctl dump-flows
```

To check ARP tables on each host, execute the *Mininet* "*arp*" command. For instance, to show the ARP table for host h1, enter the following command:

```
mininet> h1 arp
```

To clear all flow tables on all switches, enter the *Mininet* command:

```
mininet> dpctl del-flows
```

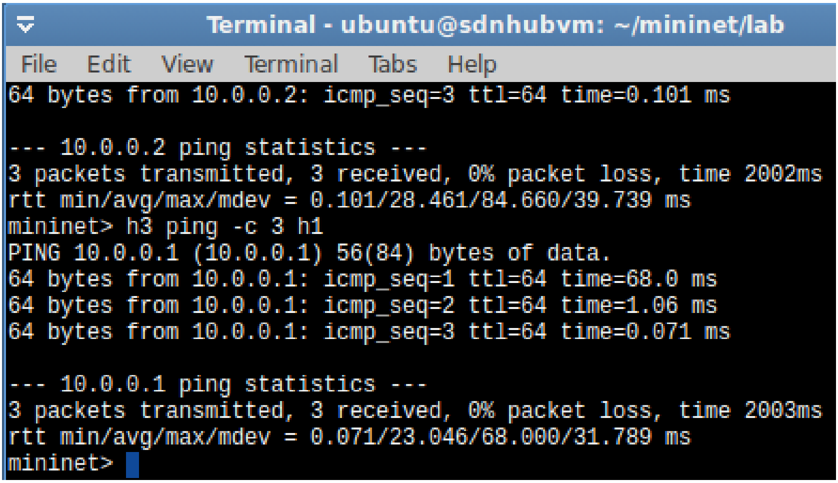


Fig. 8 – *Mininet* "*ping*" command

Analyze how the rules change. On switches 1 and 3, two new routing rules should appear (Fig. 9).

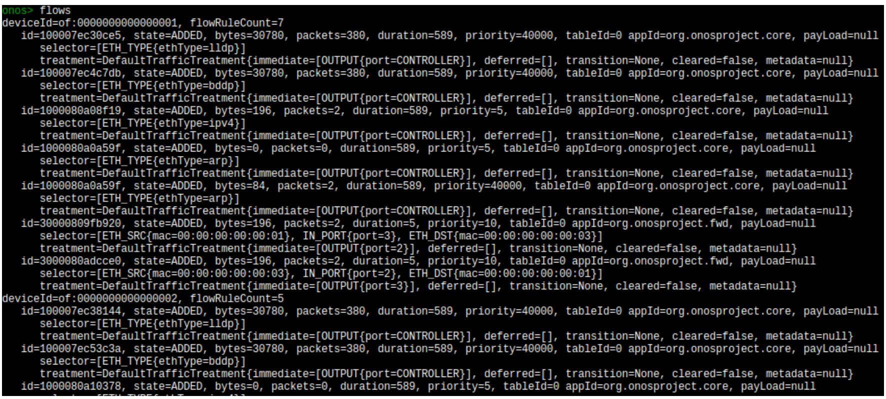


Fig. 9 – The list of routing rules

Step 5. Build a network in *Mininet* in accordance with your personal task and discover basic network management operations with *ONOS*.

An example below creates a 3-switch topology connected in a loop. A host is connected to each switch.

```
#!/usr/bin/python
from mininet.topo import Topo
class triangleTopo( Topo ):
  "Create a custom network and add nodes to it."
  def __init__( self ):
    #setLogLevel( 'info' )
    # Initialize topology
    Topo.__init__(self)
    #info( '*** Adding hosts\n' )
    h1 = self.addHost( 'h1' )
    h2 = self.addHost( 'h2' )
    h3 = self.addHost( 'h3' )
    #info( '*** Adding switches\n' )
    nodeA = self.addSwitch('s1')
    nodeB = self.addSwitch('s2')
    nodeC = self.addSwitch('s3')
    #info( '*** Creating links\n' )
    self.addLink( nodeA, nodeB )
    self.addLink( nodeB, nodeC )
    self.addLink( nodeC, nodeA )
    self.addLink( h1, nodeA )
    self.addLink( h2, nodeB )
    self.addLink( h3, nodeC )
topos = {'mytopo': (lambda: triangleTopo() ) }
```

Explore *OpenFlow* control messages and how flow tables are updated on the switches.

Explore how the other stock *ONOS* components work individually and in combination with other components or applications.

1. *ONOS* has a web-based GUI. To launch it you should click on the provided *ONOS* GUI icon (Fig. 10).

Fig. 10 – *ONOS* GUI icon

2. Login as user `onos` with password `rocks`;
3. Open the browser and go to the following link:

```
http://localhost:8181/onos/ui/index.html
```

4. Click “h”. We will see our hosts. As you will see, host two are invisible to us due to we did not ping it.

5. Ping the h2 host in the *Mininet* window (Fig. 11).

```
> h1 ping -c 3 h2
```

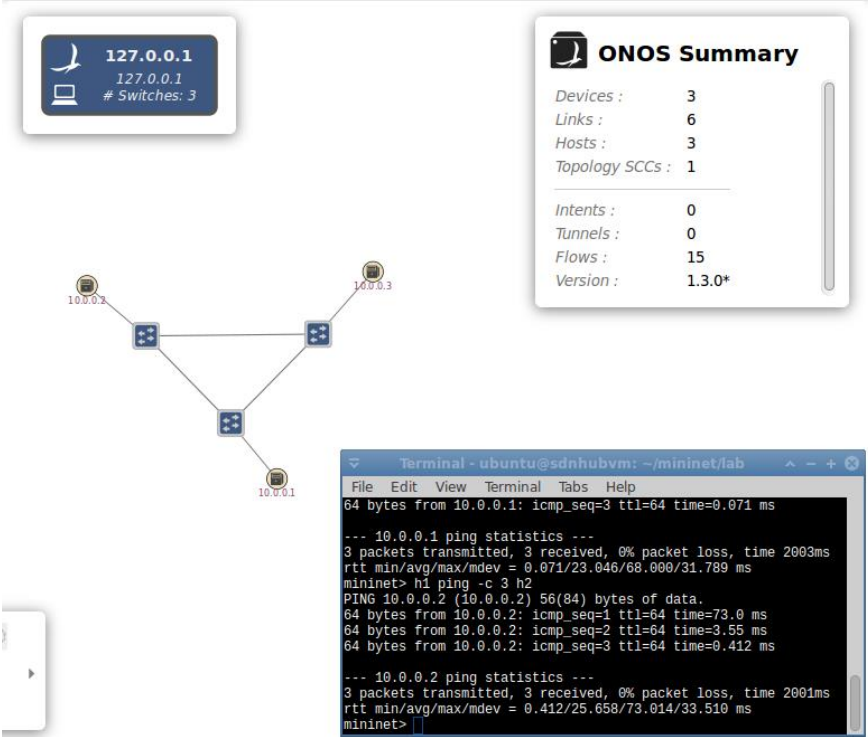


Fig. 11 – Simple tree with three switches and three hosts

For quick help press “\”.

6. On *ONOS* GUI press “F” and click to “s1” for seeing traffic flow (yellow line in Fig. 12).

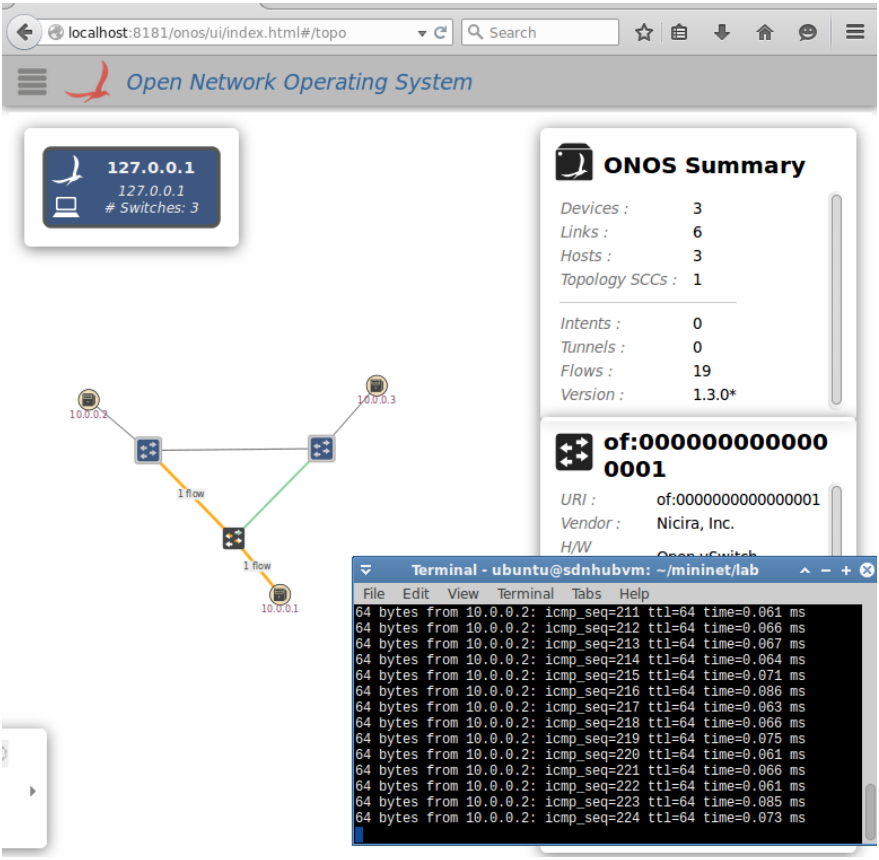


Fig. 12 – Traffic flow

In this lab, the *ONOS* controller is running on the same virtual machine that all the emulated switches and hosts created by *Mininet* are running on.

7. Run “h1 ping h3”.

Ctrl-c is an interrupt hotkey.

On *ONOS* command line type “stop onos-app-fwd”.

As it can be seen from Fig. 13, the packet forwarding has been stopped.

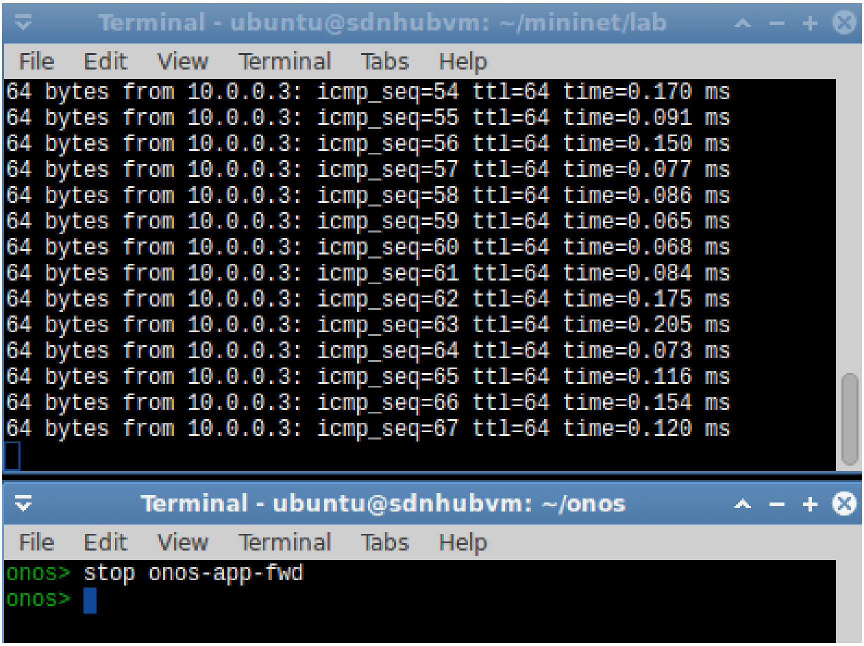


Fig. 13 – Stopping the packet forwarding

8. Restore packet flow (Fig. 14).

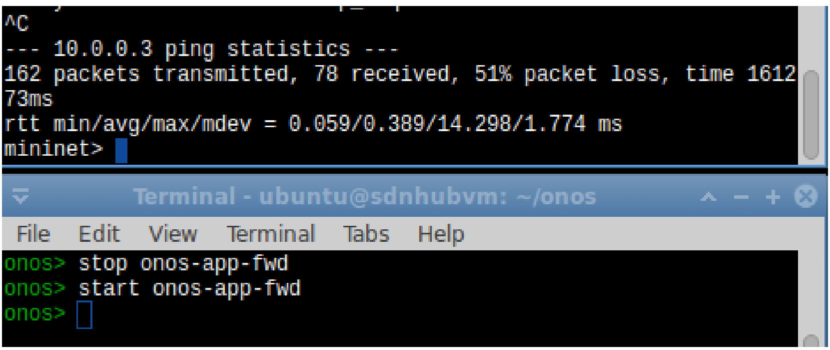


Fig. 14 – Command to restore packet flow

The command “logout” is used for *ONOS* stopping.

1.4. Requirements to the content of the report

Report should contain 5 sections: Introduction (I), Methods (M), Results (R), and Discussion (D)

– (I): background / theory, purpose and discovery questions;

– (M): complete description of the software, and procedures which was followed in the experiment, experiment overview, Fig. / scheme of testing environment, procedures;

– (R): narrate (like a story), tables, indicate final results;

– (D): answers on discovery questions, explanation of changes in traffic flow, conclusion / summary.

1.5. Control questions:

1. For what purpose *ONOS* is used?
2. List main command options available for *ONOS*.
3. How the rules changed when the host "h3" send the command "ping" to the host "h1"? Why?
4. What changes did you observe at your virtual network?
5. How to build a network in mininet?
6. What are the network management operations with *ONOS*?

1.6. Recommended literature:

1. Onosprojectorg. 2018. [Online]. Available: http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf. [Accessed: 4 Feb. 2018].

2. "Basic ONOS Tutorial" *Wikionosprojectorg*. 2018. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial. [Accessed: 4 Feb. 2018].

3. Anadiotis, A.-C. G., Galluccio, L., Milardo, S., Morabito, G. and Palazzo, S., 2015, Towards a software-defined Network Operating System for the IoT. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). 2015. doi 10.1109/wf-iot.2015.7389118. IEEE.

4. Oracle VM VirtualBox. 2018. [Online]. Available: https://www.oracle.com/technetwork/server-storage/virtualbox/overview/index.html [Accessed: 4 Feb. 2018].

5. SDN Tutorial. 2018. [Online]. Available: http://yuba.stanford.edu/~srini/tutorial/SDN_tutorial_VM_32bit.ova. [Accessed: 4 Feb. 2018].

6. ONOS is the only open source controller providing: [Online]. Available: https://onosproject.org/. [Accessed: 4 Feb. 2018].

7. Mininet: An Instant Virtual Network on your Laptop (or other PC) [Online]. Available: http://mininet.org/. [Accessed: 4 Feb. 2018].

8. Appendix A: CLI commands [Online]. Available: https://wiki.onosproject.org/display/ONOS/Appendix+A+%3A+CLI+commands. [Accessed: 4 Feb. 2018].

9. Basic ONOS Tutorial [Online]. Available: https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial. [Accessed: 4 Feb. 2018].

Laboratory work 2

QUALITY OF SERVICE IN SDN NETWORK SCENARIO USING POX CONTROLLER

Goal and objectives: In this laboratory work we will focus on the general principles of quality of services (QoS) in SDN with POX.

Learning objectives:

– study main principles of QoS in SDN;

– study the possibilities of managing SDN with POX controller.

Practical tasks:

– acquire practical skills of network traffic management in SDN;

– acquire practical skills of working with open Vswitch with POX controller.

Exploring tasks:

– exploring the possibilities of Open vSwitch.

Setting up

In preparation for laboratory work it is necessary:

– to clear the goals and mission of the research;

– to study theoretical material contained in this manual, and drill down to [1]-[6];

– to familiarize oneself with the main procedures and specify the exploration program according to defined task.

Recommended software and resources:

– SDNHub
http://yuba.stanford.edu/~srini/tutorial/SDN_tutorial_VM_32bit.ova;

– OracleVM VirtualBOX:
https://www.oracle.com/technetwork/server-storage/virtualbox/overview/index.html.

2.1. Synopsis

In this lab we will assess and analyze the Quality of Service of POX Controller in SDN. Furthermore, we outline the potential challenges and open problems that need to be addressed further for better and complete QoS abilities in SDN/OpenFlow networks. Additional information about available bandwidth measurement in SDN and simulation in SDN network scenario using the POX Controller can be found in [7] and [8] respectively.

2.2. Brief theoretical information

The one of essential features required of IoT networks is to ensure high reliability and Quality of Services (QoS). QoS is typically defined as an ability of a network to provide the required services for selected network traffic. The primary goal of QoS is to provide priority with respect to QoS parameters including but not limited to: bandwidth, delay, jitter, and loss [6].

QoS metrics.

There are the following QoS metrics applicable for the SDN analysis:

1. Network Throughput (*NT*).

NT is the number of data packets delivered from source to destination per unit of time:

$$NT = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i}{t_i}, \tag{1}$$

where b_i denotes a total amount of data, t_i is a time taken for destination to get the final packet, n is total number of application traffic.

2. Packet Delivery Ratio (*PDR*).

PDR is a ratio of the number of packets received (N_{PR}) by the destination to the number of packets send (N_{PS}) by the source:

$$PDR = \frac{N_{PR}}{N_{PC}}. \tag{2}$$

3. Packet Loss (*PL*).

PL is the measure of number of packets dropped by nodes due to various reasons.

$$\mathrm{PL} = N_{PS} - N_{PR}. \tag{22.3}$$

4. Average end-to-end delay (*EED*).

EED is defined as average time taken by data packets to propagate from source to destination across Ad hoc network.

$$EED = \sum (\text{arrived time} - \text{sent time}). \qquad (4)$$

Performance tests.

1. Network connectivity test

Pingall stamen checks the connectivity among the created network. Connectivity among the hosts ensures the data transfer is possible among them. Wget feature of Linux has been used for sending files among the reachable hosts. Fig. 15 depicts connectivity among the hosts depending on the congestion state at the particular time. Frame 1 shows that h1 and h3 has connectivity to all other hosts, whereas from h2 there is no connectivity to h1and h4 has no connectivity to h1. Frame 2 shows that h2 has no reachability to h3 and h3 has no reachability to h4.

```
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> X h3 h4
h3 -> h1 h2 h4
h4 -> X h2 h3
*** Ping: testing ping reachability
```

Fig. 15 – Pingall reachability test

2. SDN available bandwidth measurement

Bandwidth utilization in a network serves as a key method for measuring QoS of network. Available bandwidth is an important component for both service provider and application perspective [7].

Fig. 16 shows the two bandwidth frames taken at different amount of time. It measures the bandwidth between h1 and h4 at different available capacity links.

```
Testing bandwidth between h1 and h4
*** Iperf: testing TCP  bandwidth between h1 and h4
*** Results: ['3.03 Mbits/sec' , '3.15 Mbits/sec']
testing bandwidth between h1 and h4
```

```
*** Iperf: testing TCP  bandwidth between h1 and h4
*** Results: ['3.52 Mbits/sec' , '3.66 Mbits/sec']
```

Fig. 16 – Available bandwidth measurement

3. Packet loss and delay measurement

Packet loss ratio and packet delay will benefit many users and operators of network applications. Fig. 17 shows the example of packet loss and delay in the SDN network.

