**Семенов Анатолій Олексійович**, кандидат фізико-математичних наук, доцент, кафедра товарознавства непродовольчих товарів, Полтавський університет економіки і торгівлі, вул. Коваля, 3, м. Полтава, Україна, 36000
E-mail: a-semenov@li.ru

**UDC 621.391**
**DOI: 10.15587/2313-8416.2014.31734**

# TESTING OF FLOODLIGHT CONTROLLER WITH MININET IN SDN TOPOLOGY

**© Taher Abdullah**

*Програмно-конфігуруєма мережа (ПКМ) є в даний час однією з найбільш перспективних технологій в мобільних мережах передаючих каналів на основі протоколу OpenFlow. OpenFlow надає функціональні вимоги для перенесення логіки управління від комутатора в контролер. У цій статті ми застосовуємо програмне забезпечення Mininet з Floodlight-контролером для перевірки OpenFlow контролера на здатність виконувати свою функцію*
***Ключові слова:*** *SDN Openflow protocol, Floodlight-контролер, комутатор, Mininet, Wireshark*

*The Software Defined Networking (SDN) is currently one of the most promising technologies in mobile backhaul networks based on the OpenFlow protocol. OpenFlow provides a specification to migrate the control logic from a switch into the controller. In this paper we apply Mininet software with Floodlight controller to verify the OpenFlow controller to perform its function*
***Keywords:*** *SDN Openflow protocol, floodlight controller, switch, Mininet, Wireshark*

## 1. Introduction

Software Defined Networking (SDN) is a new path in networking technology, designed to create a high level abstractions and can build hardware and software infrastructure to support new cloud computing applications. SDN is considered to as programming network, because it isolates the surveillance aircraft from the aircraft data and provides an independent and centralized network control [1]. OpenFlow protocol approach SDN, and programmable control flows for network administrators to determine the path that takes the flow from the source to the destination, regardless of the network structure, and flow-based processing is used to forward packets. OpenFlow collected great interest among developers and manufacturers of network routers, servers, and switches.

Network Intelligence is (logically) centralized software based controllers SDN, which keeps a comprehensive view of the network. As a result, the grid appears to applications and drives politics as one logical switch [2]. With SDN, companies and carriers make vendor-independent control over the entire network from the point of view of the logical one, which greatly simplifies network design and operation. SDN also greatly simplifies the network devices themselves, they no longer need to understand and address thousands of standard protocol but simply reviewed the instructions from controllers SDN. Perhaps most importantly, network operators and administrators can configure programmatically this abstraction of a simplified network instead of having to hand over tens of thousands of lines of code configuration across thousands of devices. In addition, benefit from a centralized console for intelligence SDN can alter the behavior of the network in real time and the deployment of new applications and network services within hours or days, not weeks or months required today. By centralized state in the control layer, the shape 1 shows the layers of SDN technology.

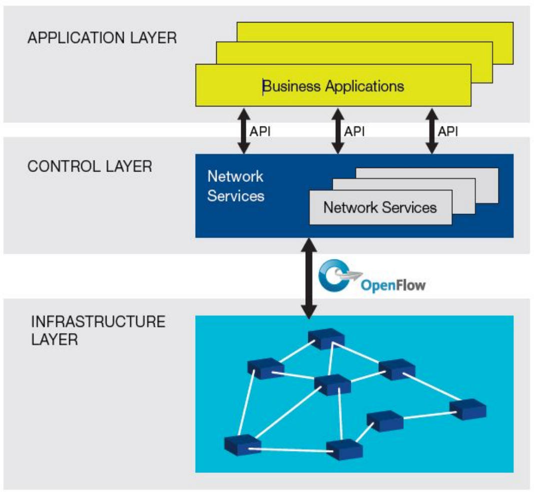


Fig. 1. ONF/SDN architecture [3].

SDN gives network engineers the automated SDN programs, optimize network resources via dynamic, manage, flexibility to configure and secure. Moreover, they can write these programs themselves and not wait for the features to be embedded in the property and closed software vendor environments in the middle of the network as well as to abstracting the network, SDN architectures. It is provided a set of APIs that make it possible to implement common network services, multicast, traffic engineering, including routing, energy usage, all forms of policy management, security, quality of service, processor and storage optimization, custom tailored to meet business objectives and access control. For example, SDN architecture makes it simple to determine and the imposition of consistent policies across both wireless connections on a campus and wire.

**2. Formulation of the problem**

SDN technology is currently under development. It is needed a lot of research and testing, because it will change a lot of the concepts of network engineering and the author presents his analysis study to understand the relationship between the layers SDN technology and test traffic generation of the networks.

