Skoltech

Skolkovo Institute of Science and Technology

Lomonosov Moscow State University

Software-Defined Networks (SDN)

Lecture 6: SDN switches. Mininet. OpenVSwitch

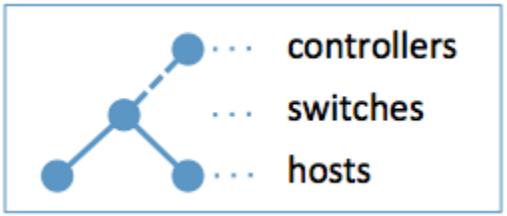
Vasily Pashkov

pashkov@lvk.cs.msu.su

1. Mininet

Mininet

Goal: OpenFlow / SDN network on your desktop / laptop.



Features:

- An excellent tool for modeling OpenFlow / SDN network when prototyping and developing controller and applications for it.
- Fast creation, configuration, interaction with OpenFlow / SDN network
- Node (host or switch) = process in Linux user space:
  - scalable to 100 nodes per PC.
  - You can run any Unix application (ping, iperf, etc.) on the hosts.
  - OpenFlow software switches (OpenVSwitch)

Skoltech
Skolkovo Institute of Science and Technology

Mininet installation

- Sudo apt-get install git
- Clone repository:

  git clone git://github.com/mininet/mininet

Mininet Version:

cd mininet

git tag # Список доступных версий

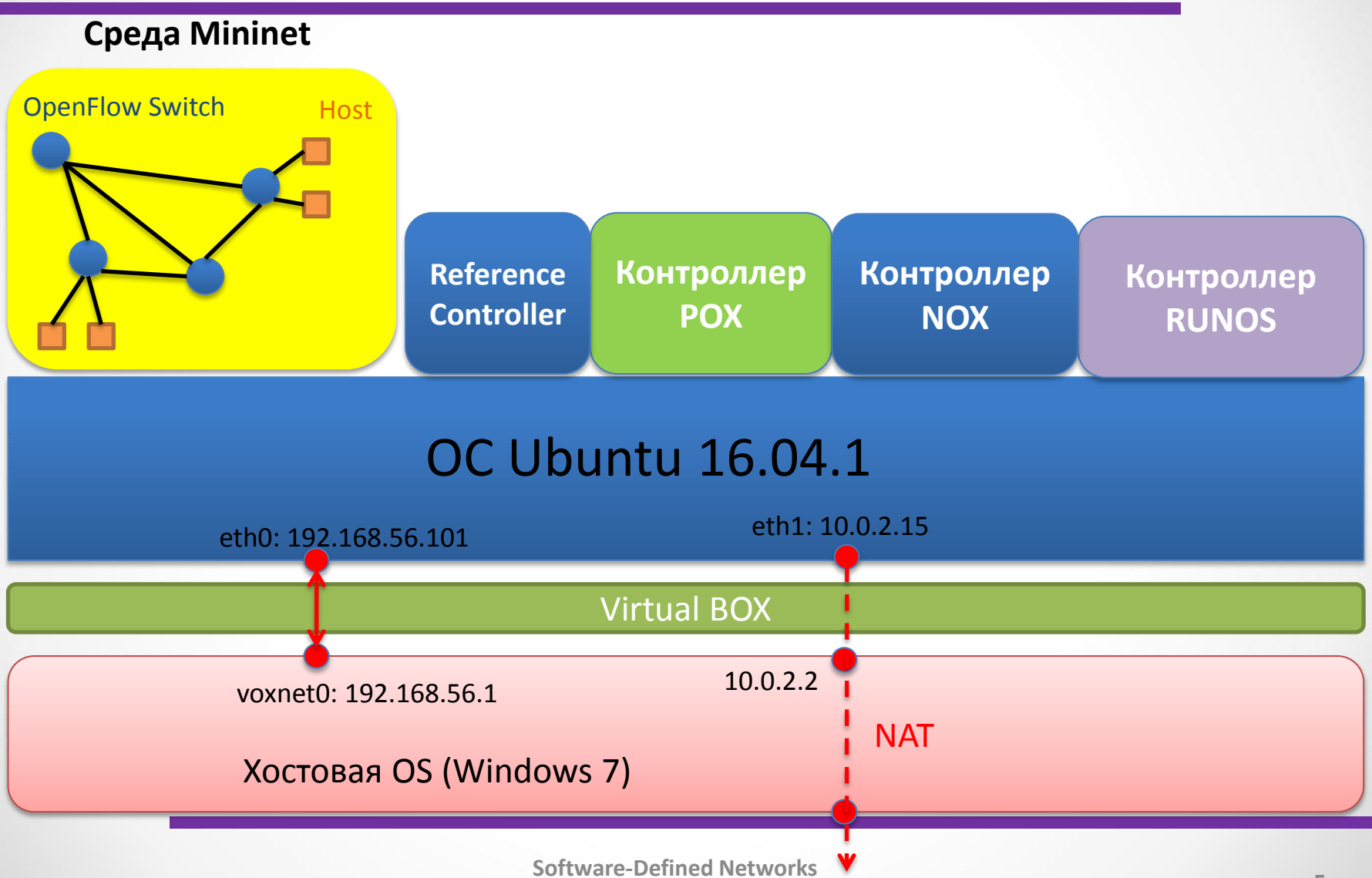
git checkout -b 2.2.1 2.2.1 # установить нужную версию