```
*** Adding links:
(10.00Mbit 5ms delay 2% loss)  (10.00Mbit 5ms delay 2% loss)  (h1, s1)
(10.00Mbit 5ms delay 2% loss)  (10.00Mbit 5ms delay 2% loss)  (h2, s1)
(10.00Mbit 5ms delay 2% loss)  (10.00Mbit 5ms delay 2% loss)  (h3, s1)
(10.00Mbit 5ms delay 2% loss)  (10.00Mbit 5ms delay 2% loss)  (h4, s1)
```

Packet loss

```
*** Results: 16% dropped (10/12 received)
```

Fig. 17 – Packet loss and delay in the SDN network

2.3. Execution order and discovery questions

The simulation scenario consists of two OpenFlow switches (S1 and S2) connected to the three hosts (h1, h2, h3) and to the POX controller as depicted in Fig. 18.

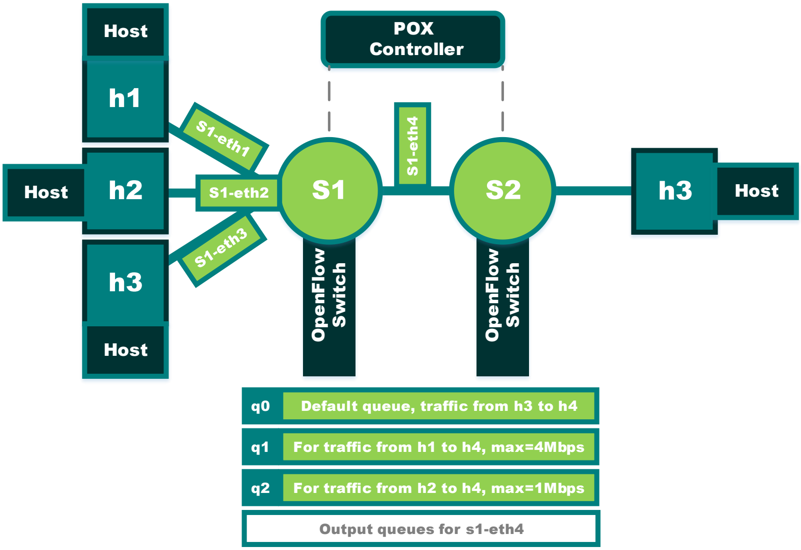


Fig. 18 – Example of network under the test

First, you need to install Linux to Oracle VM Virtual Box and deploy SDNHub ova image as you did it in previous laboratory work.

Step 1. Configure openVswitch with POX controller.

To configure openVswitch in PC1 eth0.10 interface you can use the following commands [6]:

1. Attach PC1 eth0.10 interface (IP 192.168.10.100) to the bridge connection between openVswitch in PC1 and controller.

```
> sudo ovs-vsctl add-br br0
> sudo ovs-vsctl add-port br0 eth0.10
> sudo ifconfig br0 192.168.10.100 netmask 255.255.255.0
```

2. Attach OpenVswitch to the Controller which is in 192.168.100.30.

```
> ovs-vsctl set-controller br0 tcp:192.168.100.30:6633
```

3. To remove openVswitch bridge, connection use the following command.

```
> sudo ovs-vsctl del-br br-0
> sudo ovs-vsctl del-port br-0 eth0.10
```

4. To remove the Controller type, do the following.

```
> sudo ovs-vsctl del-controller br-0
```

5. To launch multiple controllers and *Mininet* on the same VM, run *Mininet* Script given below.

```
> c1 = net.addController('c1', controller=RemoteController, ip="127.0.0.2", port=6633)
> c2 = net.addController('c2', controller=RemoteController, ip="127.0.0.1", port=6634)
```

6. Start POX controllers.

```
> ./pox.py --port=6633 MyScript.py
> ./pox.py --port=6634 MyScript.py
```

Step 2. Build the network from the Fig. 15.

```
> sudo mn --custom lab2.py --topo mytopo –mac
```

Step 3. Call command prompt h1 and h2.

```
> xterm h1 h2 h3 h4
```

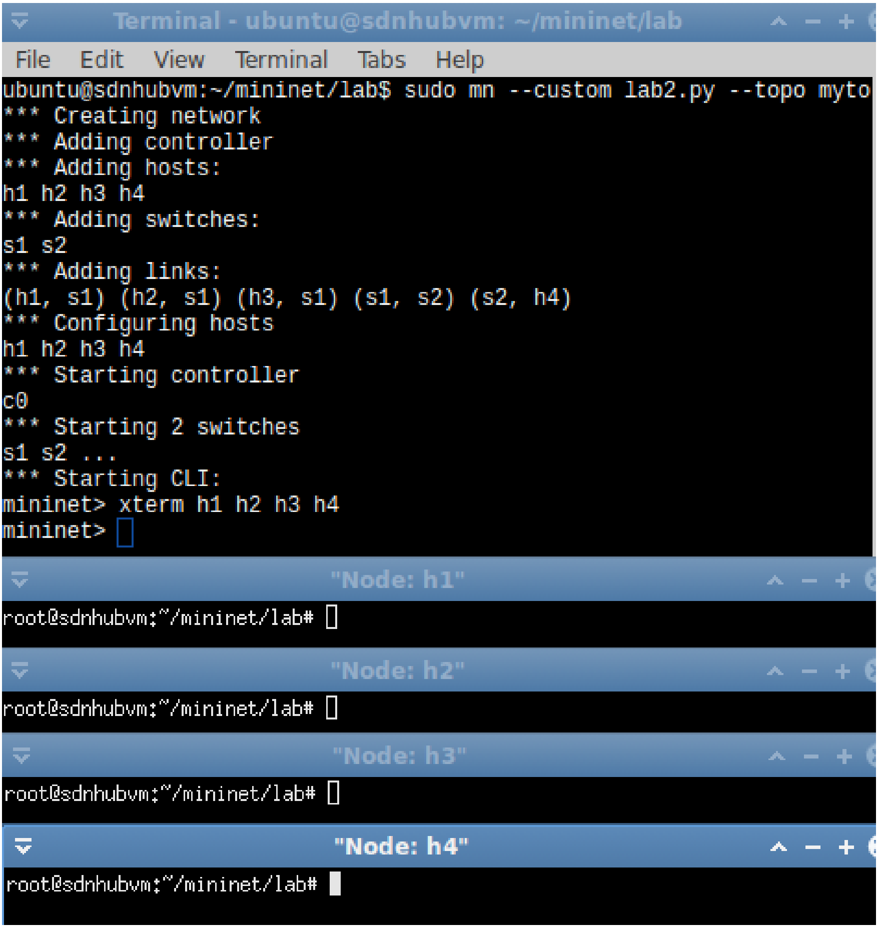


Fig. 19 – Creating the network

Step 4. In h4 windows, start Iperf servers at port 4000, 5000, 6000 respectively.

```
> iperf -s -p 4000&
iperf -s -p 5000&
iperf -s -p 6000
```

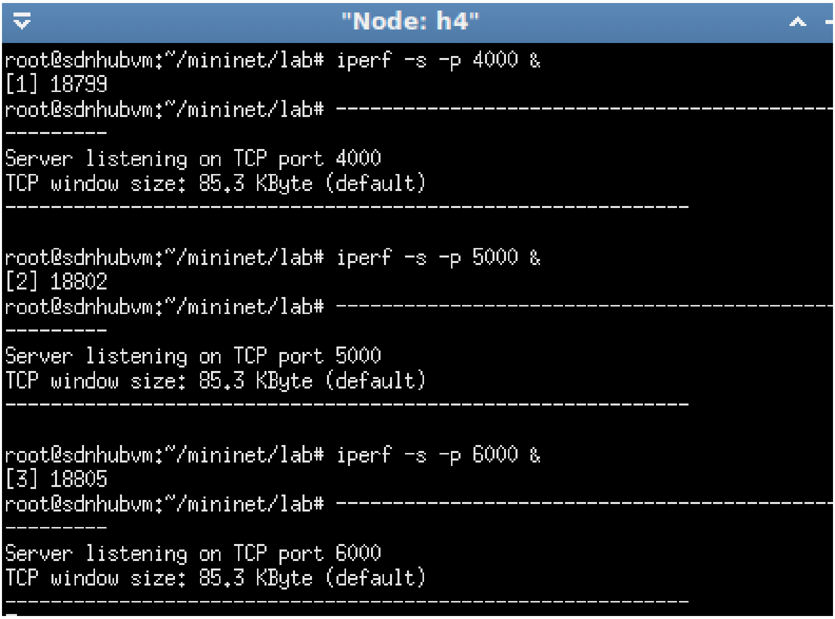


Fig. 20 – Server listening

Step 5. Conduct different tests to check the performance of SDN network. Besides the performance tests, you can use the following.

1. Node h1. Test the bandwidth between h1 and h4 (no other background traffic)

```
> iperf -c 10.0.0.4 -p 4000
```

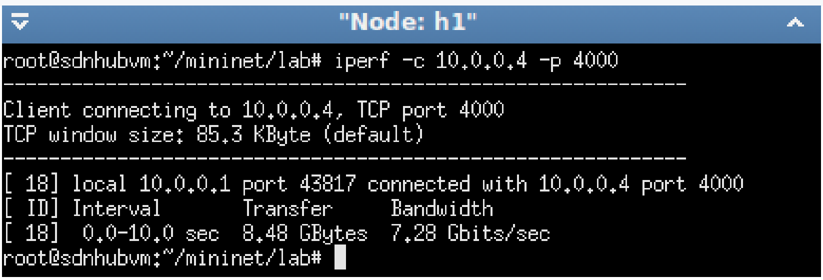


Fig. 21 – Test the bandwidth between h1 and h4

2. Node h2. Test the bandwidth between h2 and h4 (no other background traffic).

```
> iperf -c 10.0.0.4 -p 5000
```

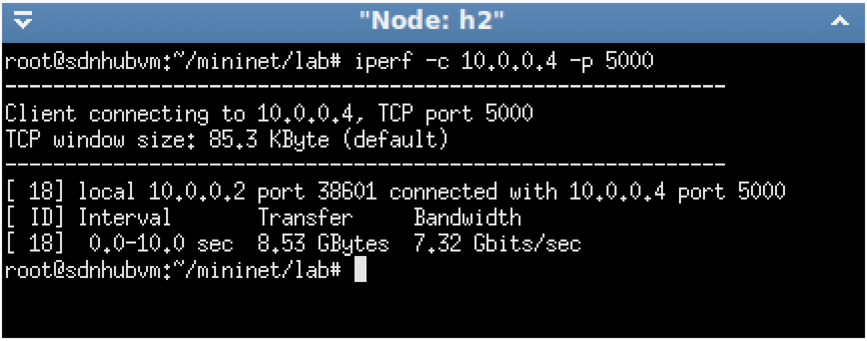


Fig. 22 – Test the bandwidth between h2 t and h4

3. Test the bandwidth between h3 and h4 (no other background traffic).

```
> iperf -c 10.0.0.4 -p 6000
```

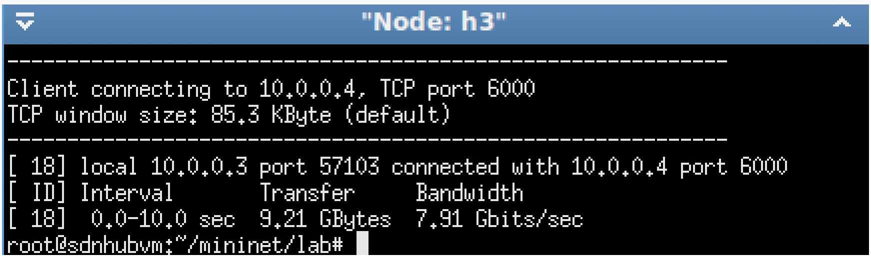
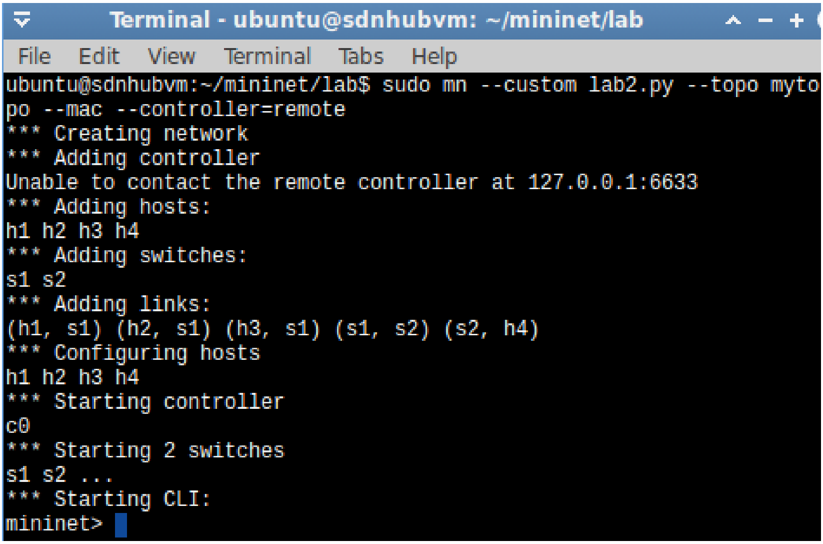


Fig. 23 – Test the bandwidth between h3 and h4

The measured bandwidth will be around 7.91 Gbits/sec. They depend from the emulation environment, such as CPU and working load.

Step 6. Restart *Mininet*.

```
> sudo mn --custom lab2.py --topo mytopo –mac --controller remote
```



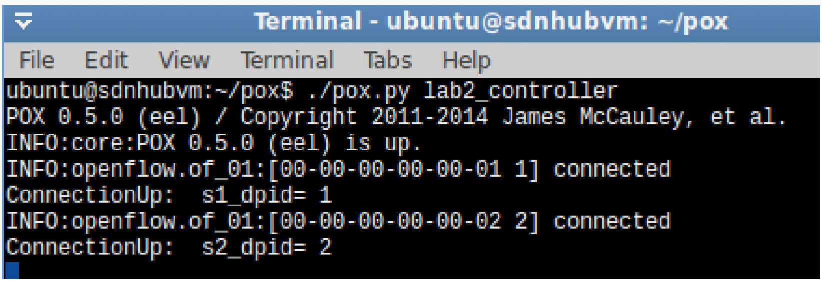
```
Terminal - ubuntu@sdnhubvm: ~/mininet/lab
File  Edit  View  Terminal  Tabs  Help
ubuntu@sdnhubvm:~/mininet/lab$ sudo mn --custom lab2.py --topo mytopo --mac --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (s1, s2) (s2, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```

Fig. 24 – Restart Mininet

Step 7. Put lab2_controller.py to home/ubuntu/pox/ext/.

Step 8. Start POX controller.

```
> cd pox
> ./pox.py lab2_controller
```



```
Terminal - ubuntu@sdnhubvm: ~/pox
File  Edit  View  Terminal  Tabs  Help
ubuntu@sdnhubvm:~/pox$ ./pox.py lab2_controller
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
ConnectionUp:  s1_dpid= 1
INFO:openflow.of_01:[00-00-00-00-00-02 2] connected
ConnectionUp:  s2_dpid= 2
```

Fig. 25 – Start POX controller

Step 9. Set up Open vSwitch for queues. Create a linux-htb QoS record that points to a few queues and use it on eth4.

Step 10. Create three queues for s1-eth4, i.e. q0, q1, and q2 and to set the rate for each queue using ovs-vsctl.

To do this, enter the following command (Fig. 26):

```
> sudo ovs-vsctl -- set Port s1-eth4 qos=@newqos -- --id=@newqos create QoS type=linux-htb other-config:max-rate=10000000 queues=0=@q0,1=@q1,2=@q2 -- --id=@q0 create Queue other-config:min-rate=1000000000 other-config:max-rate=1000000000 -- --id=@q1 create Queue other-config:min-rate=4000000 other-config:max-rate=4000000 -- --id=@q2 create Queue other-config:min-rate=100000 other-config:max-rate=1000000
```

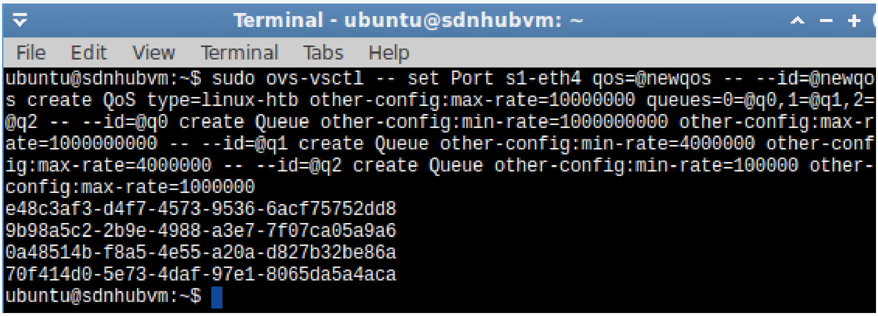


Fig. 26 – Creating queries for s1-eth4

More detailed information about Open vSwitch is given here: http://www.openvswitch.org/support/dist-docs/ovs-vsctl.8.txt.

Node h4.

```
> iperf -s -p 4000 &
iperf -s -p 5000 &
iperf -s -p 6000
```

Node h1.

```
> iperf -c 10.0.0.4 -p 4000
```

Node h2.

```
> iperf -c 10.0.0.4 -p 5000
```

Node h3.

```
> iperf -c 10.0.0.4 -p 6000
```

As you can see (Fig. 27), the bandwidth between the hosts have been changed. This is a result of performed operations.

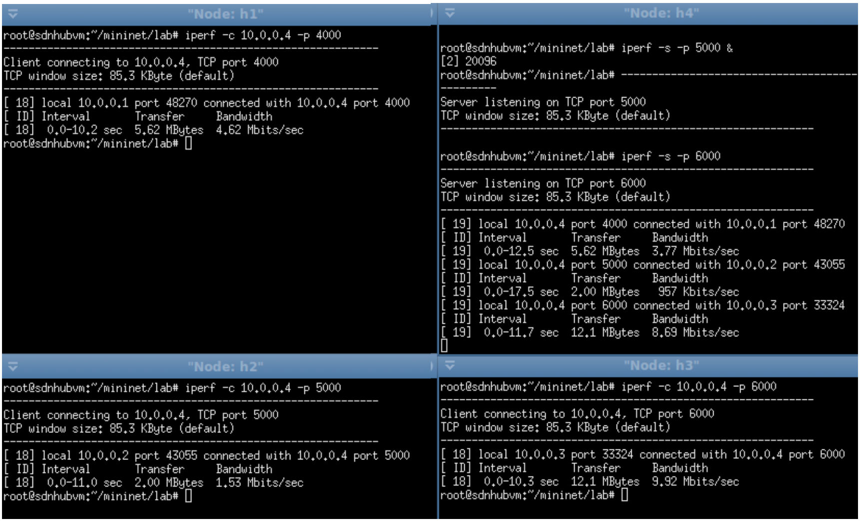


Fig. 27 – New values of the bandwidth between the hosts

After executing all available performance tests, analyze the possible changes and their causes.

2.4. Requirements for the content of the report

Report should contain 5 sections: Introduction (I), Methods (M), Results (R), and Discussion (D)

– (I): background / theory, purpose and discovery questions

– (M): complete description of the software, and procedures which was followed in the experiment, experiment overview, procedures

– (R): narrate (like a story), code for your assignment, Fig. of attack graph with marked the attacker path;

– (D): answers on discovery questions, explanation of results, conclusion / summary.

2.5. Control questions:

1. What network performance metrics in SDN do you know?
2. How to set up Open vSwitch for queues?
3. What are the key tests for measuring QoS of network with POX?
4. How to check the performance of SDN network?
5. What parameters does measured bandwidth depend on?
6. How to perform network connectivity test?
7. How to obtain measure of connectivity among the hosts depending on the congestion state at the particular time?
8. How does network architecture affect bandwidth between nodes?
9. What are the main causes of packet loss and delay measurement in SDN?
10. How to overcome issues with packet loss and delay measurement in SDN?

2.6. Recommended literature:

1. "Basic Configuration – Open vSwitch 2.12.90 documentation",*Docs.openvswitch.org*, 2019. [Online]. Available: http://docs.openvswitch.org/en/latest/faq/configuration/. [Accessed: 13 Jun. 2019].

2. QoS on OpenFlow 1.0 with OVS 1.4.3 and POX inside Mininet, 2018. [Online]. Available: http://users.ecs.soton.ac.uk/drn/ofertie/openflow_qos_mininet.pdf [Accessed: 12- Sep. 2018].