**3. Literature review**

The original idea of SDN is first described at Stanford University around 2005. The SDN concept brings the separation of network device features to the control plane, and the data plane [4]. While the control plane is programmatically accessible through well-defined API (Application Programming Interface), data plane ensures a data processing according to the rules uploaded to the device. OpenFlow has been developed since 2007, and the first protocol specification was approved in 2009. Lately, the development was adopted by the Open Networking Foundation (ONF) consortium. The protocol defines the structure of control messages and it describes the way how messages are exchanged. OpenFlow is based on the centralized approach with a controller as a main driving element. This controller runs a software platform with the API enabling the direct control of data flows in a network.

**4. Software-Defined Network architecture**

The SDN architecture is flexible significantly. It can work with different types of switches and the various protocol layers [5]. The controllers and switches can be implemented for Ethernet switches (Layer 2), Internet routers (Layer 3), transport (Layer 4) switching, or application layer switching and routing. SDN depends on the functions commonly found on networking devices, which mainly involves forwarding packets based on the form of the flow definition.

OpenFlow switch includes one or more flow tables and a group table, which performance packet searches and forwarding, and an Open Flow channel to an external controller [6]. The switch contact with the controller and the controller manages the switch via the OpenFlow protocol. Using the OpenFlow protocol, the controller can delete, update, and add flow entries in flow tables, whether reactively (in answer to packets) and proactively. Each flow table in the switch includes a set of flow entries, each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets. We can see this table in Fig. 2.

The first matching entry in the table is used when matching if found the entry, the instructions linked with a certain flow. If no match is found in the table of flow, and the result depends on the configuration of the flow table miss, for example, the packet can be forwarded to the controller via the channel OpenFlow, dropped, or may continue to the next flow table. Instructions linked with each login flow have either procedures or modify the processing pipeline. Actions existing in instructions shows packet forwarding, group table processing and packet modification. Processing instruction pipeline allows packets to be sent to the following tables for further processing and allow information, in the form of metadata, to send between tables. Stop valves address table when the associated instruction set entry flow matches the next table outlines; at this stage usually muddied package and send. The flow of entries submitted to the port. It is mostly by physical port, but it can be also logical port specific by the switch or may by reserved port specified by this specification. Reserved ports may be designated by generic forwarding actions such as giving to the controller, flooding, or forwarding using non-OpenFlow methods, like normal switch processing, during switch-defined logical ports may be designated link aggregation groups, tunnels or loopback interfaces. Actions linked with flow entries may also direct packets to a group, which sets additional processing. Groups are sets of actions for flooding, as well as more complex shipping semantics (e. g., multiple tracks, fast forwarding, and link aggregation). The layer of indirection, and empower communities multiple flow entries are also forward to one ID (for example, IP forwarding to the next joint). This abstraction allows the joint production process via the entrances to flow efficiently. Table group contains entries of the group; each entry contains a list of buckets work with specific semantics depend on the collection type. The procedures are applied in one or more of the buckets of packets sent to the group. Switch designers are free to implement internal parts in any convenient way, provided that the correct match is kept and semantics. For example, while the entry flow may use all set to forward to multiple ports, a designer switch may choose to implement this as a bit mask in the hardware forwarding table. Another example is matching. Pipeline exposed by switching the OpenFlow could be implemented physically with a number of devices.

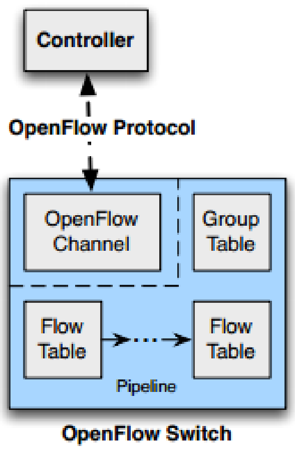


Fig. 2. Main components of OpenFlow switch

**5. Mininet with Floodlight Controller**

**5.1. Mininet Program**

Mininet is a network simulator. Through this program we can create the network infrastructure of the switch's Openflow, hosts and links, linking this topology with controller to add flow entries to control all of these networks. Mininet hosts run standard software, Linux and network switches support OpenFlow flexibility of high custom forwarding and Software-Defined Networking [7].

Mininet helps researchers to testing, developing, learning, debugging and other functions that can benefit from an experimental network on a laptop or other computer. Mininet supports the easy way to learn the system behavior and experiment with topology.