cd ..

- Запуск процесса установки

  sudo **./mininet/util/install.sh**

Stand


Среда Mininet
OpenFlow Switch
Host
Reference Controller
Контроллер POX
Контроллер NOX
Контроллер RUNOS
ОС Ubuntu 16.04.1
eth0: 192.168.56.101
eth1: 10.0.2.15
Virtual BOX
voxnet0: 192.168.56.1
10.0.2.2
NAT
Хостовая OS (Windows 7)

First launch

- Minimal command:

  sudo mn

- For exit: CLI:

  exit

- Clear all:

  sudo mn -c

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

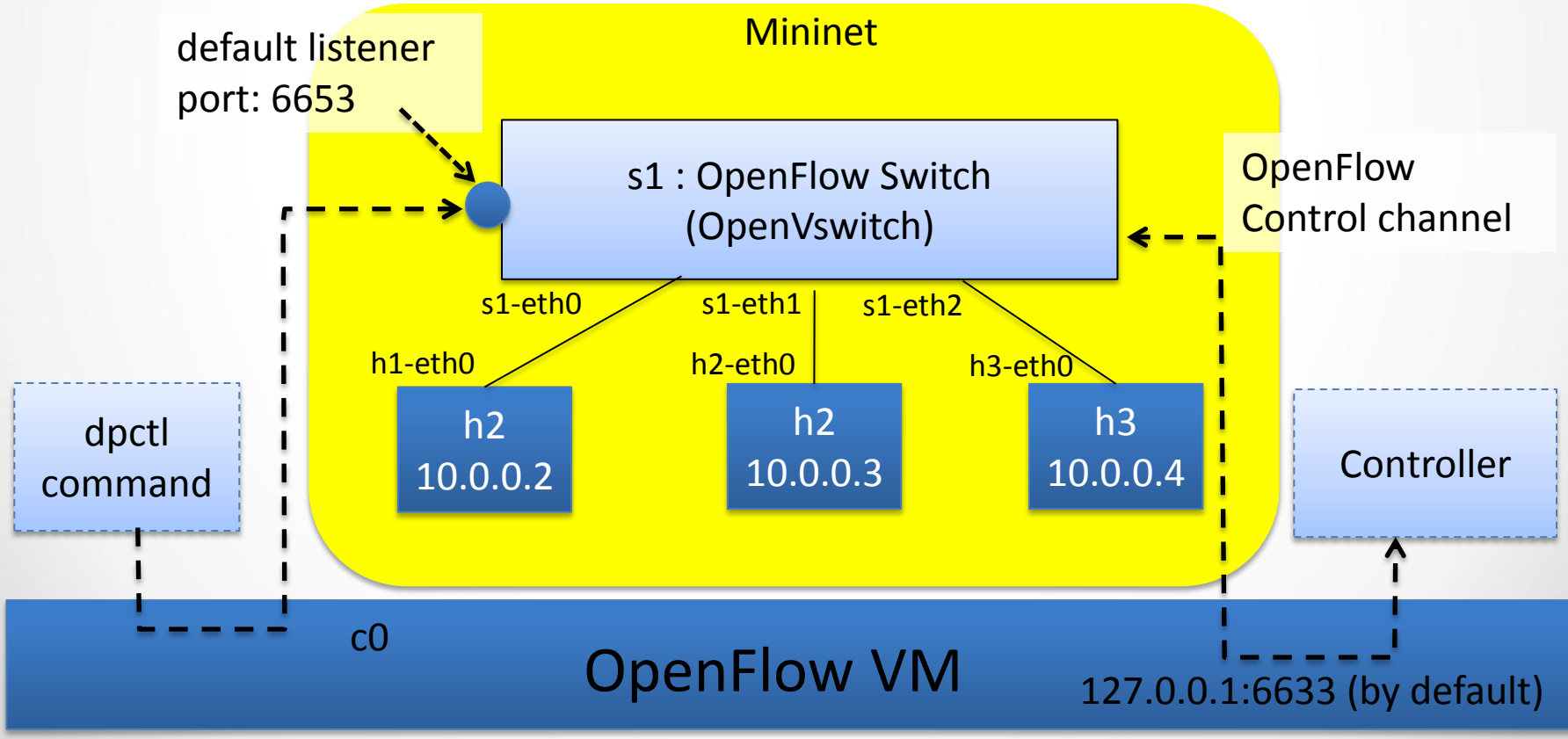
CLI Mininet commands

- *Show help:*
  - ***mininet> help***
- *Show nodes list:*
  - ***mininet> nodes***
- *Show channels:*
  - ***mininet> net***
- *Print dump for all nodes:*
  - ***mininet> dump***

Топология single

```
ubuntu@student:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

- mininet (mn) has to run by root
- '--topo single, 3' : creates pre-defined 3 hosts 1 switch topology (see below)
- '--controller remote': Set the controller to localhost:6633
- host h_x has IP address 10.0.0.x and mac address 00:00:00:00:00:y (y = x in hex)



Mininet

```
openflow@openflowtutorial:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