3. Open vSwitch 2018. [Online]. Available: http://www.openvswitch.org/support/dist-docs/ovs-vsctl.8.txt [Accessed: 12 Sep. 2018].

4. "POX Controller Tutorial | SDN Hub", *Sdnhub.org*, 2019. [Online]. Available: http://sdnhub.org/tutorials/pox/. [Accessed: 23 Jun. 2019].

5. "Configure openVswitch with POX controller", *Windysdn.blogspot.com*, 2019. [Online]. Available: http://windysdn.blogspot.com/2013/10/configure-openvswitch-with-pox.html. [Accessed: 03 Jan. 2019].

6. S. Midha, K. Tripathi, “Assessing the Quality of Service of POX Controller in SDN,” *International Journal of Computational Engineering Research (IJCER)*, 2018, vol. 8, no. 8, pp. 21-25.

7. P. Megyesi, A. Botta, G. Aceto, A. Pescapè, S. Molnár, "Available Bandwidth measurement in SDN", *ACM* 978-1-4503-3739-7/16/04

8. L. R. Prete, A. Shinoda, C. Schweitzer and R. de Oliveira, "Simulation in an SDN network scenario using the POX Controller", *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014. doi: 10.1109/colcomcon.2014.6860403.

Laboratory work 3

IOT DATA STREAMING OVER SDN

Goal and objectives: In this laboratory work, we explore the behavior of a SDN when transmitting a video stream in unstable conditions. Such network parameters as low latency, processing of "volumetric" data, and minimal distortion of information packets are important for transmitting a video stream.

Learning objectives:

– study the employing SDN to control video streaming applications;

– study the means used for real-time traffic transmission.

Practical tasks:

– acquire practical skills of working with SDN and video stream;

– acquire practical skills in analysis and improving video quality-of-experience (QoE).

Exploring tasks:

– investigate behavior of a SDN when transmitting a video stream in unstable conditions;

– explore the performance of the network with an increase in the rate of packet loss information, increasing network latency.

Setting up.

In preparation for laboratory work it is necessary:

– clear the goals and mission of the research;

– study theoretical material contained in this manual, Chapter 23 of the Multibook and in [1]-[3];

– familiarize oneself with the main procedures and specify the exploration program according to defined task.

Recommended software and resources:

VLC player: https://www.videolan.org/index.ru.html;

Mininet: http://mininet.org/;

OracleVM VirtualBOX: https://www.oracle.com/technetwork/server-storage/virtualbox/overview/index.html;

SDNHub

http://yuba.stanford.edu/~srini/tutorial/SDN_tutorial_VM_32bit.ova.

3.1. Synopsis

In this lab, we will try to solve the problem of quality of video stream transmissions via SDN. In order to improve video QoE over SDN, we will (a) study various video QoE metrics; (b) assign the best available delivery node based on provisioned network conditions; and (c) dynamically change routing paths between wide area network (WAN) routers.

3.2. Brief theoretical information

When we move IoT data to the cloud, managing the cloud resources is considered as the main issue. The data-driven process can be done by batch data processing using Hadoop. However, due to the emergence of IoT technologies that generated tremendous streamed data, processing on time is needed to obtain valuable information before the data become valueless. For example, in the case when the streamed data comes from health monitoring sensors or video streaming in surveillance systems to enhance situation awareness. SDN has a potential solution to this issue. For operators, the video QoE analytics can be used for identifying most congested segments within the video delivery network to be upgraded first, for debugging ongoing QoE-related issues in the field, and for proactively preventing QoE-related complaints from content users.

When starting the network operation, a topology discovery process is performed, and it allows the controller to select the most suitable path for the transmission of the video streams. Once an OpenFlow-enabled device connects to the SDN controller, it starts a handshaking process that allows the controller to be aware of all forwarding devices in the SDN data plane.

In the SDN controller, the topology data gathered is represented by an undirected graph. The controller computes the routes among the devices based on the information contained in this network representation. As the topology of the network changes, the graph needs to be updated to correspond to the new network topology. The controller detects OpenFlow enabled devices that are joining or leaving the data plane through the OpenFlow Channel.

In this lab, we will use an SDN-based architecture to collect information about the quality of data streaming and analyzes it to provide a more accurate estimate of end-user QoE.

Measurements of video stream playback quality.

Measurements of video stream playback quality can be classified into objective and subjective metrics [1]. Objective measurements can be collected in the user video player. Subjective metrics like Mean Opinion Score (MOS) are based on the user feedback. The video quality assessment can also be used as a feedback to adjust network settings and policy enforcements. By collecting and analyzing objective QoE measurements, the MOS indicator can be predicted

QoE metrics.

There is a wide variety of video quality metrics and different flavors of QoE optimization objectives. Some objective metrics to quantify the QoE on the client side using video over HTTP could be find in [1], [3]. These metrics are:

1. Video playback start time: time taken by the player to start the playout, from the moment the stream is requested.

2. Number of interruptions: when the playback is temporarily frozen a video interruption is computed.

3. Total duration of interruptions: the sum of the duration of all interruptions (Buffering Time) during video playout. The Initial Buffering event in most cases is ignored.

3.3. Execution order and discovery questions

The simulation scenario consists of two hosts (h1 and h2) and one switch in the current *mininet* environment (Fig. 28). The link between h1 and switch is set to be lossless. The loss rate for the link between h2 and switch is set approx to 5%.

With some settings, the hosts in the *mininet* can call on the outside. In this lab, h1 and h2 will run the VLC RTP sender program, and send the video packets to the windows environment.

Note: You can change the loss parameter in this manner:

```
> net.addLink(h2, s1, cls=TCLink, bw=10, delay='1ms', loss=5)
```

1. Obtain the initial data to perform individual tasks. An example of assignment is represented in Fig. 28.

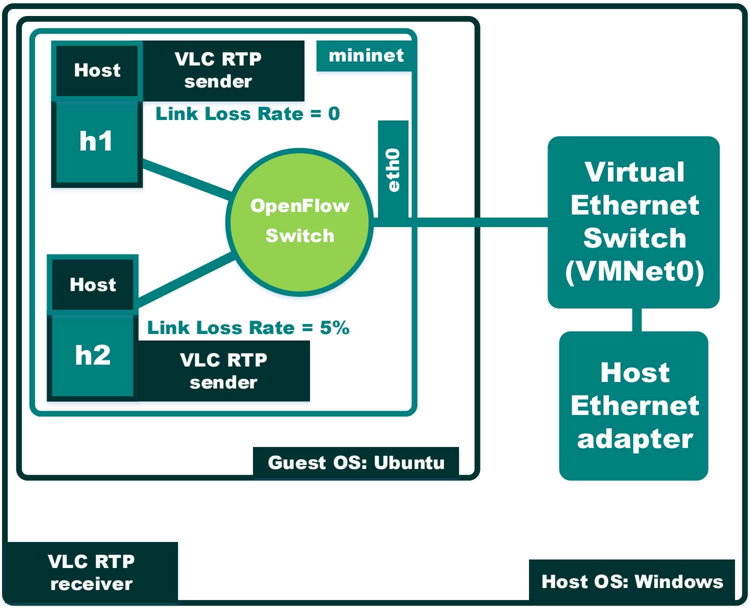


Fig. 28 – Example of network under the test

2. Download video: https://sample-videos.com/index.php#sample-mp4-video.

3. Install VLC players. For Ubuntu they can be installed using following command:

```
> sudo apt-get install vlc
```

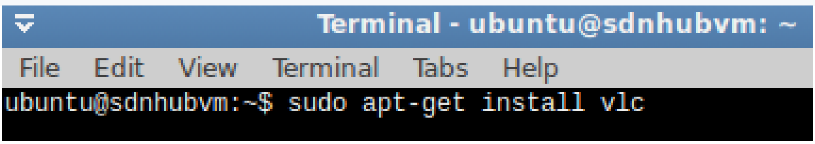


Fig. 29 – Install VLC players

4. Run a python scenario provided in Appendix D.

```
> sudo chmod +x lab3.py
> sudo ./lab3.py
```

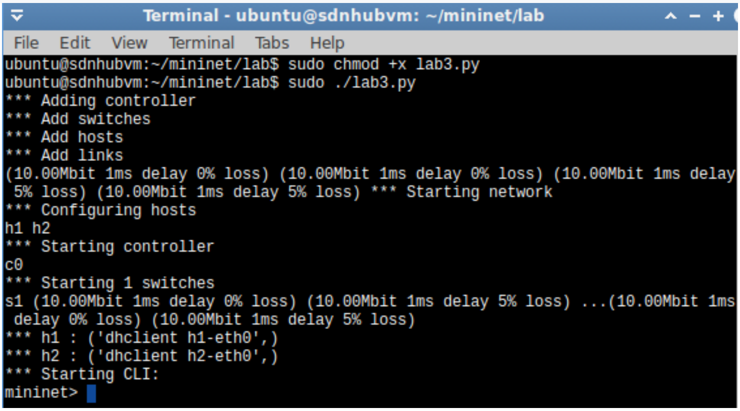


Fig. 30 – Creating a network and running a python scenarios

5. Ping external Windows host.

```
> h1 ping -c 3 192.168.1.3
```

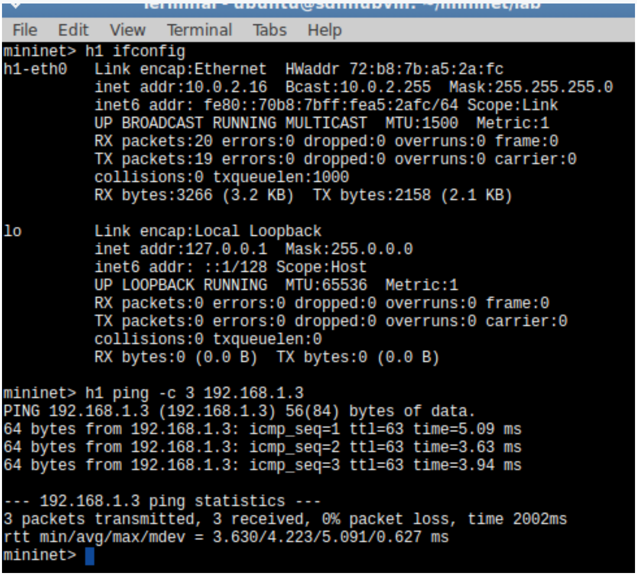


Fig. 31 – Ping the external hosts

6. Send ten ICMP packets from h2 and analyze how many of packets will be lost (usually it is about 20%).

```
> h2 ping -c 10 192.168.1.3
```

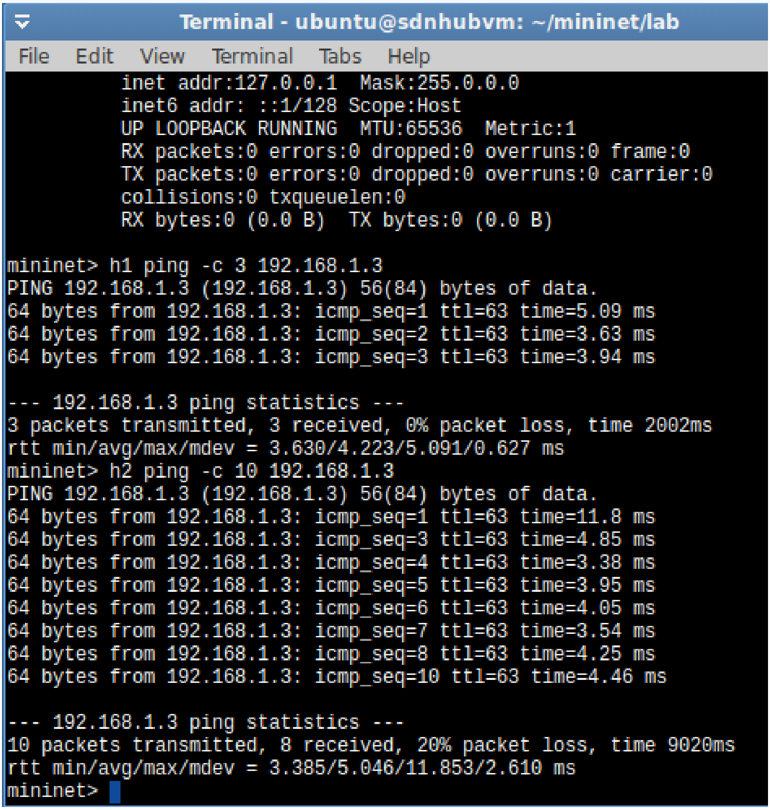


Fig. 32 – Sending ICMP packets from h2

7. Run VLC media player on receiver machine (internal).

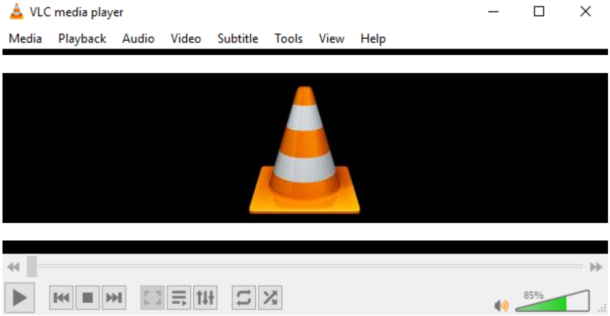


Fig. 33 – VLC player main window

8. Go to "Media" → Open Network Stream. Input "rtp://you_receiver_ip_addr_:5004". Click "Play".

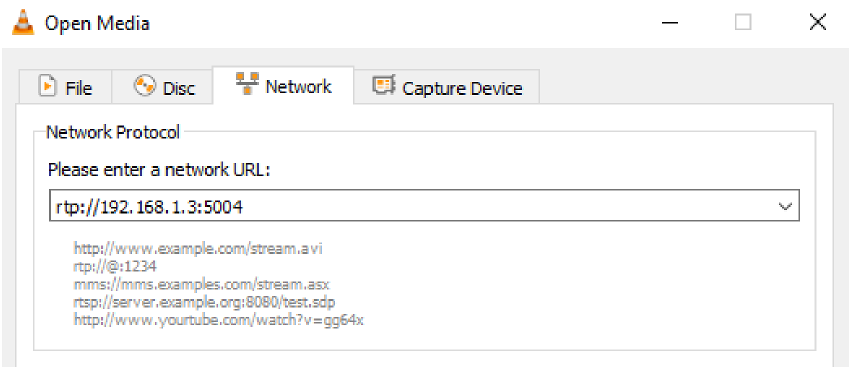


Fig. 34 – Opening network stream

9. In Mininet command line, open hosts 1 and 2.

```
> xterm h1 h2
```

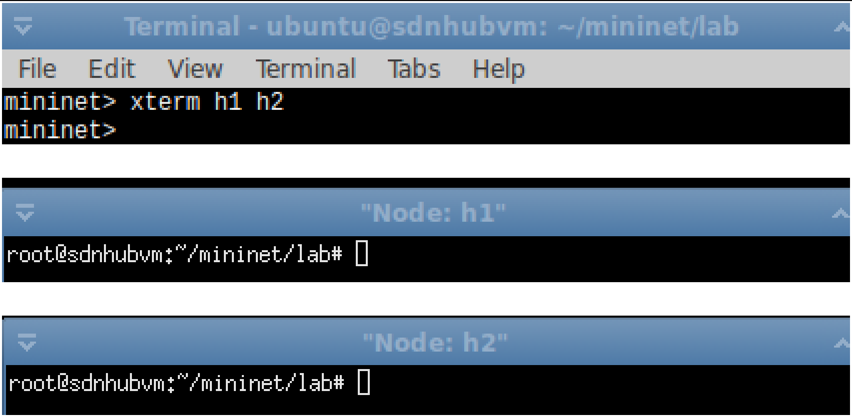


Fig. 35 – Opening h1 and h2

10. Choose h1 terminal, open VLC player. In command line: vlc-wrapper.

```
> iperf -c 10.0.0.4 -p 6000
```

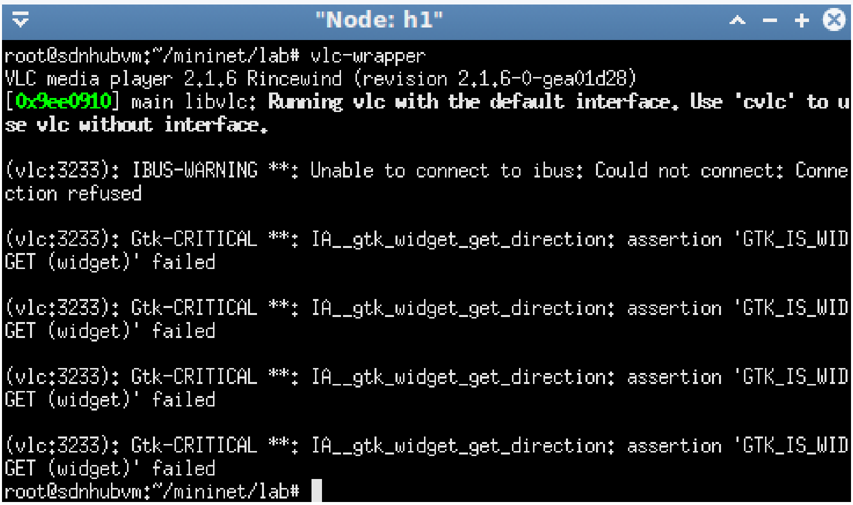


Fig. 36 – Running vlc with the default interface

11. Add the path to the video and click on Stream.

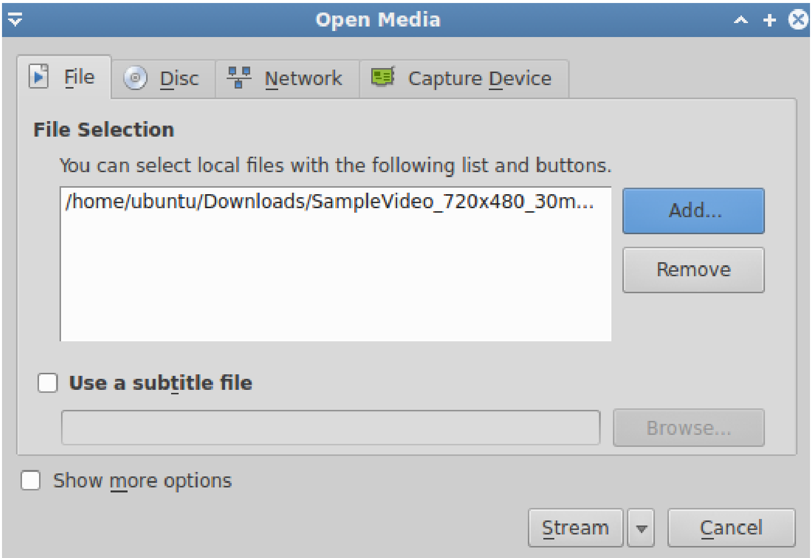


Fig. 37 – Adding the path to the video

12. Click "Next" to stream media.
13. Click "Add".

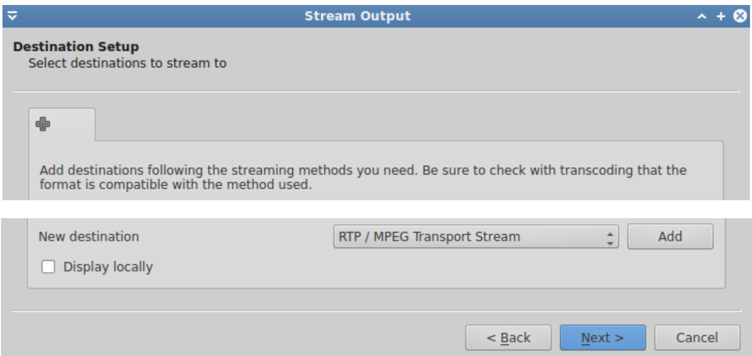


Fig. 38 – Selecting format of the video

14. Choose RTP/TS and specify the path of the recipient. Click

"Next".