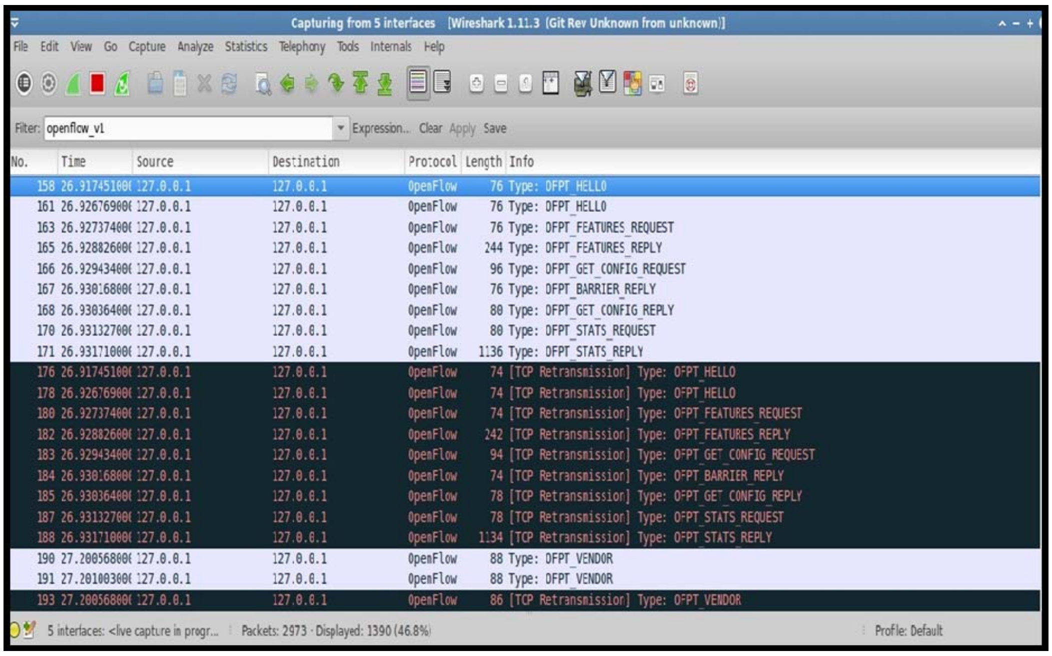
Mininet builds the virtual networks using domain based on the principle virtualization and network - features that are existing in modern Linux kernels. in Mininet hosts are simulated as bash processes running in a network domain , therefore, any code that will normally operate on a Linux server (like a client program or web server ) should start just fine within a Mininet "Host". The Host in “Mininet " has their own special network interface and can only see their own processes. The Mininet "Switches" software-based switches like OpenSwitch or switch reference OpenFlow. Links are the default Ethernet couples living in the Linux kernel and connecting our keys for the hosts to follow (operations).

We need to install these files separately .The files include virtualization program, a SSH-capable terminal, an X server, and the VM image. The tutorial image program is publishing as a compressed VirtualBox image (vbi). VirtualBox enables you to run a virtual machine inside a physical machine, and is open source image and available for Windows, Mac and Linux [8].

## 5. 2. Messages between controller and switch by OpenFlow protocol

The link between controller and switch Openflow allows using the OpenFlow protocol, where a set of specific messages can be exchanged between these actors over a secure channel [9].The secure channel is the ports that link each OpenFlow switch to a controller. The controller begins to communicate in the event of the switch on its power on. The controller's default TCP port is 6633. Switch controller authentication of the parties through an exchange of letters signed by the private key of the specified location. Each switch must have user configurable with one certificate for the authentication controller (controller certificate) and the other for authentication on the controller (a switch).

The OpenFlow protocol provides three message types: controller-to-switch, asynchronous, and symmetric, each with multiple sub-types [10]. Controller-to-switch messages are initiated by the controller and used to directly administration or viewed the state of the switch. Asynchronous messages are initiated by the switch and used to tell the controller of network activates and changes to the switch state. Symmetric messages are started by either the switch or the controller and sent without petition. The message types used by OpenFlow are described below. We can see this message by Wireshark program in Fig. 3.