*** Adding controller
*** Creating network
*** Adding hosts:
h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(s1, h2) (s1, h3) (s1, h4)
*** Configuring hosts
h2 h3 h4
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

CLI Mininet

```
mininet> help
```
Show help

```
mininet> nodes
available nodes are:
h2 h3 h4 s1 c0
```
List Nodes

```
mininet> intfs
c0:
s1: s1-eth1 s1-eth2 s1-eth3
h2: h2-eth0
h3: h3-eth0
h4: h4-eth0
```
List Interfaces of each nodes

```
mininet> net
s1 <-> h2-eth0 h3-eth0 h4-eth0
mininet> h2 ping h3
mininet> h2 ping -c 3 h3
^CPING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2003ms
mininet> xterm h2 h3 h4
mininet>
```
Ping from h2 to h3 (Q: why we don't have ping reply at all?)

Open xterm on h2, h3 and h4

Choosing a network topology

linear|minimal|reversed|single|torus|tree

- **Linear topology (linear)**

sudo mn --topo linear,4 --switch ovsk,protocols=OpenFlow13 --controller remote,ip=127.0.0.1,port=6653

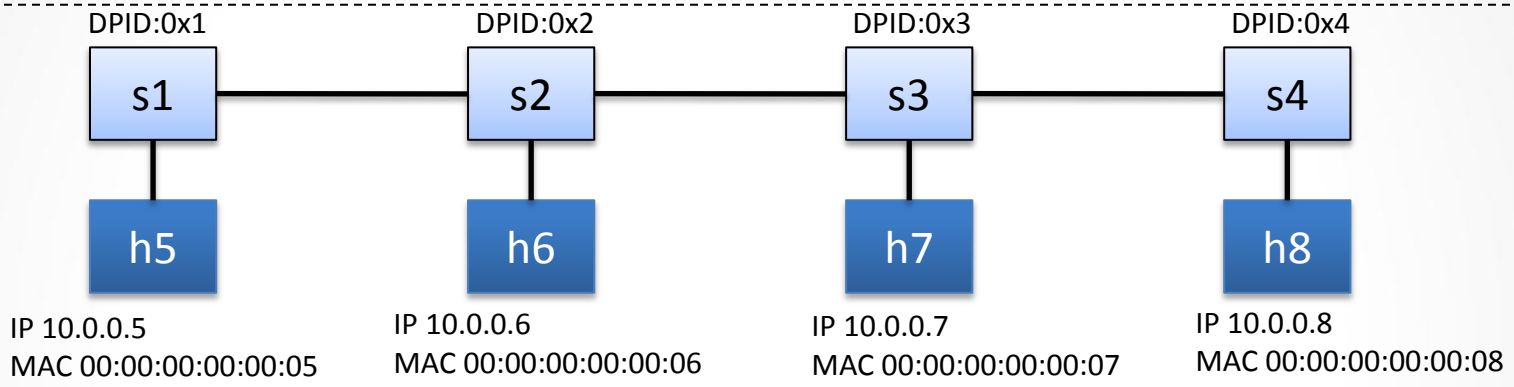
- **Tree topology (tree)**

sudo mn --switch ovsk --controller ref --topo tree,depth=2,fanout=3 --test pingall