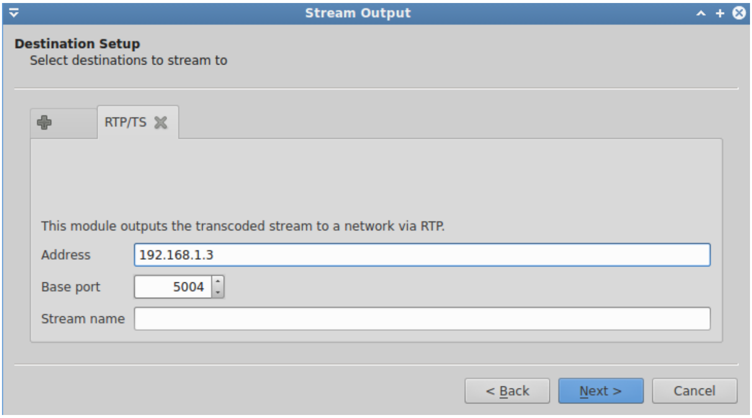


Fig. 39 – Output address

15. Uncheck with "Activate Transcoding". Click "Next".

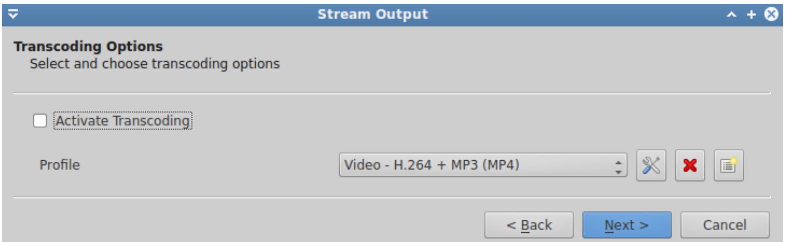


Fig. 40 – Activate transcoding

16. Click "Stream". Move on "Receiver Host".

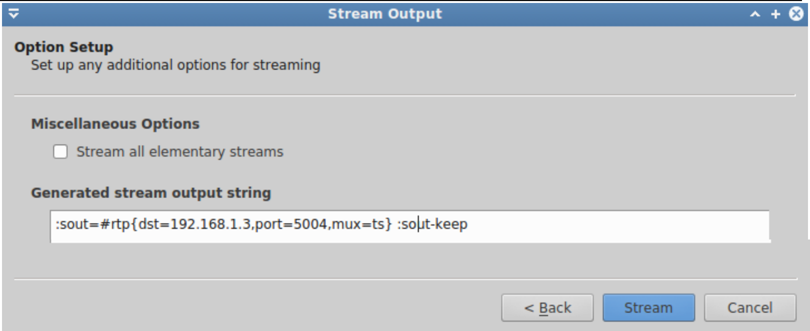


Fig. 41 – Option setup

17. As a result, a video stream from h1 host should appear (Fig. 41).

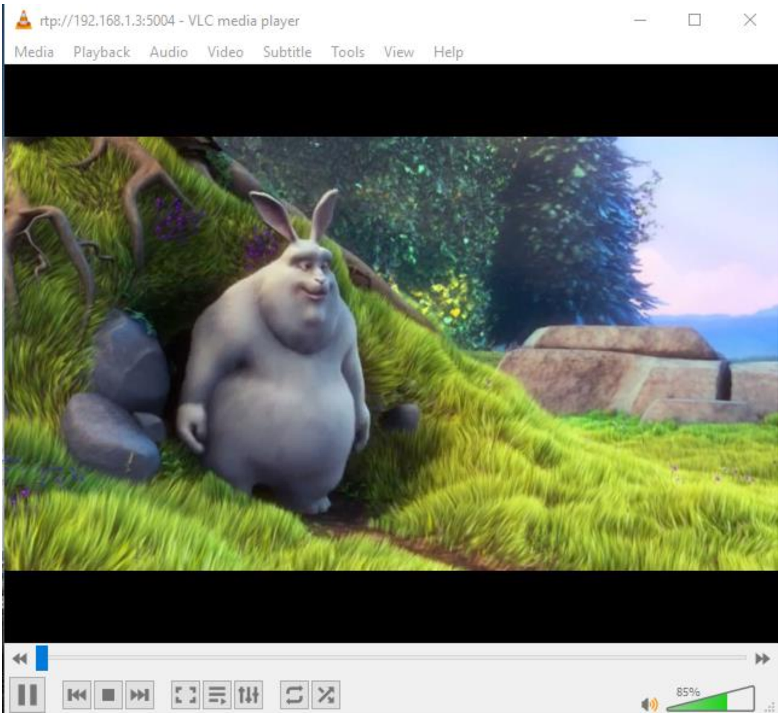


Fig. 42 – Result of video streaming from h1

18. Then stop stream on h1 and start streaming from h2. As a result, the broken video may appear (see Fig. 43).

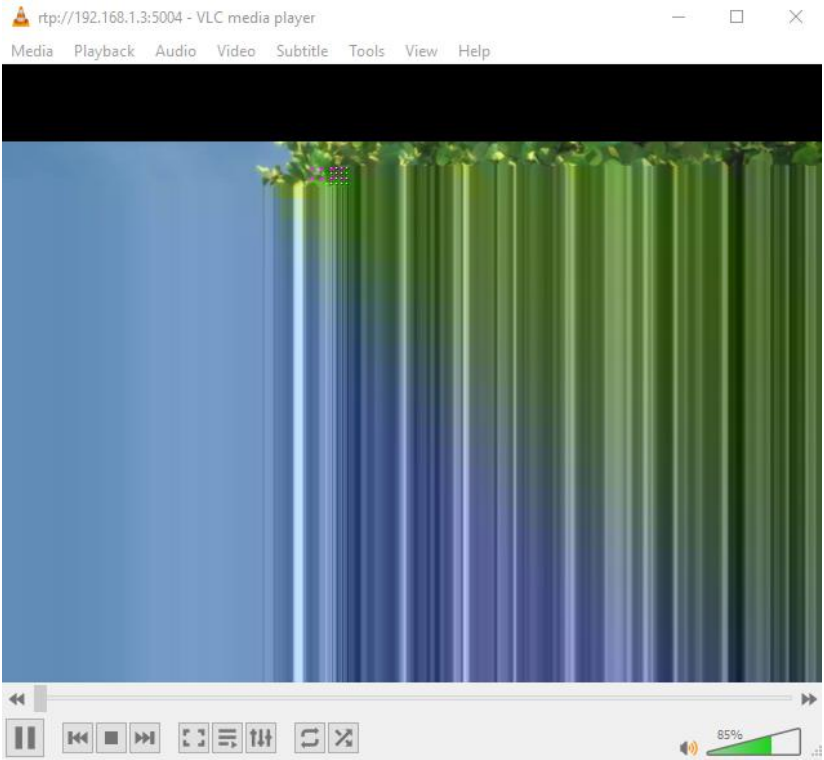


Fig. 43 – Result of streaming from h2

Do all packages reach the recipient from h2? Why?

19. To answer the question how does the loss rate of packages affect the quality of the stream conduct experiments with different percent packet loss: 1% - 10% - 25% - 50% (4 options). Conduct experiments with different quality.

3.4. Requirements to the content of the report

Report should contain 5 sections: Introduction (I), Methods (M), Results (R), and Discussion (D)

– (I): background / theory, purpose and discovery questions;

– (M): complete description of the software, and procedures which was followed in the experiment, experiment overview, Fig. / scheme of testing environment, procedures;

– (R): narrate (like a story), conditional probability tables, indicate final results;

– (D): answers on discovery questions, explanation of results, and conclusion.

3.5. Control questions:

1. From test example you can see that can see the video delivered quality is good when h1 is the sender, while the quality is poor when h2 is the sender. What do you this are the main reasons for it?
2. How does the loss rate of packages affect the quality of the stream?
3. Does network latency is increasing with an increase in the rate of packet loss information? Why?
4. How to change the percentage of packet loss between h2 and switch?
5. How SDN approach can improve the quality of video stream transmissions?
6. What criteria should meet successful solution for data streaming?

3.6. Literature:

1. P. Juluri, V. Tamarapalli, and D. Medhi, “Measurement of Quality of Experience of Video-on-Demand Services: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, Jan. 2016.

2. R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang, “Measuring the quality of experience of HTTP video streaming,” *IFIP/IEEE International Symposium on Integrated Network Management and Workshops*, vol. 1, pp. 485–492, May 2011.

3. I. Zacarias, J. Schwarzrock, L. P. Gaspary, A. Kohl, R. Q. Fernandes, J. M. Stocchero, and E. P. de Freitas, “*Employing SDN to control video streaming applications in military mobile networks,” IEEE 16th International Symposium on Network Computing and Applications (NCA), October 2017,* pp. 1-4. doi:10.1109/nca.2017.8171390.

Tutorial 1

ALGORITHMS FOR CALCULATING THE OPTIMAL POSITION OF THE SDN CONTROLLER

Goal and objectives: In this work we will focus on the analyzing methods for the location of controllers in software-defined networks as well as calculation of the optimal location of the controller using Pareto optimality method, Pareto-Optimal Resilient Controller Placement.

Learning objectives:

– study the methods for calculating the optimal placement of the SDN controller.

Practical tasks:

– acquire practical skills in working with the tools for calculating the optimal position of the SDN controller;

– conduct an analytical review of existing methods / algorithms;

– acquire practical skills in calculating the optimal placement of the SDN controller;

– draw up the report.

Setting up

In preparation for this practical training it is necessary:

– clarify the goals and mission of the research;

– study theoretical material contained in this manual, and in [1]-[3] (additional useful information you can find in [4]-[6]);

– familiarize oneself with the main procedures and specify the program according to defined task.

Recommended software and resources:

– Matlab 2007 to 2018;

– Pareto-Optimal Controller Placement tool (POCO):
https://github.com/lsinfo3/poco;
https://euros.informatik.uni-wuerzburg.de/public/localbackup.zip;

– Network topologies (task options for this practice work could be downloaded from http://www.topology-zoo.org/dataset.htm).

1.1. Synopsis

When considering several performance and stability indicators, there is usually not the only best solution for controller placement, but there is a trade-off between these indicators. For this practical lesson, the platform for flexible placement based on the Pareto Optimal COntroller (POCO) is used, which provides the network operator with the entire Pareto-optimal placement. POCO-PLC toolset facilitates the analysis and optimization of the controller placement in SDN networks under dynamic conditions.

1.2. Execution order

Step 1. Install Matlab;

Step 2. Download POCO from https://github.com/lsinfo3/poco;

Step 3. Deploy POCO to Matlab;

Step 4. Launch POCO and calculate the optimal location of the SDN controller. To use POCO PLC, follow the steps given below.

1. Extract the localbackup folder (https://euros.informatik.uni-wuerzburg.de/public/localbackup.zip) to the POCO root folder (i.e., the localbackup folder should be in the same folder as poco_GUI.m). Each of these CSV files contains RTT values for each pair of nodes in the planetlabV2.topo.mat topology for a given timestamp.
2. From http://www.topology-zoo.org/dataset.html download network model (select variants with many of network devices) in GraphML format.
3. Start Matlab. Open POCO folder (change destination).
4. Launch POCO by running poco_GUI in MATLAB.
5. From the menu, select POCO PLC → Start POCO PLC.
6. In the opened explorer window, specify the path to your file. As a result, you can see your network (Fig. 44).

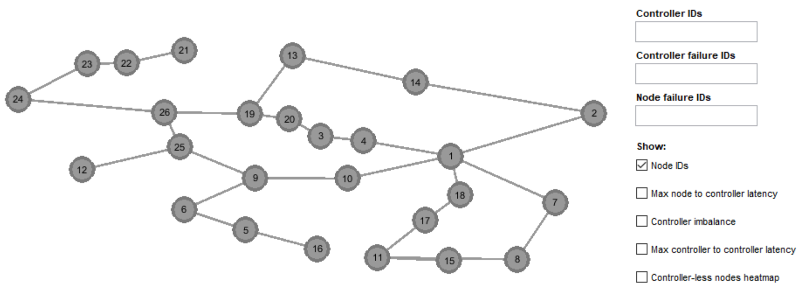


Fig. 44 – GTS Poland network (Europe)

Scenario configuration will be defined now. It includes the number of controllers for failure tree.

7. From the menu, select Placements → Calculate placements → Failure free → k = ... In the example below (see Fig. 45), we placed different numbers of controllers.

8. In Pareto-plot, click on a placement to activate it.

9. From the menu, select POCO PLC → Start Planetlab Plot Loop.

To get started with custom PLC scripts, check the PLCPlotLoop function in poco_GUI.m and adapt the code in the code / * PLC.m files.

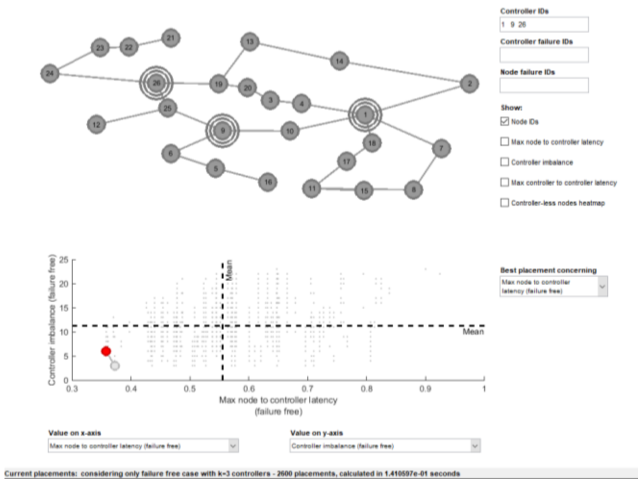


Fig. 45 – Controller placements

Step 5. Analyze various network configurations and the quality of the algorithm. To do it you can use system reconfiguration changing the values of controller imbalance, max node of controller latency, etc.

Do the substeps given below.

1. Click on "controller imbalance" and choose k = 3. You will see high-loaded sections of the network (areas in Fig. 46 marked with red). The more nodes the controller controls, the higher the load on that controller is. When the number of requests from the node to the controller in the network increases, the probability of additional delays due to queuing up to the controller increases.

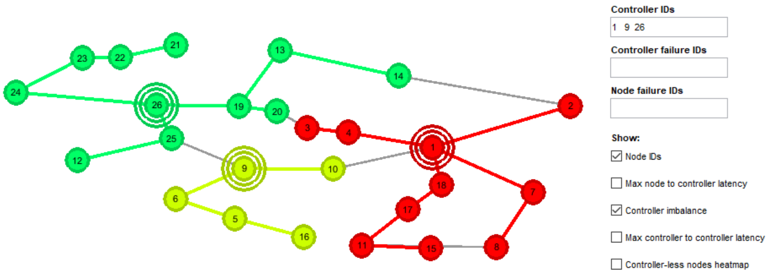


Fig. 46 – Result of changing values of the controller imbalance

Thus, in scenarios where the nodes often communicate with their controller, it is necessary that the distribution between the nodes of the controller is well balanced.

2. Click "max node controller latency".

As can be seen from Fig. 47, if both indicators of delay and reliability are taken into account at once, then usually there is no single best solution for the placement of the controller, but instead a compromise.

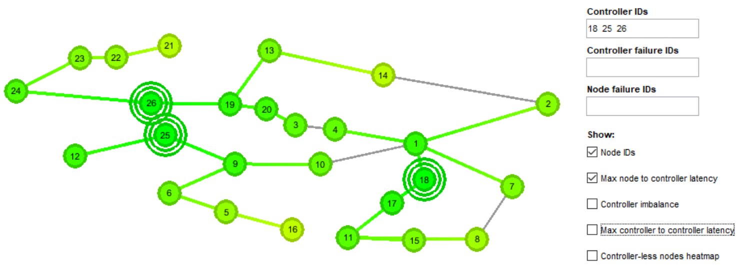


Fig. 47 – Max node controller latency

In Fig. 47, most delay-based deployment schemes currently mainly focus on transmission delay (TD) or propagation delay (PD).

3. Click “controller-less nodes heatmap”. It will indicate whether the risk of controller-less nodes exists in current architecture or not. Red nodes indicate an increased risk.

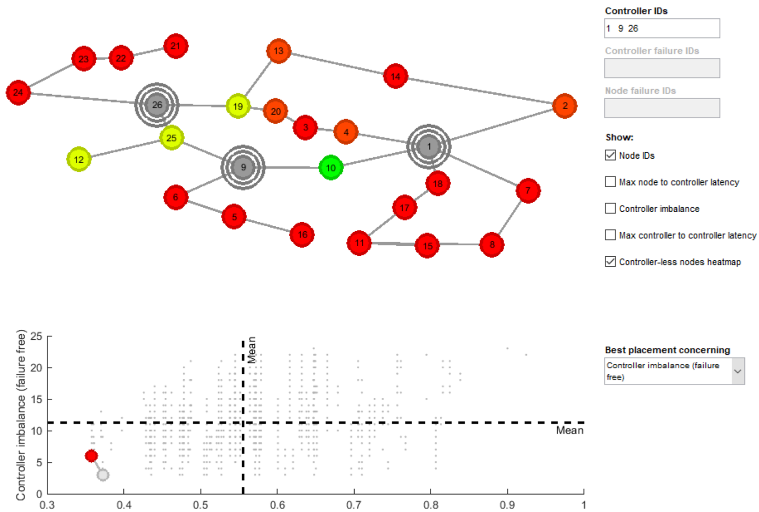


Fig. 48 – Controller-less nodes heatmap

4. Click “Max controller to controller latency”.

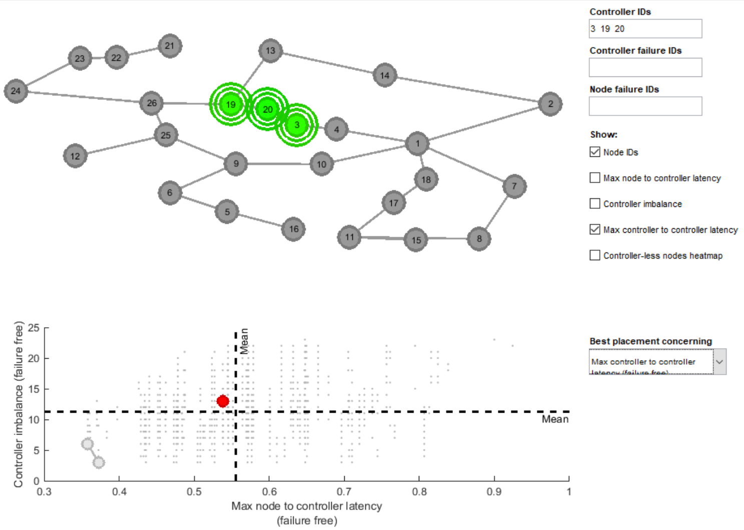


Fig. 49 – Max controller to controller latency

The graph below the network shows the Pareto optimality level, which changes when different options are selected. Vertical and horizontal convergence (dotted line) - this is the most optimal option for the location of the controller. As you can see, in the last figure, the location was changed to close to optimal, and it turned out to be the location of the controllers next to each other, but other network parameters were not taken into account, therefore it is not optimal.

Step 6. Make an analytical review of existing methods, algorithms and tools for calculating the optimal placement of the SDN controller. Draw up a report, answer the questions.

1.3. Control questions:

1. What methods for calculating the optimal location of the SDN controller do you familiar with?
2. What is Pareto optimality level regarding the SDN tasks?
3. When optimization algorithms are used in SDN?
4. What consequences can there be if the location of the SDN controller is not optimal?
5. What is controller latency? How it can be measured?
6. What are advantages / disadvantages of the proposed tool?