Capturing from 5 interfaces [Wireshark 1.11.3 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: openflow_v1
Expression... Clear Apply Save

```
No.  Time         Source     Destination  Protocol Length Info
 158 26.91745100( 127.0.0.1  127.0.0.1    OpenFlow     76 Type: OFPT_HELLO
 161 26.92676900( 127.0.0.1  127.0.0.1    OpenFlow     76 Type: OFPT_HELLO
 163 26.92737400( 127.0.0.1  127.0.0.1    OpenFlow     76 Type: OFPT_FEATURES_REQUEST
 165 26.92882600( 127.0.0.1  127.0.0.1    OpenFlow    244 Type: OFPT_FEATURES_REPLY
 166 26.92943400( 127.0.0.1  127.0.0.1    OpenFlow     96 Type: OFPT_GET_CONFIG_REQUEST
 167 26.93016800( 127.0.0.1  127.0.0.1    OpenFlow     76 Type: OFPT_BARRIER_REPLY
 168 26.93036400( 127.0.0.1  127.0.0.1    OpenFlow     80 Type: OFPT_GET_CONFIG_REPLY
 170 26.93132700( 127.0.0.1  127.0.0.1    OpenFlow     80 Type: OFPT_STATS_REQUEST
 171 26.93171000( 127.0.0.1  127.0.0.1    OpenFlow   1136 Type: OFPT_STATS_REPLY
 176 26.91745100( 127.0.0.1  127.0.0.1    OpenFlow     74 [TCP Retransmission] Type: OFPT_HELLO
 178 26.92676900( 127.0.0.1  127.0.0.1    OpenFlow     74 [TCP Retransmission] Type: OFPT_HELLO
 180 26.92737400( 127.0.0.1  127.0.0.1    OpenFlow     74 [TCP Retransmission] Type: OFPT_FEATURES_REQUEST
 182 26.92882600( 127.0.0.1  127.0.0.1    OpenFlow    242 [TCP Retransmission] Type: OFPT_FEATURES_REPLY
 183 26.92943400( 127.0.0.1  127.0.0.1    OpenFlow     94 [TCP Retransmission] Type: OFPT_GET_CONFIG_REQUEST
 184 26.93016800( 127.0.0.1  127.0.0.1    OpenFlow     74 [TCP Retransmission] Type: OFPT_BARRIER_REPLY
 185 26.93036400( 127.0.0.1  127.0.0.1    OpenFlow     78 [TCP Retransmission] Type: OFPT_GET_CONFIG_REPLY
 187 26.93132700( 127.0.0.1  127.0.0.1    OpenFlow     78 [TCP Retransmission] Type: OFPT_STATS_REQUEST
 188 26.93171000( 127.0.0.1  127.0.0.1    OpenFlow   1134 [TCP Retransmission] Type: OFPT_STATS_REPLY
 190 27.20056800( 127.0.0.1  127.0.0.1    OpenFlow     88 Type: OFPT_VENDOR
 191 27.20100300( 127.0.0.1  127.0.0.1    OpenFlow     88 Type: OFPT_VENDOR
 193 27.20056800( 127.0.0.1  127.0.0.1    OpenFlow     86 [TCP Retransmission] Type: OFPT_VENDOR

5 interfaces: <live capture in progr...   Packets: 2973 · Displayed: 1390 (46.8%)   Profile: Default
```

Fig. 3. Message between controller and switches by Wireshark program. (Done by the author)

➢**Symmetric**

Symmetric messages are sent without solicitation, in either direction.

**Hello**: Hello messages are exchanged between the switch and controller upon connection startup.

**Echo**: Echo request/reply messages can be sent from either the switch or the controller, and must return an echo reply. They are mainly used to verify the liveness of a controller-switch connection, and may as well be used to measure its latency or bandwidth.

**Experimenter**: Experimenter messages provide a standard way for OpenFlow switches to offer additional functionality within the OpenFlow message type space. This is a staging area for features meant for future OpenFlow revisions.

➢**Controller-to-Switch**

Controller/switch messages are initiated by the controller and may or may not require a response from the switch.

**Features**: The controller may request the capabilities of a switch by sending a features request.

**Configuration**: The controller is able to set and query configuration parameters in the switch.

**Modify-State**: Modify-State messages are sent by the controller to manage state on switches.

**Read-State**: Read-State messages are used by the controller to collect various information from the switch, such as current configuration, statistics and capabilities.

**Packet-out**: These are used by the controller to send packets out of a specified port on the switch, and to forward packets received via Packet-in messages.

**Barrier**: Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.

**Role-Request**: Role-Request messages are used by the controller to set the role of its OpenFlow channel, or query that role.

**Asynchronous-configuration**: The Asynchronous-configuration message is used by the controller to set an additional filter on the asynchronous messages that it wants to receive on its OpenFlow channel, or to query that filter.

➢ **Asynchronous**

Asynchronous messages are sent without a controller soliciting them from a switch. Switches sent asynchronous messages to controllers to denote a packet arrival, switch state change, or error. The four main asynchronous message types are described below.