7. How the levels of controller imbalance influence the system performance?

8. What is the best solution for controller placement?

9. What network parameters are affected by the placement of the controller?

10. How does a controller layout change network capabilities?

1.4. Recommended literature:

1. D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, P. Tran-Gia, “Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks,” *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*. September 2013, paper no. 83, pp. 1-9. DOI: 10.1109/ITC.2013.6662939

2. Pareto efficiency. Enwikipediaorg. 2018. [Online]. Available: https://en.wikipedia.org/wiki/Pareto_optimality. [Accessed: 4 Feb. 2018].

3. D. Hock, M. Hartmann, S. Gebert, T. Zinner, and P. Tran-Gia, 2014, “POCO-PLC: Enabling dynamic pareto-optimal resilient controller placement in SDN networks,” *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2014. DOI: 10.1109/infcomw.2014.6849182.

4. K. S. Sahoo, S. Sahoo, S. K. Mishra, S. Mohanty, B. Sahoo, “Analyzing Controller Placement in Software Defined Networks,” *National Conference on Next Generation Computing and its Applications in Computer Science & Technology*, 2016.

5. “K-means and K-medoids” *Mathleacuk*. 2018. [Online]. Available:http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html. [Accessed: 4 Feb. 2018].

6. The Internet Topology Zoo. Topology-zooorg. 2018. Available at: http://www.topology-zoo.org/dataset.html. [Accessed: 4 Feb. 2018].

7. POCO-Framework for Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks [Online]. Available: https://github.com/lsinfo3/poco. [Accessed: 4 Feb. 2018].

8. A survey and classification of controller placement problem in SDN [Online]. Available: https://www.researchgate.net/publication/323974224_A_survey_and_classification_of_controller_placement_problem_in_SDN [Accessed: 4 Feb. 2018].

Tutorial 2

ALGORITHMS FOR LOAD BALANCING IN SDN

Goal and objectives: In this laboratory work, the http requests from different clients will be directed to different pre-defined http servers. The server is based on round-robin scheduling algorithm.

Learning objectives:

– study network load balancing methods, algorithms and tools.

Practical tasks:

– acquire practical skills in working with network load balancing tools;

– conduct an analytical review of existing methods, algorithms and tools for SDN load balancing;

– draw up the report.

Setting up

In preparation for this practical training it is necessary:

– clarify the goals and mission of the research;

– study theoretical material contained in this manual, and in [1]-[4];

– familiarize oneself with the main procedures and specify the program according to defined task.

Recommended software and resources:

– Mininet http://mininet.org/;

– OracleVM VirtualBOX: https://www.oracle.com/technetwork/server-storage/virtualbox/overview/index.html;

- SDNHub http://yuba.stanford.edu/~srini/tutorial/SDN_tutorial_VM_32bit.ova.

2.1. Synopsis

Load balancing is a significant technology and it can help save power and improve resource utilization in network. A typical load balancing technique is to use a dedicated load balancer to forward the client requests to different servers, this technique requires dedicated hardware support which is expensive, lacks flexibility and is easy to become a single point failure. In this tutorial, we will deal with the implementation of a dynamic load balancing algorithm to distribute the different traffic flows carried by a network through the different parallel paths between source and destination. OpenFlow is the most common protocol used in SDN networks which are used to communicate the controller with all the Network Elements.

The simulation scenario consists of two http servers (h1 and h2), four http clients (h3, h4, h5, h6), one POX controller and one switch in the current *mininet* environment (see Fig. 50).

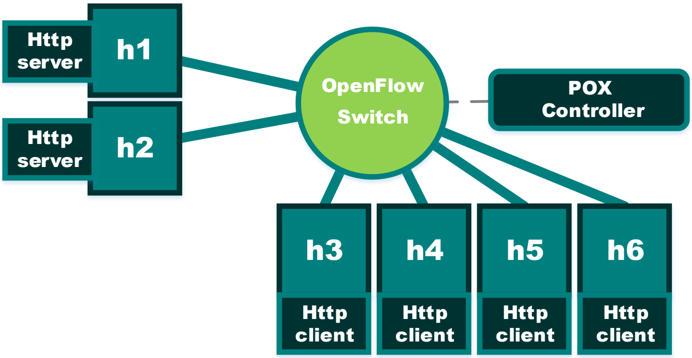


Fig. 50 – Example of network under the test

In this work, we use a POX controller to implement load balancing. *Mininet* is used to create a virtual network topology. The server balancing algorithm is assumed to be based on round-robin scheduling. The load balancing switch overwrites the destination IP address of each client packet to the destination replica address. The round-robin algorithm uses a circular queue to decide where to send the query. The load-based policy sends a request to the server with the lowest load, where the load is defined as the number of pending requests.

2.2. Execution order

Step 1. Run POX controller and debug IP load balancer.

```
> cd pox
> ./pox.py log.level -DEBUG misc.ip_loadbalancer --ip=10.0.1.1 -servers=10.0.0.1, 10.0.0.2
```

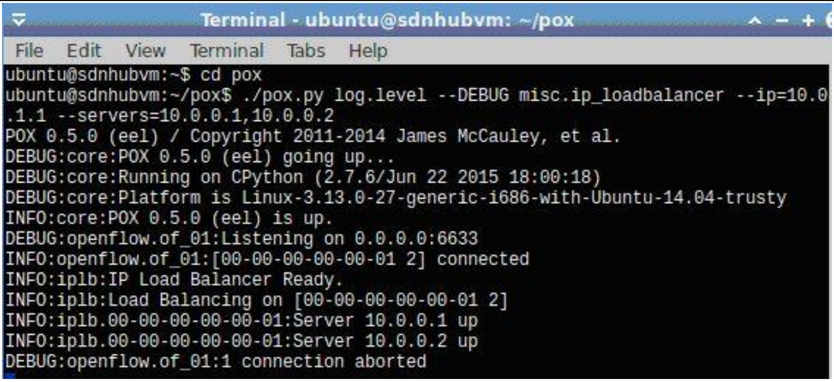


Fig. 51 – Run POX controller

Step 2. Create a network topology in *Mininet* according to your individual task or as it depicted in Fig. 52.

```
> sudo mn --topo single,6 --mac --arp --controller=remote
```

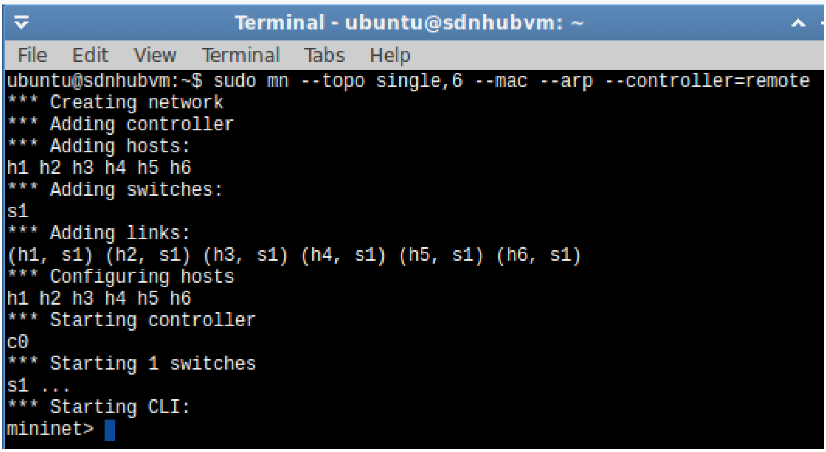


Fig. 52 – Creating a network

Step 3. Open up windows of hosts.

```
> xterm h1 h2 h3 h4 h5 h6
```

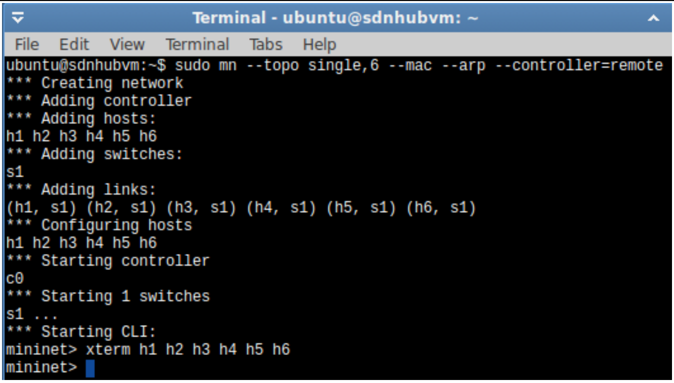


Fig. 53 – Adding hosts

Step 4. Launch two http servers.

After creating your own topology, you need to configure the servers. For the current example, in nodes 1 and 2, the HTTP Server consists of port number 80.

On h1 host:

```
> python -m SimpleHTTPServer 80
```

On h2 host:

```
> python -m SimpleHTTPServer 80
```

The servers must be configured with a unique IP address. Use the IPerf network tool to measure TCP and UDP bandwidth performance. The user can perform a series of tests that provide insight into network bandwidth availability, data loss, latency, and jitter.

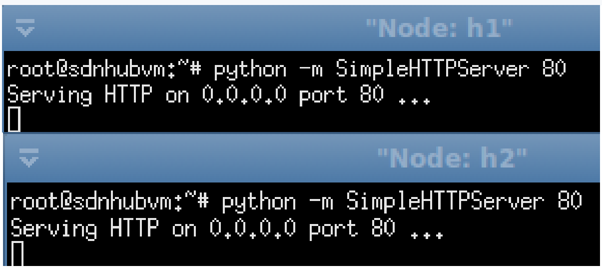


Fig. 54 – Launching the HTTP Servers for node h1 and h2

Depending on the flow of traffic to the server from client nodes, the server is scheduled. Clients access the service through one public IP address, accessible through the gateway switch.

Step 5. Send traffic to the server. To do this use the curl command:

```
> curl 10.0.1.1
```

The curl command can be used from all the four HTTPClient nodes. The Fig. 55 shows the http clients which send the traffic to the servers. This receives the web page from the server IP address. Thus, using a round robin algorithm, the client receives a server in a circle.

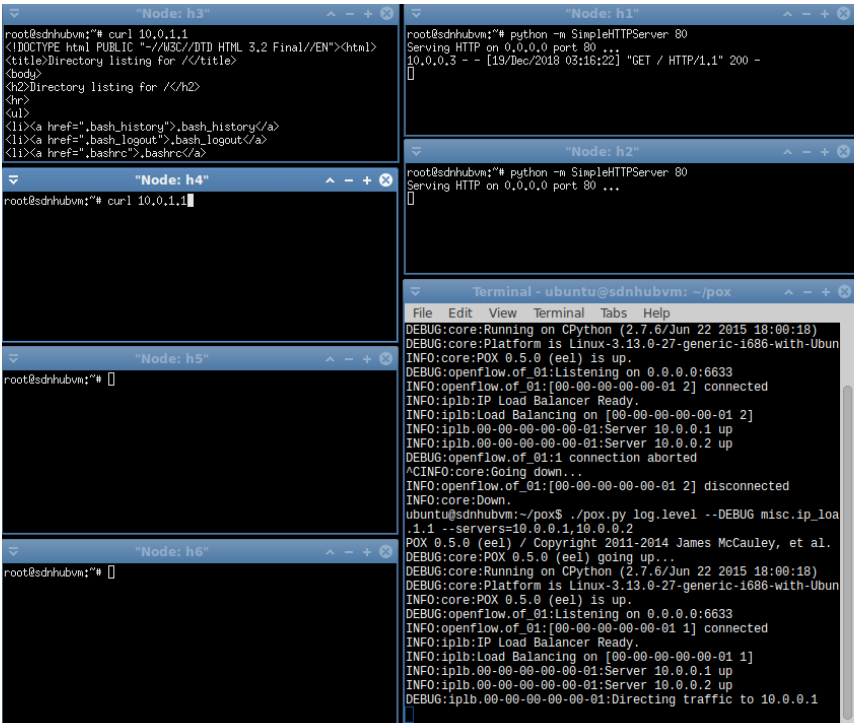


Fig. 55 – List of the http clients which send the traffic to the servers

Step 6. Send requests from all hosts to the controller using the same command.

By sending requests, we can observe how the controller balances traffic between h1 and h2 (Fig. 56). Requests are also visible on the servers themselves (h1, h2 nodes).

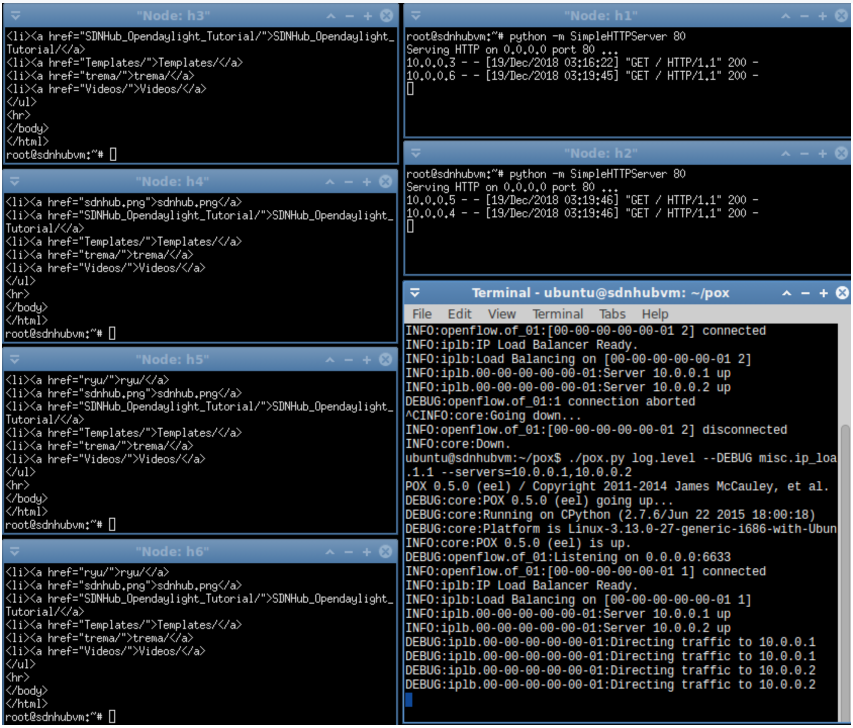


Fig. 56 – Controller window

Step 7. Observe the distribution of requests in the root of the controller.

Step 8. Make an analytical review of existing methods, algorithms and tools for load balancing in SDN. Draw up a report, answer the questions.

<u>2.3. Control questions</u>:

1. What are balancing methods used for?
2. Describe one of the methods / algorithms for balancing traffic.

3. How load balancing can improve the quality of transmissions in SDN?
4. What network load balancing algorithms do you know?
5. What are the differences between traditional routing and SDN routing?
6. How load balancing can be achieved?
7. How POX controller is used for implementing the load balancing?
8. What network tool can be used for measuring TCP and UDP bandwidth performance?
9. What are the main features of dynamic Load Balancing?

2.4. Recommended literature:

1. K. Qin and X. Liu, “Internet-of-Things monitoring system of robot welding based on software defined networking,” 2016 *First IEEE International Conference on Computer Communication and the Internet (ICCCI)*. October 2016, pp. 112-117. DOI 10.1109/cci.2016.7778889.

2. J. McCauley, “Recursive SDN for Carrier Networks,” *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 4, pp. 1-6, Oct. 2016.

3. M. Pak, “Equal-Cost Multi-Path (ECMP) Routing in Software-Defined Networking,” Csbrownedu. 2018. Available at: https://cs.brown.edu/research/pubs/theses/capstones/2015/pak.pdf. [Accessed: Feb. 2018].

4. G.N. Sentil, S. Ranjani, “Dynamic load balancing using Software Defined networks,” *International Journal of Computer Applications*, 2015, pp, 11-14. [Online]. Available: https://pdfs.semanticscholar.org/4003/55f7f9632e6c2f33024c45788ed4ae279519.pdf [Accessed: 4 Feb. 2018].

Tutorial 3
ALGORITHMS FOR FINDING THE SHORTEST PATH IN NETWORK

Goal and objectives: Analysis of the Bellman–Ford algorithm for finding the shortest path between two nodes in a network using SDN environment.

Learning objectives:

– study existing algorithms for finding the optimal path in networks;

– study the principles of operation of protocols based on algorithms.

Practical tasks:

– acquire practical skills in working with shortest path algorithms;

– conduct an analytical review of existing methods and algorithms to find a shortest path in SDN;

– draw up the report.

Setting up

In preparation for this practical training it is necessary:

– clarify the goals and mission of the research;

– study theoretical material contained in this manual, and in [1]-[4];

– familiarize oneself with the main procedures and specify the program according to defined task.

Recommended software and resources:

– Mininet emulator: http://mininet.org/;

– OracleVM VirtualBOX:
https://www.oracle.com/technetwork/server-storage/virtualbox/overview/index.html;

– SDNHub:
http://yuba.stanford.edu/~srini/tutorial/SDN_tutorial_VM_32bit.ova.

3.1. Synopsis

As it can be seen from previous works, traffic in SDN can be shaped from controller without configuring the individual switches. The administrator can build application based on their organization requirement, thus giving flexibility and efficiently managing traffic.

In this tutorial, we will implement the Bellman–Ford algorithm to find the shortest path between two nodes in a network using SDN environment. POX will be used to implement the Bellman–Ford algorithm. The Bellman–Ford algorithm will be used as an efficient approach since each node only needs to know its own number and be able to derive the number of neighbors it has. The calculation of the shortest path to a given destination will be done by hoping neighbor by neighbor from each source to destination [1].

The simulation scenario consists of eight hosts (h1, h2, h3, h4, h5, h6, h7, h8), seven switches (s1) and one POX controller in the current *mininet* environment (see Fig. 57).

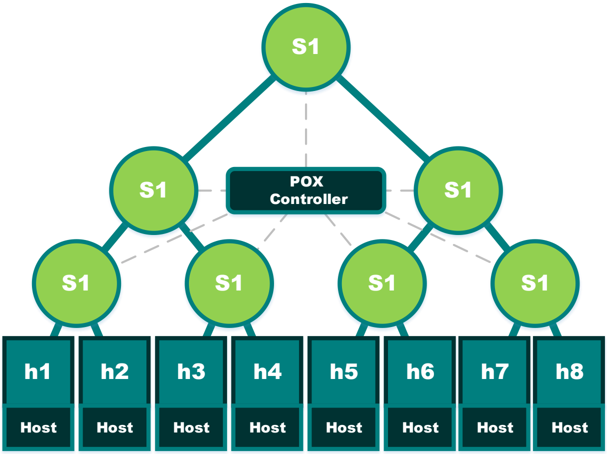


Fig. 57 – Example of network under the test

3.2. Execution order

Step 1. Prepare file with l2_bellmanford.py from the Appendix E or download it from [2] and save this file under /pox/ext folder.