**Packet-in**: Transfer the control of a packet to the controller. For all packets forwarded to the controller reserved port using a flow entry or the table-miss flow entry, a packet-in event is always sent to controllers.

**Flow-Removed**: Inform the controller about the removal of a flow entry from a flow table

**Port-status**: Inform the controller of a change on a port.

**Error**: The switch is able to notify controllers of problems using error messages.

### 5. 3. Controller Adds Flow Entries in Switch

The controller adds flow entries for switch by secure channel that managed by Openflow protocol. When reach to switch the flow entries for switch puts in flow table, switch uses the flow entries for taking decision of forwarding packets[11]. We can see flow entries by Mininet in Fig. 4. Main components of a flow entry in a flow table [12].

• **Match fields**: to match against packets. It consist of the ingress port and packet headers, and optionally metadata specified by a previous table.

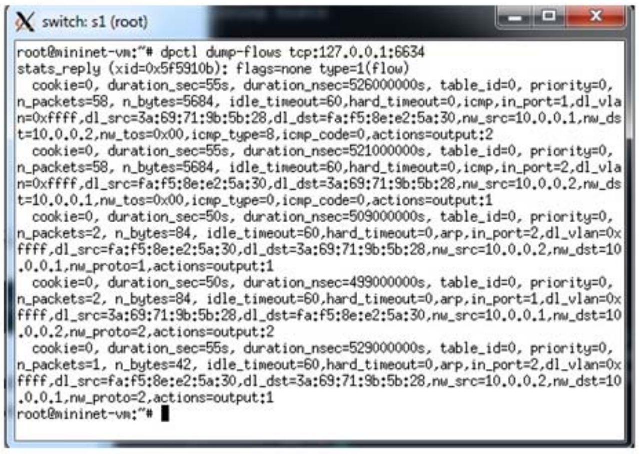
• **Priority**: matching precedence of the flow entry.

• Counters: updated when packets are matched.

• **Instructions**: to modify the action set or pipeline processing.

• **Timeouts**: maximum amount of time or idle time before flow is expired by the switch.

• **Cookie**: opaque data value chosen by the controller. It may be used by the controller to filter flow statistics, flow modification and flow deletion. It is not used when processing packets.



```
root@mininet-vm:~# dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x5f5910b): flags=none type=1(flow)
  cookie=0, duration_sec=55s, duration_nsec=526000000s, table_id=0, priority=0,
n_packets=58, n_bytes=5684, idle_timeout=60,hard_timeout=0,icmp,in_port=1,dl_vla
n=0xffff,dl_src=3a:69:71:9b:5b:28,dl_dst=fa:f5:8e:e2:5a:30,nw_src=10.0.0.1,nw_ds
t=10.0.0.2,nw_tos=0x00,icmp_type=8,icmp_code=0,actions=output:2
  cookie=0, duration_sec=55s, duration_nsec=521000000s, table_id=0, priority=0,
n_packets=58, n_bytes=5684, idle_timeout=60,hard_timeout=0,icmp,in_port=2,dl_vla
n=0xffff,dl_src=fa:f5:8e:e2:5a:30,dl_dst=3a:69:71:9b:5b:28,nw_src=10.0.0.2,nw_ds
t=10.0.0.1,nw_tos=0x00,icmp_type=0,icmp_code=0,actions=output:1
  cookie=0, duration_sec=50s, duration_nsec=509000000s, table_id=0, priority=0,
n_packets=2, n_bytes=84, idle_timeout=60,hard_timeout=0,arp,in_port=2,dl_vlan=0x
ffff,dl_src=fa:f5:8e:e2:5a:30,dl_dst=3a:69:71:9b:5b:28,nw_src=10.0.0.2,nw_dst=10
.0.0.1,nw_proto=1,actions=output:1
  cookie=0, duration_sec=50s, duration_nsec=499000000s, table_id=0, priority=0,
n_packets=2, n_bytes=84, idle_timeout=60,hard_timeout=0,arp,in_port=1,dl_vlan=0x
ffff,dl_src=3a:69:71:9b:5b:28,dl_dst=fa:f5:8e:e2:5a:30,nw_src=10.0.0.1,nw_dst=10
.0.0.2,nw_proto=2,actions=output:2
  cookie=0, duration_sec=55s, duration_nsec=529000000s, table_id=0, priority=0,
n_packets=1, n_bytes=42, idle_timeout=60,hard_timeout=0,arp,in_port=2,dl_vlan=0x
ffff,dl_src=fa:f5:8e:e2:5a:30,dl_dst=3a:69:71:9b:5b:28,nw_src=10.0.0.2,nw_dst=10
.0.0.1,nw_proto=2,actions=output:1
root@mininet-vm:~# 
```

Fig. 4. Flow entries (Done by the author)

### 6. Design of topologies in Mininet

The Mininet is one of many simulator network programs for the SDN technology and the most common provides many of the most important features that hasn't a lot of resources and just needs a laptop to create hundreds of nodes and tested, is an open source program needs topologies that are defined in Python for its execution, built with some topological simple and can create the appropriate topology for the test by typing some lines in the Python language. So each usable Mininet topology is similar, having the same content in the head and the tail of a file, in Mininet version 2.1.0 not need for remote controller because there are some controllers as pox and nox but when need to use floodlight controller must be separate for Mininet.

Mininet can use to modify of the bandwidth in Mbps is also possible by setting the B.w value of all hosts to the given value. Further, the parser requires the files to be located in the same directory and without specifying input parameters the program will terminate. However, some values can be deleted, like the bandwidth limitation, which is otherwise initialized to 10 Mbps. In this case the remote controller IP is initialized with "10.0.2.2" which is the standard IP for the host OS when using Oracle Virtualbox for virtualization.

### 6.1. Floodlight controller

The Floodlight Open SDN Controller is a company-class, open source, Java-based OpenFlow Controller. It is supported by a group of developers among them of engineers from Big Switch Networks. OpenFlow protocols are an open standard managed by ONF. It specifies a protocol through switch a remote controller can modify the behavior of networking devices through a well-defined "forwarding instruction set"

Floodlight is designed to work with the increasing number of switches, routers, virtual switches, and access points that support the OpenFlow standard [13].

Feature Floodlight:

- Offered module loading system makes it easy to extend and improvements.
- Simple to set up with minimum accreditation.
- Support a wide range of virtual and physical OpenFlow switches.

- Possibility of implementation mixed OpenFlow and non-OpenFlow networks it can manage Multiple of OpenFlow switches.
- High-performance design is the essence of a commercial product from Big Switch Networks.
- Support for OpenStack (link) cloud synchronization platform.

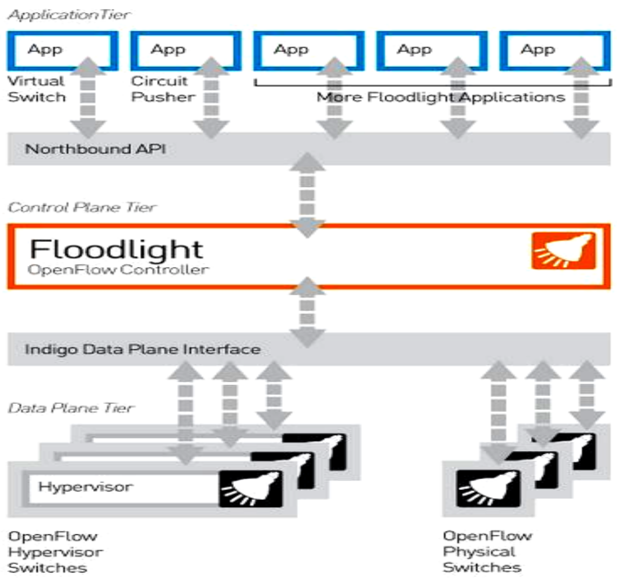


Fig. 5. Floodlight OpenFlow controller [12]

### 6. 2. Internet Traffic Generator (D-ITG)

To evaluate the performance of Mininet D-ITG in version 2.8.0-rc1 was used: "Distributed Internet Traffic Generator (D-ITG) – platform capable to produce traffic that accurately adheres to patterns defined by the inter departure time between packets (IDT) and the packet size (PS) stochastic processes" [14]. Therefore, it offers a rich variety of probability distributions for the traffic generation and uses some models proposed to emulate sources of various protocols. With this, it is possible to generate various packet streams and collect statistics with a logging server. Important modules of the D-ITG are depicted. The ITGSend module is responsible for the traffic generation, while the ITGRecv module is the sink for the packets, which are delivered over a data channel. To collect logging information both, the ITGSend and ITGRecv are communicated via a Log Channel with the ITGLog module. For remote control the ITGManager offers the functionalities to adjust parameters of ITGSend through the Signaling Channel.

### 6. 3. Measurement Trials

Fig. 6 depicts the topology of our measurement trials. S1 to S6 are the switches in the topology, while sender/receiver denotes the hosts that are handling generated traffic. On each sender of ITGSend process is called to generate the traffic, while on each receiver ITGRecv handles the receipt of the packets. Besides, the log server collects the relevant statistical data of the hosts by running an instance of ITGLog. S4 is shutdown for a few measurement trials and therefore all dashed links are unavailable. To sum up, in total we performed trials, each for TCP and UDP with a duration of one minute for each trial.