Step 2. Create a network topology in *Mininet* according to your individual task or as it depicted in Fig. 57.

```
> sudo mn --topo three, 3
```

Step 3. Stop default controller:

```
> ps -aux | grep controller
> kill "your_process_ID"
```

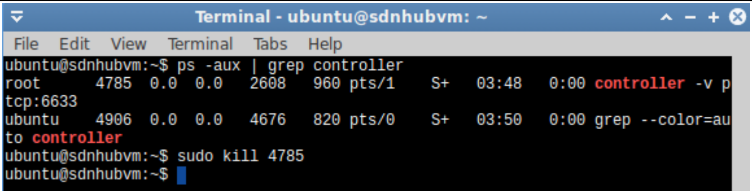


Fig. 58 – Stopping default controller

1. *Step 4*. Configure POX and run bellmanford module from POX controller. To do it, in pox folder execute the command:

```
> ./pox.py log.level -CRITICAL l2_bellmanford openflow.discovery
```

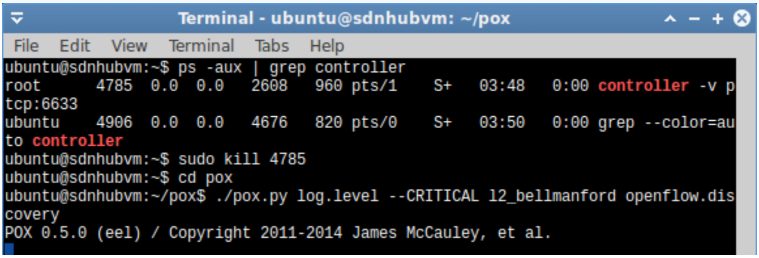


Fig. 59 – Configure POX

Step 5. Ping the h7 host from h1 host:

```
> h1 ping -c 3 h7
```

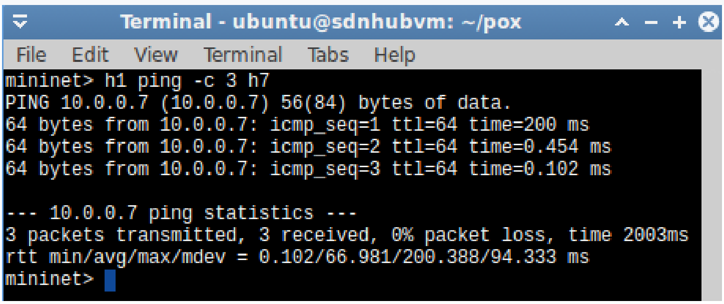


Fig. 60 – Ping test result from host h1 to host h7

We will see the new paths that are built using the Bellman-Ford algorithm for IP and ARP packets transmission.

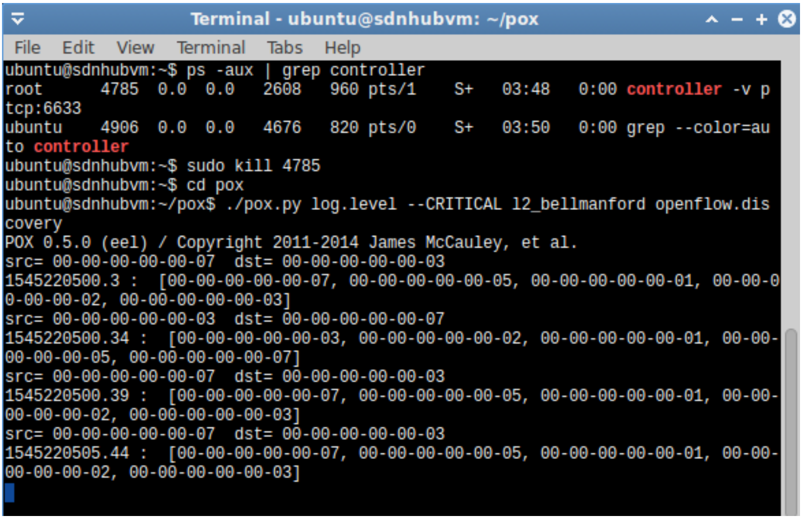


Fig. 61 – List of paths that have been built using Bellman-Ford algorithm

Step 6. Ping the h6 host from h3 host:

```
> h3 ping -c 3 h6
```

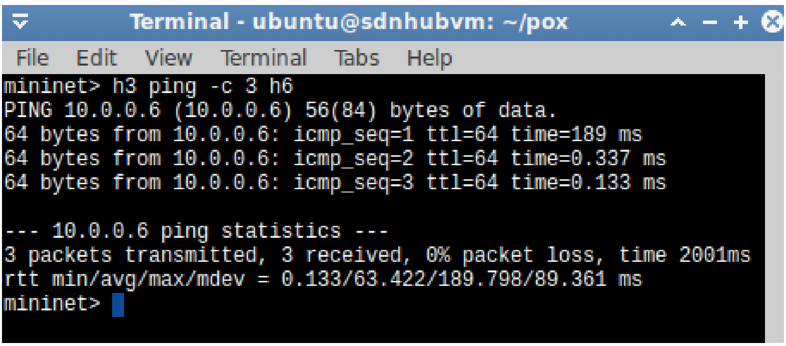


Fig. 62 – Ping h3 to h6

A new route will be created. Once all the switches and links have been detected, transmission of packets is now possible. When host h3 pings h6, Bellman-Ford algorithm finds the shortest path for the transmission as shown in Fig. 63.

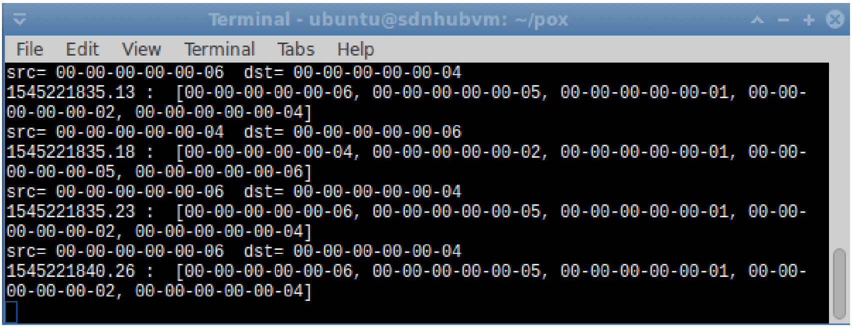


Fig. 63 – Routing along shortest path

Step 7. Observe the bandwidth and throughput of the network using a TCP stream.

Step 8. Make an analytical review of existing methods, algorithms and tools for load balancing in SDN. Draw up a report, answer the questions.

An example of individual tasks (different number of hosts):

Variant 1:

```
> sudo mn --topo tree,depth=2,fanout=5
```

Variant 2:

```
> sudo mn --topo tree,depth=2,fanout=6
```

Variant 3:

```
> sudo mn --topo tree,depth=2,fanout=7
```

3.3. Control questions:

1. What tools are used for communication between the controller and the switches?
2. How the controller coordinates with the switch using open flow protocol?
3. What algorithms can be used algorithm to find the shortest path between two nodes in a network using SDN environment?
4. What shortest path algorithms do you know?
5. What algorithm is considered as optimal? Why?
6. What network protocols use shortest path algorithms?
7. What are the differences between traditional routing and SDN routing?
8. How load balancing can be achieved?
9. How POX controller is used for implementing the load balancing?

3.4. Recommended literature:

1. A. Shivendu, D. Ghakal, D. Sharma, “Emulation of Shortest Path Algorithm in Software Defined Networking Environment,” *International Journal of Computer Applications*, 2015, vol. 116, no. 1, pp. 47-49.
2. C.-H. Ke “Using Bellman-Ford to find a shortest path (version 2),” 2018. [Online]. Available: http://csie.nqu.edu.tw/smallko/sdn/bellmanford2.htm [Accessed: 4 Feb. 2018].
3. T. Huang, “Path Computation Enhancement in SDN Networks,” Canada, 2015. [Online]. Available: http://digital.library.ryerson.ca/islandora/object/RULA%3A4465/datastream/OBJ/download/Path_computation_enhancement_in_SDN_networks.pdf [Accessed: 4 Feb. 2018].
4. S. M. Shamim, M. B. Miah, A. Sarker , A. N. Bahar, and A. Sarker, “Simulation of Minimum Path Estimation in Software Defined Networking Using Mininet Emulator,” *British Journal of Mathematics & Computer Science*, 2017, vol. 21, no. 3, pp. 1-8.

PCM2.4. Development of project for SDN-DevOps using modern CI/CD tools

Assoc. Prof., Dr. D.D. Uzun

1. Objectives and tasks

Objectives: to study the known techniques and tools used in Continuous Integration / Continuous Delivery (Deployment) (CI/CD) pipeline and apply them to provide processes of software development lifecycle (SDLC).

Learning tasks:

– to study the principles of DevOps techniques;

– to study the connection between DevOps techniques and processes of software development lifecycle;

– to select the correct tools to provide dependable functioning of software development lifecycle.

Practical tasks:

– to gain experience with installation and configuring all tools through the CI/CD pipeline;

– to develop the summary project using configured on previous step CI/CD pipeline.

Exploring tasks:

– to make grounded choice of each tools during CI/CD pipeline development.

Setting up

– to study the theoretical basics can be used materials contained in the according chapter, as well as a list of references.

Synopsis

For successfully development of this summary project, students should go through the process of CI/CD pipeline configuring and appropriate application development.

2. A brief introduction to DevOps and the CI/CD pipeline

DevOps, like agile, has evolved to encompass many different disciplines, but most people will agree on a few things: DevOps is a software development practice or a software development lifecycle (SDLC) and its central tenet is cultural change, where developers and non-developers all breathe in an environment where formerly manual things are automated; everyone does what they are best at; the number of deployments per period increases; throughput increases; and flexibility improves.

While having the right software tools is not the only thing you need to achieve a DevOps environment, some tools are necessary. A key one is continuous integration and continuous deployment (CI/CD). This pipeline is where the environments have different stages, manual things are automated, and developers can achieve high-quality code, flexibility, and numerous deployments.

This brief introduction describes an approach to creating a DevOps pipeline, using open source tools. The results are given in Table 1.

Table 1 - Open source tools for creating a DevOps pipeline

CI/CD framework	Source control management	Build automation tool	Web application server	Code testing coverage	Middleware automation tools
Jenkins Travis CI Cruise Control Buildbot Apache Gump Cabie	Git Subversion Concurrent Versions System (CVS) Vesta Mercurial	Maven Ant Gradle Bazel Make Grunt Gulp Buildr Rake A-A-P SCons BitBake Cake ASDF Cabal	Tomcat Jetty WildFly GlassFish Django Tornado Gunicorn Python Paste Rails Node.js	JUnit EasyMock Mockito PowerMock Pytest Hypothesis Tox	Ansible SaltStack Chef Puppet

Also, optional tools can be applied into the summary project development. It means, that application server can be hosted on a virtual machine (VM), on a server, or in containers, which is a frequently used solution for now. The short explanation is that a VM needs the huge footprint of an operating system, which overwhelms the application size, while a container just needs a few libraries and configurations to run the application. There are clearly still important uses for a VM, but a container is a lightweight solution for hosting an application, including an application server. Although there are other options for containers, Docker and Kubernetes are the most popular. The results of comparison between Virtual Machines and Containers are given in Table 2.

Table 2 – The results of comparison between VMs and Containers

Virtual Machines (VMs)	Containers
Represents hardware-level virtualization	Represents operating system virtualization
Heavyweight	Lightweight
Slow provisioning	Real-time provisioning and scalability
Limited performance	Native performance
Fully isolated and hence more secure	Process-level isolation and hence less secure

Server virtualization reproduces an entire computer in hardware, which then runs an entire operation system (OS). The OS runs one application. That’s more efficient than no virtualization at all, but it still duplicates unnecessary code and services for each application you want to run.

Containers take an alternative approach. They share an underlying OS kernel, only running the application and the things it depends on, like software libraries and environment variables. So, this makes containers smaller and faster to deploy.

Additional materials for development of summary project for SDN-DevOps course using modern CI/CD tools are described in Part VI (sections 20-22 and especially section 23) of the book [6].

3. Execution order

In order to successfully development of summary project for SDN-DevOps course using modern CI/CD tools the following steps are needed:

– first, create a prototype of the final project, it may be simple, because it is not the aim of this course to make a complex fully functional project;

– second, add the project source code to the Version Control System (VCS) (for example, GitHub (add, commit and push));

– third, install and configure the CI/CD framework (Jenkins, for example);

– forth, install and configure the build automation tool;

– fifth, install and setup middleware automation tools (if needed);

– sixth, justify the using of Virtual Machines vs Containers;

– seventh, install and configure of chosen web application server

– eighth, install and setup the code testing coverage tool;

– ninth, prepare the presentation for all previous steps;

– tenth, formulate outcomes.

All mentioned above steps can be presented on Fig. 1, 2.

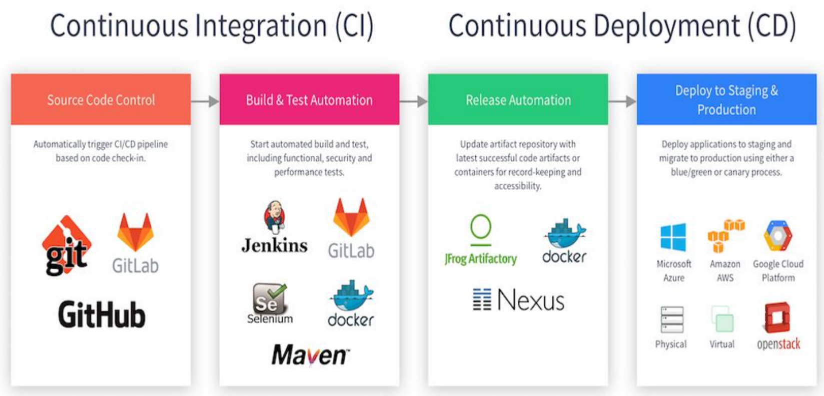


Fig. 1 – CI/CD process staging using open source tools

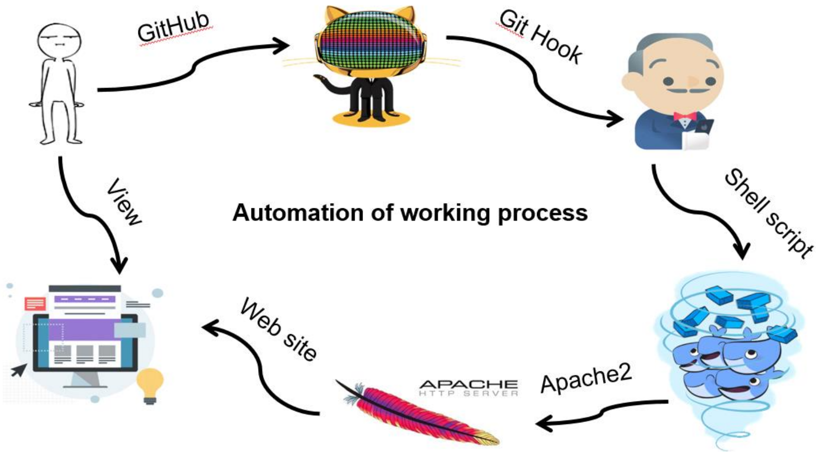


Fig. 2 – CI/CD process pipeline using open source tools

4. Requirements to the content of the report

The report should include:
– title page;
– project name, goal and tasks;
– description of chosen VCS;
– reasons for chosing CI/CD framework;
– advantages of chosen build automation tool;
– argumentation for chosing middleware automation tools;
– features of using Virtual Machines or Containers;
– advantages of chosen web application server
– description of chosen code testing coverage tool;
– presentation for all previous steps;
– developed project online working presentation;
– conclusions.

5. Testing questions

1. Define what continuous integration is?
2. Enumerate, what continuous integration tools do you know?
3. Describe the processes of continuous delivery/deployment, what is the difference between them?

4. Why the open source tools are preferred in CI/CD pipeline?
5. What version control systems do you know?
6. What CI/CD frameworks do you know?
7. What build automation tool do you know?
8. What middleware automation tools do you know?
9. What web application server do you know?
10. What code testing coverage tool do you know?

6. References

1. Installing Jenkins [Online]. Available: https://jenkins.io/doc/book/installing/.

2. Git Handbook [Online]. Available: https://guides.github.com/introduction/git-handbook/

3. Get Started with Docker, Part 1: Orientation and setup, [Online]. Available: https://docs.docker.com/get-started/

4. The Apache HTTP Server Project, [Online]. Available: https://httpd.apache.org/ABOUT_APACHE.html

5. Set Up a Jenkins Build Server, [Online]. Available: https://aws.amazon.com/getting-started/projects/setup-jenkins-build-server/

6. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 2. Modelling and Developmentment/V. S. Kharchenko (ed.) - Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. – 547 p.

Seminar

Methodology of DevOps in context of SDN and Internet of Things application

Prof., DrS V.S. Kharchenko, Assoc. Prof., Dr. D.D. Uzun, Senior Lecturer Y.O. Uzun, PhD student P.A. Hodovaniuk (KhAI)

1. Seminar objectives

The objectives areto provide knowledge and practical skills on:

– preparation of a report (analytical review or vision and brief specification of developed project - SDP) on analysis of:

a) methodology DevOps and its security related modification DevSecOps development and application;

b) implementation of methodology DevOps/DevSecOpsin context of application of software defined networks (SDN) and Internet of Things (IoT);

– preparation of a ppt presentation according with report results for short lecture/seminar for other students;

–- presentation and defence of received results.

2. Seminar preparation

Seminar preparation includes the following steps.

1) **Assignment (choice) of report subject**(analytical review, SDP) and tasks specification.

The report subject is to be agreed with the lecturer. It can be chosen by students on their own based on the following suggested list:

– principles, methodologies, methods, tools, technologies…;

– DevOps, DevSecOps:

– SW defined networks, SW defined datacenters, SW defined computing, SW defined infrastructure, SW defined everything...;

– Internet of things, Industrial IoT, IoT architectures, data transfer, Internet routing…;

– cloud computing, fog computing, edge computing…;

– industrial systems, enterprise, manufacturing…;

– human-machine interfaces, user interfaces, human factors...;

– cyber security, safety, dependability, maintainability…;

– AI, machine learning, neural nets…;

– web services, SOA, monolith services, micro services, serverless technology….

Suggested report subjects (can be extended):

– Analysis of concepts, principles and technologies DevOps;

– DevOps as a stage of system IT engineeringevolution;

– Comparative analysis of DevOps and DevSecOps;

– DevSecOps integration into software development lifecycle;

– DevOps and SDX technologies: crossing for synergy;

– DevOps and IoT technologies: crossing for synergy;

– DevOps and cloud technologies: crossing for synergy;

– Analysis of SDX technologies (X = {networking, computing, infrastructuring});

– Metrics and assessment techniques for DevOps application efficiency;

– Metrics and assessment techniques for DevSecOps application efficiency;

– Life cycle of Dev(Sec)Ops related project: features and processes;

– Analysis of tools for Dev(Sec)Ops;

– DevOps in top technology trends (Gartner based analysis);

– Analysis of standards for Dev(Sec)Ops;

– AI, machine learning, neural nets…: How can modern technologies be used for Dev(Sec)Ops implementation and enhancing?

Report subject is to be agreed with the lecturer and consist with the subject area of the course (IoT and modern technologies for Industry 4.0, 5.0).

2) **Work plan development and responsibility assignment among target group members.** Work plan can be presented as a Gantt chart that includes the main events, time-frames and assignment of responsibility among the target group members.

The target group consists of 2-3 persons.

Time resource is 8x(2/3) = 16/24 hours (+ 20 minutes for the presentation and defence). The responsibility assignment is determined by the group members.

Suggested responsibility assignment:

– manager responsible for planning and coordination of activities and presents the idea on the seminar (1st part of the overall report - task statement),

– analyst or system developer (2nd part of the report),

– application developer (3rd part of the report + style concept).

3) **Search of the information about report subject** (library, the Internet, resources from department) and primary analysis. The search of the information is conducted using the keywords given in paragraph 2 (1). Methodological guidelines and the selected readings are given individually (per groups).

Please use reference list [1-21]. Theoretical issues for DevOps and SDN/IoT interaction particularities are described in Part VI (SDN –

section 20-22, DevOps and SDN/IoT – section23) of the book [1]. Additional references can be searched in Internet according with keywords and recommendations of lecturer.

4) **Report and presentation plans development**. Report plan includes:

- introduction (relevance, reality challenges, brief analysis of the problem - references, purpose and tasks of the report, structure and contents characteristics);
- systematized description of the main report parts (classification schemes, models, methods, tools, technologies, selection of indexes and criteria for assessment, comparative studies);
- conclusions (established goal achievement, main theoretical and practical results, result validity, ways of further work on the problem);
- list of references;
- appendixes.

5) **Report writing.** The report should stand for 15-20 A4 pages (font size 14, spacing 1.5., margins 2 cm) including the title page, contents, main text, list of references, appendixes. Unstructured reports or reports compiled directly from Internet sources (more 50%), having incorrect terms and no conclusion shall not be considered.