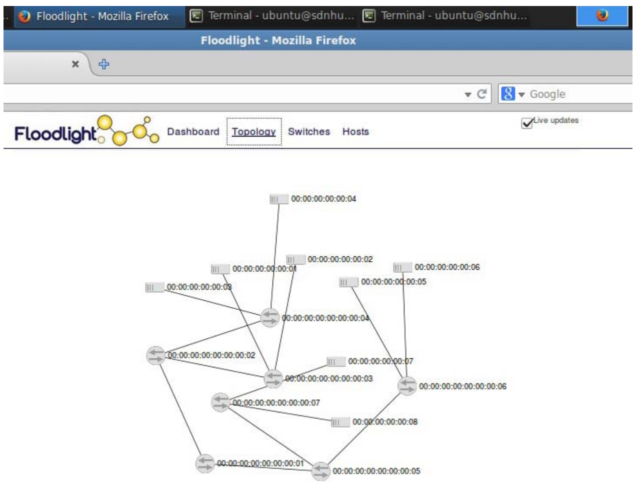


Fig. 6. Topology in web interface Floodlight controller (Done by the author)

The D-ITG decoder provides data files that can be analyzed with MatLab. Initially, every trial is evaluated and plotted as shown in Fig. 7. The graphs show specific characteristics in the time sequence from 0 s to 60 s in the plots. In more detail, the upper left plot depicts the throughput in Mbps, while the upper right shows the delay in ms. The lower left plot is evaluating the jitter value in ms.

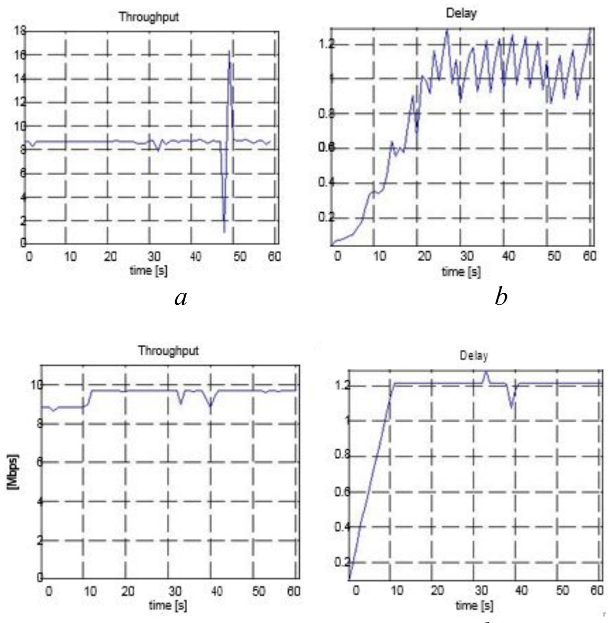


Fig. 7. Evaluation of traffic with a constant test bitrate of 10Mbps. The delay within the edges of the SDN topology is 1ms: *a, b* – traffic generation over TCP; *c, d* – traffic generation over UDP

### 7. Conclusion

This paper presents the SDN technology. We have analyzed all messages between switch OpenFlow and Floodlight controller and captured in Wireshark that allows us to understand the relationship between controller and switch Openflow and relationship between switches.

In second part we have used the Mininet simulators and built by the new topology consists of

7 switches and 8 host in the Python languages, all this devices controlled by Floodlight controller through add flow entry for switches to teach switches how reach for each other and D-ITG program for generate traffic between devices. We show the results in graphs.

**References**

1. Khatri, V. Khatri vikramajeet analysis of openflow protocol in local area Net – degree programme in information technology [Text] / V. Khatri. – Tampere University of technology, 2013. – 74 p.
2. Open networking Foundation Software-Defined Networking [Text] / The New Norm for Networks, 2012.
3. OpenFlow-enabled SDN and Network Functions Virtualization [Text] / Open networking Foundation, 2014.
4. Hegr, T. OpenFlow Deployment and Concept Analysis [Text] / T. Hegr, L. Bohac, V. Uhlir, P. Chlumsky // Advances in Electrical and Electronic Engineering. – 2013. – Vol. 11, Issue 5. – P. 327–335. doi: 10.15598/aeee.v11i5.884
5. Octopress, M. T.-P. A Quarterly Technical Publication for Internet and Intranet Professionals [Text] / M. T.-P. Octopress. – A Quarterly Technical Publication for Internet and Intranet Professionals, 2013
6. Open Networking Foundation [Text] / OpenFlow Switch Specification, 2013. – P. 1–205.
7. Mininet Team-Powered by Octopress [Electronic resource] / Mininet Overview, 2014. – Available at: http://mininet.org/overview/
8. Openflow [Text] / Explain all the requirements to run Mininet, 2011.
9. Azodolmolky, S. Software Defined Networking with OpenFlow [Text] / S. Azodolmolky. – Packt Publishing, 2013. – 153 p.
10. Open Networking Foundation [Text] / OpenFlow Switch Specification. – 2014. – Vol. 4. – P. 1–171.
11. Kontesidou, G. Openflow Virtual Networking : A Flow – Based Network Virtualization Architecture Openflow Virtual Networking : A Flow-Based [Text] / G. Kontesidou, K. Zarifis. – Royal Institute of Technology, 2009.
12. Technical Solution Guide [Text] / HP OpenFlow Protocol Overview, 2013. – 18 p.
13. Opendaylight [Electronic resource] / Available at: http://www. projectfloodlight.org/floodlight/
14. Avallone, S. D-ITG Distributed Internet Traffic Generator [Text] / S. Avallone, S. Guadagno, D. Emma. – University's di Napoli Federico II COMICS Lab, Department di Informatics Sistemistica, 2004. – 4 p.

**Refernces**

1. Khatri, V. (2013). Khatri vikramajeet analysis of openflow protocol in local area Net – degree programme in information technology. Tampere University of technology, 74.
2. Open networking Foundation Software-Defined Networking (2012). The New Norm for Networks.
3. OpenFlow-enabled SDN and Network Functions Virtualization (2014). Open networking Foundation.
4. Hegr, T., Bohac, L., Uhlir, V., Chlumsky, P. (2013). OpenFlow Deployment and Concept Analysis. Advances in Electrical and Electronic Engineering, 11 (5), 327–335. doi: 10.15598/aeee.v11i5.884
5. Octopress, M. T.-P. (2013). A Quarterly Technical Publication for Internet and Intranet Professionals. A Quarterly Technical Publication for Internet and Intranet Professionals.
6. Open Networking Foundation (2013). OpenFlow Switch Specification, 1–205.
7. Mininet Team-Powered by Octopress (2014). Mininet Overview. Available at: http://mininet.org/overview/
8. Openflow (2011). Explain all the requirements to run Mininet.
9. Azodolmolky, S. (2013). Software Defined Networking with OpenFlow. Packt Publishing, 153.
10. Open Networking Foundation (2014). OpenFlow Switch Specification, 4, 1–171.
11. Kontesidou, G., Zarifis, K. (2009). Openflow Virtual Networking : A Flow- Based Network Virtualization Architecture Openflow Virtual Networking : A Flow-Based. Royal Institute of Technology.
12. Technical Solution Guide (2013). HP OpenFlow Protocol Overview, 18.
13. Opendaylight. Available at: http://www. projectfloodlight.org/floodlight/
14. Avallone, S., Guadagno, S., Emma, D. (2004). D-ITG Distributed Internet Traffic Generator. University's di Napoli Federico II COMICS Lab, Department di Informatics Sistemistica, 4.

*Рекомендовано до публікації д-р техн. наук Тіхонов В. І.*
*Дата надходження рукопису 13.11.2014*

**Taher Abdullah,** PhD, Telecommunication systems department, Odessa National Academy of Telecommunications named after O. S. Popov, st. Blacksmith, 1, Odesa, Ukraine, 65000
E-mail: abidalla_2004@yahoo.com

**УДК 004.925**
**DOI: 10.15587/2313-8416.2014.**

## ОБ ЭФФЕКТИВНОМ ИСПОЛЬЗОВАНИИ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ СОЗДАНИЯ ИНТЕРАКТИВНЫХ ВЕБ-ДОКУМЕНТОВ

**© И. Н. Егорова, А. С. Трофименко**

*Статья посвящена исследованию наиболее востребованных современных технологий разработки веб-документов, таких как: API Canvas, WebGL, SVG и CSS3. Рассмотрены достоинства и недостатки каждой из технологий в визуальном представлении информации: анимации, интерфейсах, 2D и 3D графике. Проведен анализ областей их применения, разработаны рекомендации по наиболее эффективному их использованию для разработки интерактивных WEB-документов*

***Ключевые слова:*** *веб-документ, интернет, интерактивность, разработка, спецификация, HTML5, SVG, векторная графика, WebGL*