The work plan and responsibility assignment (Gantt chart), presentation slides and an electronic version of all material related to the work are required to be included in appendixes.

6) **Presentation preparation.** The presentation is to be designed in PowerPoint and be consistent with the report plan (10-15 slides); the time-frame for the presentation is 15 minutes.

The presentation should include the slides as follows:

– title slide (specification of the educational institution, department, course of study, report subject, authors, date);
– contents (structure) of the report;
– relevance of the issues covered, the purpose and the tasks of the report based on the relevance analysis;
– slides with the details of the tasks;
– report conclusion;
– list of references;
– testing questions.

Each slide should include headers with the report subject and authors.

The contents of the slides should include the keywords, figures, formulas rather than the parts from the report.

The information can be presented dynamically.

3. Presentation

The presentation should be given at the seminar during 20 minutes including presentation (10-15 minutes) and discussion (5-10 minutes).

Time schedule can be specified by lecturer.

4. Report assessment

The work is assessed on the following parameters:

a) report text quality (form and contents),

b) presentation quality (contents and style),

c) report quality (contents, logical composition, timing shared among parts, conclusion),

d) fullness and correctness of the answers.

Each student is given an individual mark for the report and the presentation based on the results and responsibility assignment.

5. References

1. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 2. Modelling and Development /V. S. Kharchenko (ed.) - Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. –547 p.

2. Top 10 strategical technology trends in 2019 have been presented by Gartner company [https://www.gartner.com/ en/doc/383829-top-10-strategic-technology-trends-for-2019-a-gartner-trend-insight-report]

3. Introduction to DevOps on AWS, https://d0.awsstatic.com/ whitepapers/AWS_DevOps.pdf

4. A beginner's guide to building DevOps pipelines with open source tools, https://opensource.com/article/19/4/devops-pipeline

5. DevOps - Technology and Tools overview, https://www.gecko.rs/sites/default/files/pdf/Gecko_Solutions_DevOps_Technology_Overview.pdf

6. Practicing Continuous Integration and Continuous Delivery on AWS, https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf

7. The DevOps Handbook: An Introduction Summary, https://caylent.com/devops-handbook-introduction-summary/

8. The Definitive Guide to Scrum: The Rules of the Game, https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

9. DevOps in the Internet of Things. Six reasons it matters and how to get there, https://events.windriver.com/wrcd01/wrcm/2016/08/WP-devops-in-the-internet-of-things.pdf

10. DevOps for IoT Applications using Cellular Networks and Cloud, Athanasios Karapantelakis, Hongxin Liang, Keven Wang, Konstantinos Vandikas, Rafia Inam, Elena Fersman, Ignacio Mulas-Viela, Nicolas Seyvet, Vasileios Giannokostas, https://www.ericsson.com/assets/local/publications/conference-papers/devops.pdf

11. Vlasov, Y., Illiashenko, O., Uzun, D., Haimanov, O. Prototyping tools for IoT systems based on virtualization techniques(Conference Paper). Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018, 9 July 2018, P. 87-92

12. M. H. Syed, E. B. Fernández. Cloud Ecosystems Support for Internet of Things and DevOps Using Patterns, Conference: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI),DOI: 10.1109/IoTDI.2015.31

13. AWS IoT Plant Watering Sample, https://docs.aws.amazon.com/iot/latest/developerguide/iot-plant-watering.html

14. Cloud Tutorial: AWS IoT, https://www.cse.wustl.edu/~lu/cse521s/Slides/aws-iot.pdf

15. Network Transformation with NFV and SDN, https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000628-en.pdf

16. Operationalizing SDN and NFV Networks, https://www2.deloitte.com/content/dam/Deloitte/us/Documents/technology-media-telecommunications/us-tmt-operations-sdn-and-nfv-networks.pdf

17. Allan K. (2018), “Automated Security Testing Best Practices” https://phoenixnap.com/blog/devsecopsbest-practices-automated-security-testing

18. Litz S. (2015) “What is DevSecOps” http://www.devsecops.org/ blog/2015/2/15/what-is-devsecops

19. Savant S(2018) “What is the difference between DevOps and DevSecOps”, https://www.quora.com/What-is-the-difference-between-DevOps-andDevSecOps

20. GitLab, (2018) “Static Application Security Testing (SAST)”, https://docs.gitlab.com/ee/user/project/merge_requests/sast.html

21. S. Harris, “Physical and Environmental Security. In CISSP Exam Guide”, USA McGraw-Hill, 6th ed., pp.427-502 2013.

APPENDIX A. TEACHING PROGRAM OF THE COURSE PCM 2 "SOFTWARE DEFINED NETWORKS AND IOT"

DESCRIPTION OF THE MODULE

TITLE OF THE MODULE	Code
Software defined networks basics	**PCM 2.1**

Teacher(s)	Department
Coordinating: Assoc. Prof., Dr. R.K. Kudermetov **Others: Modules PCM2.1, PCM2.2**: Assoc. Prof., Dr. V.V. Shkarupylo, Assoc. Prof., Dr. R.K. Kudermetov, MSc student D.S. Mazur. **Module PCM2.3**: DrS. I.S. Skarga-Bandurova, PhD student A.Yu. Velykzhanin, Assoc. Prof. Dr. L.O. Shumova. **Module PCM2.4**: Prof., DrS V.S. Kharchenko, Assoc. Prof., Dr. D.D. Uzun, Senior Lecturer Y.O. Uzun, PhD student P.A. Hodovaniuk	Computer systems and networks (ZNTU), Computer Engineering (EUNU), Computer Systems, Networks and Cybersecurity Department (KhAI)

Study cycle	Level of the module	Type of the module
PhD	A	Full-time tuition

Form of delivery	Duration	Language(s)
full-time tuition, distance tuition	Four weeks	English

Prerequisites	
Prerequisites: Computer Systems and Networks, Software Engineering, Modern Programming Methods, Modeling Basics	**Co-requisites (if necessary):**

Credits of the module	Total student workload	Contact hours	Individual work hours
4	120	56	64

Aim of the module (course unit): competences foreseen by the study program
The aim of the course is to give PhD students a deep understanding of Software

Defined Networking (SDN) – the important emerging network technology – to teach to select, evaluate and implement SDN controllers for various platforms and applications, theoretical and practical skills in the field of research, design and modelling safety systems based on SDN with emphasize on IoT. During the training, graduate students can study and analyze approaches to managing the IoT with SDN, smart routing and scheduling, traffic management and optimization in IoT.		
Learning outcomes of module (course unit)	**Teaching/learning methods**	**Assessment methods**
At the end of module the successful student will be able to: 1. Explain and discuss the basic concepts and architecture of SDN, concepts of managing the IoT applications with SDN.	Lectures, Learning in laboratories	Module Evaluation Questionnaire
2. Compare and contrast common networking approaches and SDN.	Interactive lectures, Learning in laboratories	Module Evaluation Questionnaire
3. Describe the SDN data plane. Explain the operation of SDN control plane.	Interactive lectures, Learning in laboratories	Module Evaluation Questionnaire
4. Explain network virtualization.	Interactive lectures, Learning in laboratories	Module Evaluation Questionnaire
5. Compare and contrast OpenFlow releases.	Interactive lectures, Learning in laboratories	Module Evaluation Questionnaire
6. Formulate requirements for configuration SDN. Create and analyze the configuration of SDN.	Learning in laboratories	Module Evaluation Questionnaire
7. Employ obtained theoretical knowledge for the purpose of SDN simulation and deployment.	Learning in laboratories	Module Evaluation Questionnaire
8. Formulate main approaches, techniques and tools for smart routing and scheduling SDN to IoT	Learning in laboratories	Module Evaluation Questionnaire
9. Formulate traffic management tasks, traffic parameters, traffic types and related data services,	Learning in laboratories	Module Evaluation Questionnaire

traffic management mechanisms		
10. Use SDN-related languages and programming approaches in practice.	Learning in laboratories	Module Evaluation Questionnaire
11. Explain the operation of the SDN for support IoT scalability, agility and flexibility. Conduct SDN composing, configuring and scaling by way of simulation.	Learning in laboratories	Module Evaluation Questionnaire
12. Design the architecture of software-defined network with respect to given requirements.	Learning in laboratories	Module Evaluation Questionnaire
13. Implement the design solutions, obtained by way of simulation, in practice.	Learning in laboratories	Module Evaluation Questionnaire
14. Perform administration of the switches, management and analysis of the results of traffic monitoring.	Learning in laboratories	Module Evaluation Questionnaire

Themes	**Contact work hours**						**Time and tasks for individual work**
	Lectures	Seminars	Practiacl work	Laboratory work	**Total contact work**	**Individual work**	**Tasks**
1. Understanding SDN. Historical Background and Key Concepts 1.1. The Evolution of Switches and Control Planes 1.2. The Evolution of Networking Technology 1.3. Predecessors of SDN 1.4. Network Virtualization	2				**2**	**4**	1.5. Survey of Computer Networks Historical Evolution 1.6. Analysis of the Research Papers about SDN.
2. SDN Architecture and its Components. Devices, Controller, Applications	2				**2**	**4**	2.6. Perform a Comparative Analysis of Existing

2.1. Fundamental Characteristics of SDN. Plane Separation 2.2. SDN Operation 2.3. SDN Devices 2.4. SDN Controller. Existing SDN Controller Implementations 2.5. SDN Applications							SDN Controller Implemen-tations 2.7. Create Existing SDN Device Implemen-tations 2.8. Classify the Functions of SDN Application 2.9. Compare SDN with Alternative Technologies
3. OpenFlow Protocol. The Basics, Peculiarities and Limitations 3.1. OpenFlow Overview 3.2. The OpenFlow Switch and Controller 3.3. The OpenFlow Protocol 3.4. OpenFlow Releases 1.0, 1.2, 1.3 Survey 3.5. OpenFlow Limitations	2				**2**	**4**	3.6. Overview of Open Networking Foundation Activities 3.7. Create a Chronological Report on the Development (Innovations) of OpenFlow Switch Specifications
4. Mininet installation and configuring. Building simple networking applications				2	**2**	**4**	4.1. Create the Report on the Topic and answer the Questions
5. Exploring OpenDaylight, installation and configuring. SDN Emulation with Mininet and OpenDaylight				2	**2**	**4**	5.1. Create the Report on the Topic and answer the Questions
6. SDN Simulation. The Basics, Toolboxes and Main Concepts. 6.1. Considering SDN as a System. Key Components: Controllers, Switches, Hosts. 6.2. Simulating SDN Infrastructure. Network Configuring and Scaling. 6.3 Network Orchestration and Virtualization. The	2				**2**	**4**	6.4 An Overview on SDN Simulation Toolboxes. 6.5 Measurement and Assessment of QoS-related SDN-metrics.

Simulation of Dataflows.							
7. SDN Testing. OpenFlow Protocol and Network Validation. 7.1. Setting up the Configuration of SDN in Mininet Testing Environment. 7.2. Testing the Soundness and Consistency of SDN Infrastructure. 7.3. Dataflows Orchestration. SDN Reconfiguration and Scaling.	2			2	**4**	**6**	7.4. The API for SDN Programming and OpenFlow Protocol.
8. Software-defined Networks Programing and Python Scripting. 8.1. An in-depth Look at SDN-related Programming Approaches, Principles and Concepts. 8.2. Setting up SDN Configuration by Way of Python Scripting. 8.3. Sophisticating the Python Scripting. Bringing in the Automation.	2			4	**6**	**6**	8.4 The API for SDN Programming and OpenFlow Protocol.
9. Managing the IoT with SDN SLA management. 9.1. Metrics. Smart routing and scheduling in SDN. 9.2. Data streaming over SDN.	2		2	4	**8**	**6**	9.3. Metrics for evaluation performance of QoS routing algorithms. QoS routing algorithms applicable to large-scale SDN.
10. Optimization of SDN Traffic Flow for IoT. 10.1. Traffic scheduling algorithms.	2		2	4	**8**	**6**	10.2.Algo-rithms for calculating the optimal position of the SDN-controller. 10.3. Balancing algorithms in IoT-

							based software defined networks. 7.4. Algo-rithms for finding the optimal path in SDN networks.
11. SDN Performance prediction. 11.1. Algorithms performance metrics. 11.2. An overall approach to detect and diagnose failures in SDN.	2		2	4	**8**	**6**	11.3. Case study
12. Development of project for SDN-DevOps using modern CI/CD tools. 12.1. Principles of DevOps techniques. 12.2. To study the connection between DevOps techniques and processes of software development lifecycle.	2		4		**6**	**6**	12.3. To select the correct tools to provide dependable functioning of software development lifecycle. 12.4. To gain experience with installation and configuring all tools through the CI/CD pipeline. 12.5. To develop the summary project using configured on previous step CI/CD pipeline. 12.6. To make grounded choice of each tools during CI/CD pipeline development.
13. Methodology of DevOps in context of SDN and IoT.		4			**4**	**4**	13.1. Preparation of a report (analytical review or vision and brief specification of

							developed project - SDP) on analysis of: a) methodo-logy DevOps and its security related modification DevSecOps development and application; b) implemen-tation of methodology DevOps/DevSecOpsin context of application of software defined networks (SDN) and Internet of Things (IoT).
Total	**20**	**4**	**10**	**22**	**56**	**64**	

Assessment strategy	**Weight in %**	**Deadlines**	**Assessment criteria**
Lecture activity, including fulfilling special self-tasks	10	2, 4	85% – 100% An outstanding piece of work, superbly organized and presented, excellent achievement of the objectives, evidence of original thought. 70% – 84% Students will show a thorough understanding and appreciation of the material, producing work without significant error or omission. Objectives achieved well. Excellent organization and presentation. 60% – 69% Students will show a clear understanding of the issues involved and the work should be well written and well organized. Good work towards the objectives. The exercise should show evidence that

			the student has thought about the topic and has not simply reproduced standard solutions or arguments. 50% – 59% The work should show evidence that the student has a reasonable understanding of the basic material. There may be some signs of weakness, but overall the grasp of the topic should be sound. The presentation and organization should be reasonably clear, and the objectives should at least be partially achieved. 45% – 49% Students will show some appreciation of the issues involved. The exercise will indicate a basic understanding of the topic, but will not have gone beyond this, and there may well be signs of confusion about more complex material. There should be fair work towards the laboratory work objectives. 40% – 44% There should be some work towards the laboratory work objectives, but significant issues are likely to be neglected, and there will be little or no appreciation of the complexity of the problem. 20% – 39% The work may contain some correct and relevant material, but most issues are neglected or are covered incorrectly. There should be some signs of appreciation of the laboratory work requirements. 0% – 19% Very little or nothing that is correct and relevant and no real appreciation of the laboratory work requirements.
Learning in practicums	30	3,4	85% – 100% An outstanding piece of work, superbly organized and presented, excellent achievement of the objectives, evidence of original thought. 70% – 84% Students will show a thorough understanding and appreciation

			of the material, producing work without significant error or omission. Objectives achieved well. Excellent organization and presentation. 60% – 69% Students will show a clear understanding of the issues involved and the work should be well written and well organized. Good work towards the objectives. The exercise should show evidence that the student has thought about the topic and has not simply reproduced standard solutions or arguments. 50% – 59% The work should show evidence that the student has a reasonable understanding of the basic material. There may be some signs of weakness, but overall the grasp of the topic should be sound. The presentation and organization should be reasonably clear, and the objectives should at least be partially achieved. 45% – 49% Students will show some appreciation of the issues involved. The exercise will indicate a basic understanding of the topic, but will not have gone beyond this, and there may well be signs of confusion about more complex material. There should be fair work towards the laboratory work objectives. 40% – 44% There should be some work towards the laboratory work objectives, but significant issues are likely to be neglected, and there will be little or no appreciation of the complexity of the problem. 20% – 39% The work may contain some correct and relevant material, but most issues are neglected or are covered incorrectly. There should be some signs of appreciation of the laboratory work requirements.

			0% – 19% Very little or nothing that is correct and relevant and no real appreciation of the laboratory work requirements.
Module Evaluation Quest	60	4	The score corresponds to the percentage of correct answers to the test questions

Author	**Year of issue**	**Title**	**No of periodical or volume**	**Place of printing. Printing house or Internet link**
Compulsory literature				
P. Goransson, C. Black, T. Culver	2016	Software Defined Networks: A Comprehensive Approach - 2nd Edition		Morgan Kaufmann
T.D. Nadeau, K. Gray	2013	SDN: Software Defined Networks		O'Reilly Media, Inc.
F. Hu	2014	Network Innovation through OpenFlow and SDN: Principles and Design		6000 Broken Sound Parkway NW, Suite 300, Boca Raton, Florida, USA. CRC Press.
Additional literature				
N. Feamster, J. Rexford and E. Zegura.	2014	The Road to SDN: An Intellectual History of Programmable Networks	Vol. 44(2)	ACM SIGCOMM Computer Communication Review
B. Underdahl, G. Kinghorn	2015	Software Defined Networking For Dummies		John Wiley & Sons, Inc.
M. Casado, M.J. Freedman, J. Pettit, at al	2007	Ethane: Taking control of the enterprise.	Vol. 37(4)	ACM SIGCOMM Computer

				Communication Review
B. Nunes, M. Mendonca, X. N. Nguyen, at al.	2014	A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks	Vol. 16(3)	IEEE Communications Surveys & Tutorials
M. Casado, N. Foster and A. Guha.	2014	Abstractions for Software-Defined Networks	Vol. 57(10)	Communications of the ACM
B. Pfaff, J. Pettit, K. Amidon, at al.	2009	Extending Networking into the Virtualization Layer.		In *Proc. Hotnets*
P. Rekha and M. Dakshayini	2015	A Study of Software Defined Networking with OpenFlow	Vol. 122(5)	International Journal of Computer Applications
W. Braun and M. Menth.	2014	Software-Defined Networking Using Openflow: Protocols, Applications and Architectural Design Choices.	Vol. 6(2)	Future Internet
	2009 -2016	OpenFlow Switch Specification Ver 1.x.x		https://www.opennetworking.org/technical-communities/areas/specification
K. Kaur, J. Singh, and N. S. Ghumman.	2014	Mininet as Software Defined Networking testing platform		In International Conference on Communication, Computing & Systems (ICCCS,

				2014).
F. Keti and Sh. Askar	2015	Emulation of Software DEFINED networks Using Mininet in Different Simulation Environments		In Intelligent Systems, Modeling and Simulation (ISMS), 2015 6th International Conference
J. Medved, R. Varga, A. Tkacik, and K. Gray.	2014	OpenDaylight: Towards a Model-Driven SDN Controller Architecture.		A World of Wireless, Mobile and Multimedia Networks. IEEE 15th International Symposium, 2014.
T. Bakhshi	2017	State of the Art and Recent Research Advances in Software Defined Networking	Vol. 2017	Wireless Communications and Mobile Computing. doi: 10.1155/2017/7191647

Appendix B. Program code for Laboratory Work 3

```
from mininet.topo import Topo

class MyTopo( Topo ):
  "Simple topology example."
  def __init__( self ):
        "Create custom topo."
        # Initialize topology
        Topo.__init__( self )
        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2 = self.addHost( 'h2' )
        h3 = self.addHost( 'h3' )
        h4 = self.addHost( 'h4' )
        s1 = self.addSwitch( 's1' )
        s2 = self.addSwitch( 's2' )
        # Add links
        self.addLink( h1, s1 )
        self.addLink( h2, s1 )
        self.addLink( h3, s1 )
        self.addLink( s1, s2 )
        self.addLink( s2, h4 )

topos = { 'mytopo': ( lambda: MyTopo() ) }

from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr

log = core.getLogger()
s1_dpid=0
s2_dpid=0

def _handle_ConnectionUp (event):
  global s1_dpid, s2_dpid
  print "ConnectionUp: ",
  dpidToStr(event.connection.dpid)

  #remember the connection dpid for switch
  for m in event.connection.features.ports:
        if m.name == "s1-eth1":
              s1_dpid = event.connection.dpid
              print "s1_dpid=", s1_dpid
        elif m.name == "s2-eth1":
              s2_dpid = event.connection.dpid
              print "s2_dpid=", s2_dpid
```

```
    def _handle_PacketIn (event):
      global s1_dpid, s2_dpid
      #          print          "PacketIn:          ",
dpidToStr(event.connection.dpid)
      if event.connection.dpid==s1_dpid:
            msg = of.ofp_flow_mod()
            msg.priority =1
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0806
            msg.actions.append(of.ofp_action_output(port
= of.OFPP_ALL))
            event.connection.send(msg)
            msg = of.ofp_flow_mod()
            msg.priority =100
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0800
            msg.match.nw_src = "10.0.0.1"
            msg.match.nw_dst = "10.0.0.4"
            msg.actions.append(of.ofp_action_enqueue(port
= 4, queue_id=1))
            event.connection.send(msg)
            msg = of.ofp_flow_mod()
            msg.priority =100
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0800
            msg.match.nw_src = "10.0.0.2"
            msg.match.nw_dst = "10.0.0.4"
            msg.actions.append(of.ofp_action_enqueue(port
= 4, queue_id=2))
            event.connection.send(msg)
            msg = of.ofp_flow_mod()
            msg.priority =10
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0800
            msg.match.nw_dst = "10.0.0.1"
            msg.actions.append(of.ofp_action_output(port
= 1))
            event.connection.send(msg)
            msg = of.ofp_flow_mod()
            msg.priority =10
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0800
            msg.match.nw_dst = "10.0.0.2"
```

```
            msg.actions.append(of.ofp_action_output(port
= 2))
            event.connection.send(msg)
            msg = of.ofp_flow_mod()
            msg.priority =10
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0800
            msg.match.nw_dst = "10.0.0.3"
            msg.actions.append(of.ofp_action_output(port
= 3))
            event.connection.send(msg)
            msg = of.ofp_flow_mod()
            msg.priority =10
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.dl_type = 0x0800
            msg.match.nw_dst = "10.0.0.4"
            msg.actions.append(of.ofp_action_output(port
= 4))
            event.connection.send(msg)
      elif event.connection.dpid==s2_dpid:
            msg = of.ofp_flow_mod()
            msg.priority =1
            msg.idle_timeout = 0
            msg.hard_timeout = 0
            msg.match.in_port =1
            msg.actions.append(of.ofp_action_output(port
= 2))
            event.connection.send(msg)
            msg= of.ofp_flow_mod()
            msg.priority=1
            msg.idle_timeout= 0
            msg.hard_timeout= 0
            msg.match.in_port=2
            msg.actions.append(of.ofp_action_output(port
= 1))
            event.connection.send(msg)

    def launch ():

      core.openflow.addListenerByName("ConnectionUp",
_handle_ConnectionUp)
            core.openflow.addListenerByName("PacketIn",
_handle_PacketIn)
```

Appendix C. Program code for Tutorial 3

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.revent import *
from pox.lib.recoco import Timer
from collections import defaultdict
from pox.openflow.discovery import Discovery
from pox.lib.util import dpid_to_str
import time

log = core.getLogger()

# Adjacency map.  [sw1][sw2] -> port from sw1 to sw2
adjacency                                                   =
defaultdict(lambda:defaultdict(lambda:None))

# Switches we know of.  [dpid] -> Switch
switches = {}

# ethaddr -> (switch, port)
mac_map = {}

# Waiting path.  (dpid,xid)->WaitingPath
waiting_paths = {}

# Time to not flood in seconds
FLOOD_HOLDDOWN = 5

# Flow timeouts
FLOW_IDLE_TIMEOUT = 10

FLOW_HARD_TIMEOUT = 30

# How long is allowable to set up a path?
PATH_SETUP_TIME = 4

def _get_raw_path (src,dst):
  distance = {}
  previous = {}
  sws = switches.values()

  for dpid in sws:
    distance[dpid] = 9999
    previous[dpid] = None

  distance[src]=0
```

```
    for m in range(len(sws)-1):
      for p in sws:
        for q in sws:
          if adjacency[p][q]!=None:
             w = 1
             if distance[p] + w < distance[q]:
               distance[q] = distance[p] + w
               previous[q] = p
    r=[]
    p=dst
    r.append(p)
    q=previous[p]

    while q is not None:
      if q == src:
        r.append(q)
        break

      p=q
      r.append(p)
      q=previous[p]

    r.reverse()
    return r

  def _check_path (p):
    for a,b in zip(p[:-1],p[1:]):
      if adjacency[a[0]][b[0]] != a[2]:
        return False
      if adjacency[b[0]][a[0]] != b[1]:
        return False
    return True

  def _get_path (src, dst, first_port, final_port):
    if src == dst:
      path = [src]
    else:
      path = _get_raw_path(src, dst)
      if path is None: return None
      print "src=",src," dst=",dst
      print time.time(),": ",path

    r = []
    in_port = first_port

    for s1,s2 in zip(path[:-1],path[1:]):
      out_port = adjacency[s1][s2]
      r.append((s1,in_port,out_port))
```

```
        in_port = adjacency[s2][s1]

      r.append((dst,in_port,final_port))
      assert _check_path(r), "Illegal path!"

      return r

    class WaitingPath (object):
      def __init__ (self, path, packet):
        self.expires_at = time.time() + PATH_SETUP_TIME
        self.path = path
        self.first_switch = path[0][0].dpid
        self.xids = set()
        self.packet = packet

        if len(waiting_paths) > 1000:
          WaitingPath.expire_waiting_paths()

      def add_xid (self, dpid, xid):
        self.xids.add((dpid,xid))
        waiting_paths[(dpid,xid)] = self

      @property
      def is_expired (self):
        return time.time() >= self.expires_at

      def notify (self, event):
        self.xids.discard((event.dpid,event.xid))
        if len(self.xids) == 0:
          if self.packet:
            log.debug("Sending delayed packet out %s"
                          %
(dpid_to_str(self.first_switch),))
            msg = of.ofp_packet_out(data=self.packet,

action=of.ofp_action_output(port=of.OFPP_TABLE))
            core.openflow.sendToDPID(self.first_switch,
msg)

core.l2_multi.raiseEvent(PathInstalled(self.path))

      @staticmethod
      def expire_waiting_paths ():
        packets = set(waiting_paths.values())
        killed = 0
        for p in packets:
          if p.is_expired:
            killed += 1
```

```
            for entry in p.xids:
              waiting_paths.pop(entry, None)
        if killed:
          log.error("%i   paths   failed   to   install"   %
(killed,))

    class PathInstalled (Event):
      def __init__ (self, path):
        Event.__init__(self)
        self.path = path

    class Switch (EventMixin):
      def __init__ (self):
        self.connection = None
        self.ports = None
        self.dpid = None
        self._listeners = None
        self._connected_at = None

      def __repr__ (self):
        return dpid_to_str(self.dpid)

      def  _install  (self,  switch,  in_port,  out_port,
match, buf = None):
        msg = of.ofp_flow_mod()
        msg.match = match
        msg.match.in_port = in_port
        msg.idle_timeout = FLOW_IDLE_TIMEOUT
        msg.hard_timeout = FLOW_HARD_TIMEOUT
        msg.actions.append(of.ofp_action_output(port     =
out_port))
        msg.buffer_id = buf
        switch.connection.send(msg)

      def _install_path (self, p, match, packet_in=None):
        wp = WaitingPath(p, packet_in)
        for sw,in_port,out_port in p:
          self._install(sw, in_port, out_port, match)
          msg = of.ofp_barrier_request()
          sw.connection.send(msg)
          wp.add_xid(sw.dpid,msg.xid)

      def  install_path  (self,  dst_sw,  last_port,  match,
event):
        p    =    _get_path(self,    dst_sw,    event.port,
last_port)
        if p is None:
          log.warning("Can't   get   from   %s   to   %s",
```

```
match.dl_src, match.dl_dst)

          import pox.lib.packet as pkt
          if (match.dl_type == pkt.ethernet.IP_TYPE and
              event.parsed.find('ipv4')):
            log.debug("Dest unreachable (%s -> %s)",
                      match.dl_src, match.dl_dst)

            from pox.lib.addresses import EthAddr
            e = pkt.ethernet()
            e.src = EthAddr(dpid_to_str(self.dpid))
            e.dst = match.dl_src
            e.type = e.IP_TYPE
            ipp = pkt.ipv4()
            ipp.protocol = ipp.ICMP_PROTOCOL
            ipp.srcip = match.nw_dst
            ipp.dstip = match.nw_src
            icmp = pkt.icmp()
            icmp.type = pkt.ICMP.TYPE_DEST_UNREACH
            icmp.code = pkt.ICMP.CODE_UNREACH_HOST
            orig_ip = event.parsed.find('ipv4')

            d = orig_ip.pack()
            d = d[:orig_ip.hl * 4 + 8]

            import struct
            d = struct.pack("!HH", 0,0) + d #FIXME: MTU

            icmp.payload = d
            ipp.payload = icmp
            e.payload = ipp

            msg = of.ofp_packet_out()
            msg.actions.append(of.ofp_action_output(port
= event.port))

            msg.data = e.pack()
            self.connection.send(msg)

          return

        log.debug("Installing path for %s -> %s %04x (%i
hops)",
            match.dl_src,   match.dl_dst,   match.dl_type,
len(p))

        self._install_path(p, match, event.ofp)
        p       =       [(sw,out_port,in_port)       for
```

```
sw,in_port,out_port in p]
        self._install_path(p, match.flip())

      def _handle_PacketIn (self, event):
        def flood ():
          if self.is_holding_down:
            log.warning("Not    flooding    --    holddown
active")
          msg = of.ofp_packet_out()
          msg.actions.append(of.ofp_action_output(port   =
of.OFPP_FLOOD))
          msg.buffer_id = event.ofp.buffer_id
          msg.in_port = event.port
          self.connection.send(msg)

        def drop ():
          if event.ofp.buffer_id is not None:
            msg = of.ofp_packet_out()
            msg.buffer_id = event.ofp.buffer_id
            event.ofp.buffer_id = None # Mark is dead
            msg.in_port = event.port
            self.connection.send(msg)

        packet = event.parsed
        loc  =  (self,  event.port)  #  Place  we  saw  this
ethaddr
        oldloc = mac_map.get(packet.src)  # Place we last
saw this ethaddr

        if          packet.effective_ethertype           ==
packet.LLDP_TYPE:
          drop()
          return

        if oldloc is None:
          if packet.src.is_multicast == False:
            mac_map[packet.src]  =  loc  #  Learn  position
for ethaddr
            log.debug("Learned %s at %s.%i", packet.src,
loc[0], loc[1])
        elif oldloc != loc:

          if
core.openflow_discovery.is_edge_port(loc[0].dpid,
loc[1]):
            log.debug("%s  moved  from  %s.%i  to  %s.%i?",
packet.src,
                      dpid_to_str(oldloc[0].dpid),
```

```
oldloc[1],
                      dpid_to_str(          loc[0].dpid),
loc[1])
            if packet.src.is_multicast == False:
              mac_map[packet.src] = loc # Learn position
for ethaddr
              log.debug("Learned     %s     at     %s.%i",
packet.src, loc[0], loc[1])

          elif packet.dst.is_multicast == False:
            if packet.dst in mac_map:
              log.warning("Packet   from   %s   to   known
destination %s arrived "
                          "at   %s.%i   without   flow",
packet.src, packet.dst,
                          dpid_to_str(self.dpid),
event.port)

        if packet.dst.is_multicast:
          log.debug("Flood     multicast     from     %s",
packet.src)
          flood()
        else:
          if packet.dst not in mac_map:
            log.debug("%s   unknown   --   flooding"   %
(packet.dst,))
            flood()
          else:
            dest = mac_map[packet.dst]
            match = of.ofp_match.from_packet(packet)
            self.install_path(dest[0],   dest[1],   match,
event)

      def disconnect (self):
        if self.connection is not None:
          log.debug("Disconnect %s" % (self.connection,))

self.connection.removeListeners(self._listeners)
          self.connection = None
          self._listeners = None

      def connect (self, connection):
        if self.dpid is None:
          self.dpid = connection.dpid
        assert self.dpid == connection.dpid
        if self.ports is None:
          self.ports = connection.features.ports
        self.disconnect()
```

```
        log.debug("Connect %s" % (connection,))
        self.connection = connection
        self._listeners = self.listenTo(connection)
        self._connected_at = time.time()

      @property
      def is_holding_down (self):
        if self._connected_at is None: return True
        if    time.time()    -    self._connected_at    >
FLOOD_HOLDDOWN:
          return False
        return True

      def _handle_ConnectionDown (self, event):
        self.disconnect()

    class l2_multi (EventMixin):
      _eventMixin_events = set([
        PathInstalled,
      ])

      def __init__ (self):
        def startup ():
          core.openflow.addListeners(self, priority=0)
          core.openflow_discovery.addListeners(self)
        core.call_when_ready(startup,
('openflow','openflow_discovery'))

      def _handle_LinkEvent (self, event):
        def flip (link):
          return           Discovery.Link(link[2],link[3],
link[0],link[1])

        l = event.link
        sw1 = switches[l.dpid1]
        sw2 = switches[l.dpid2]

        clear = of.ofp_flow_mod(command=of.OFPFC_DELETE)
        for sw in switches.itervalues():
          if sw.connection is None: continue
          sw.connection.send(clear)

        if event.removed:
          if      sw2      in      adjacency[sw1]:      del
adjacency[sw1][sw2]
          if      sw1      in      adjacency[sw2]:      del
adjacency[sw2][sw1]
```

```
          for ll in core.openflow_discovery.adjacency:
            if  ll.dpid1  ==  l.dpid1  and  ll.dpid2  ==
l.dpid2:
              if                flip(ll)                in
core.openflow_discovery.adjacency:
                adjacency[sw1][sw2] = ll.port1
                adjacency[sw2][sw1] = ll.port2
                break

        else:
          if adjacency[sw1][sw2] is None:
            if                flip(l)                  in
core.openflow_discovery.adjacency:
              adjacency[sw1][sw2] = l.port1
              adjacency[sw2][sw1] = l.port2
          bad_macs = set()

          for mac,(sw,port) in mac_map.iteritems():
            if   sw   is   sw1   and   port   ==   l.port1:
bad_macs.add(mac)
            if   sw   is   sw2   and   port   ==   l.port2:
bad_macs.add(mac)
          for mac in bad_macs:
            log.debug("Unlearned %s", mac)
            del mac_map[mac]

      def _handle_ConnectionUp (self, event):
        sw = switches.get(event.dpid)
        if sw is None:
          sw = Switch()
          switches[event.dpid] = sw
          sw.connect(event.connection)
        else:
          sw.connect(event.connection)

      def _handle_BarrierIn (self, event):
        wp   =   waiting_paths.pop((event.dpid,event.xid),
None)
        if not wp:
          return
        wp.notify(event)

    def launch ():
      core.registerNew(l2_multi)
    timeout = min(max(PATH_SETUP_TIME, 5) * 2, 15)
     Timer(timeout,     WaitingPath.expire_waiting_paths,
recurring=True)
```

АНОТАЦІЯ

УДК 004.7:004.6+004.415/.416](076.5)=111

В.В. Шкарупило, Р.В. Кудерметов, Д.С. Мазур, І.С. Скарга-Бандурова, Л.О. Шумова, А.Ю. Великжанін, В.С. Харченко, Д.Д. Узун, Ю.О. Узун, П.А. Годованюк. Програмно-конфігуровані мережі та Інтернет Речей: Практикум / За ред. Кудерметова Р.К. – МОН України, Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ». – 129 с.

Викладено матеріали практичної частини курсу PC4 “ Програмно-конфігуровані мережі та IoT ”, підготовленого в рамках проекту ERASMUS+ ALIOT “ Internet of Things: Emerging Curriculum for Industry and Human Applications” (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP).

Матеріали для практикуму повинні використовуватись докторантами у галузі комп'ютерних мереж, інженерії програмного забезпечення тощо та спрямовані на надання необхідних знань та практичних навичок на тему використання емулятора Mininet для вирішення типових інженерних завдань, що охоплюють, зокрема, аспекти програмування - для вирішення завдань автоматизації. Крім того, практикум присвячений інженерам та дослідженням, що займаються розробкою, впровадженням та тестуванням IoT-рішень на основі SDN.

Призначено для інженерів, розробників та науковців, які займаються розробкою та впровадженням IoT систем, для аспірантів університетів, які навчаються за напрямами IoT, кібербезпеки в мережах, комп'ютерних наук, комп'ютерної та програмної інженерії, а також для викладачів відповідних курсів.

Бібл. – 39, рисунків – 65, таблиць - 2.

ЗМІСТ

ABSTRACT

UDC 004.7:004.6+004.415/.416](076.5)=111

Shkarupylo V.V., Kudermetov R.K., Skarga-Bandurova I.S., Velykzhanin A.Yu., Shumova L.O., Mazur D.S., Kharchenko V.S., Uzun D.D., Uzun Y.O., Hodovaniuk P.A. Software defined networks and IoT: Practicum / Kudermetov R.K. (Ed.) – Ministry of Education and Science of Ukraine, Zaporizhzhia National Technical University, Volodymyr Dahl East Ukrainian National University, National Aerospace University "KhAI", 2019. – 129 p.

The materials of the practical part of the study course "PC2. Software defined networks and IoT", developed in the framework of the ERASMUS+ ALIOT project "Modernization Internet of Things: Emerging Curriculum for Industry and Human Applications Domains" (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP).

Practicum materials are supposed to be used by PhD-students in sphere of computer networking, software engineering etc., and aimed at delivering the essential knowledge and practical skills on the topic of Mininet emulator usage for the purpose of typical engineering tasks solving, covering, in particular, the aspects of programming – for the purpose of automation tasks solving. Practicum is devoted to development, implementation and testing of SDN-based IoT-solutions. Moreover, techniques and tools of DevOps application in context IoT and Big Data are described.

Practicum materials are intended to be used by the PhD-students which are studied on computer networking software engineering, engineers and researches involved in the development, implementation and testing of SDN-based IoT-solutions, methodology and techniques of DevOps.

Ref. – 39 items, figures – 65, tables - 2 .

CONTENTS

Вадим Вікторович Шкарупило
Раіль Камілович Кудерметов
Деніс Сергійович Мазур
Інна Сергіївна Скарга-Бандурова
Лариса Олександрівна Шумова
Артем Юрійович Великжанін
Вячеслав Сергійович Харченко
Дмитро Дмитрович Узун
Юлія Олександрівна Узун
Павло Андрійович Годованюк

ПРОГРАМНО-КОНФІГУРОВАНІ МЕРЕЖІ ТА ІНТЕРНЕТ РЕЧЕЙ

Практикум
(англійською мовою)

Редактор Кудерметов Р.К.

Комп'ютерна верстка
Р.К. Кудерметов,
О.О. Ілляшенко

Зв. план, 2019
Підписаний до друку 27.08.2019
Формат 60x84 1/16. Папір офс. No2. Офс. друк.
Умов. друк. арк. 7,09. Уч.-вид. л. 7,62. Наклад 150 прим.
Замовлення 270819-2

Національний аерокосмічний університет ім. М. Є. Жуковського
"Харківський авіаційний інститут"
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu

Випускаючий редактор: ФОП Голембовська О.О.
03049, Київ, Повітрофлотський пр-кт, б. 3, к. 32.
Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
серія ДК No 5120 від 08.06.2016 р.

Видавець: ТОВ «Видавництво «Юстон»
01034, м. Київ, вул. О. Гончара, 36-а, тел.: +38 044 360 22 66
www.yuston.com.ua
Свідоцтво про внесення суб’єкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
серія ДК No 497 від 09.09.2015 р.