More on Mininet (1)

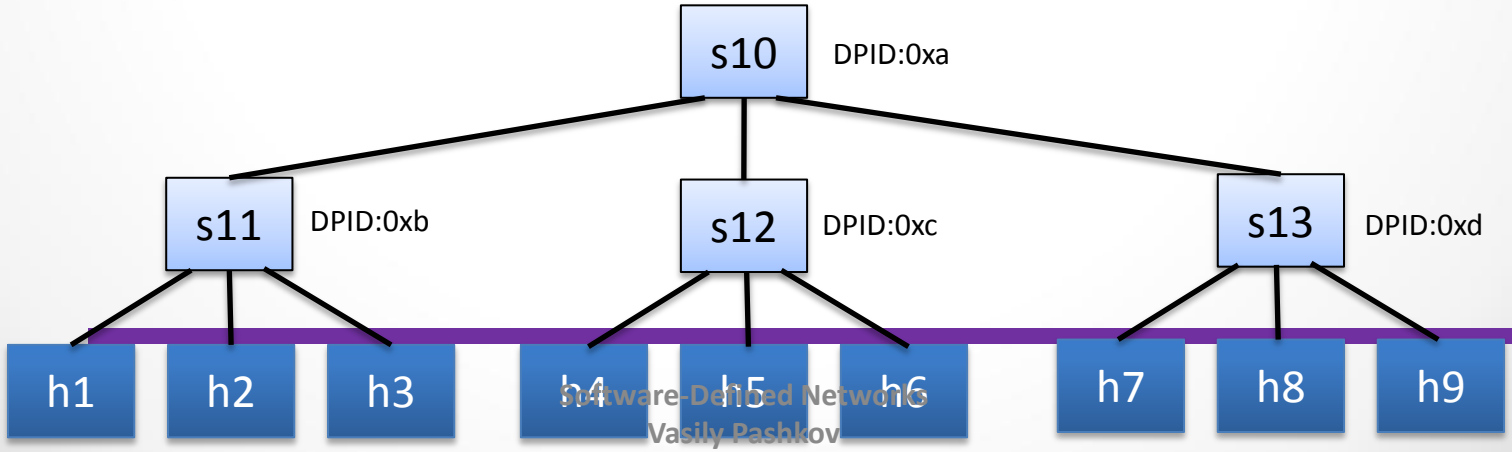
Linear Topology: ‘linear, n’

```
openflow@openflowtutorial:~$ sudo mn --topo linear,4 --mac --switch ovsk --controller remote
```



Tree Topology: ‘tree, n, m’ (n: depth, m: width)

```
openflow@openflowtutorial:~$ sudo mn --topo tree,2,3 --mac --switch ovsk --controller remote
```



Interaction with the network

- *mininet> h2 ping h3*

- *mininet> h2 python -m SimpleHTTPServer 80 >& /tmp/http.log &*
- *mininet> h3 wget -O - h2*
- *…*
- mininet> h2 kill %python

Network configuration

```
from mininet.net import Mininet
from mininet.topolib import TreeTopo
from mininet.node import OVSController

tree4 = TreeTopo(depth=2,fanout=2)
net = Mininet(topo=tree4, controller = OVSController)
net.start()
h1, h4 = net.hosts[0], net.hosts[3]
print h1.cmd('ping -c1 %s' % h4.IP())
net.stop()
```

Ifconfig

- *mininet> h1 ifconfig -a*
- *mininet> s1 ifconfig -a*

Settings for communication channels

$ sudo mn --link tc,bw=10,delay=10ms

mininet> iperf

…

mininet> h1 ping -c10 h2

ID == MAC


```
$ sudo mn
...
mininet> h1 ifconfig
h1-eth0 Link encap:Ethernet HWaddr f6:9d:5a:7f:41:42
                    inet addr:10.0.0.1 Bcast:10.255.255.255
Mask:255.0.0.0
                    UP BROADCAST RUNNING MULTICAST MTU:1500
Metric:1
                    RX packets:6 errors:0 dropped:0 overruns:0 frame:0
                    TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
                    collisions:0 txqueuelen:1000 RX bytes:392 (392.0 B)
                    TX bytes:392 (392.0 B)
mininet> exit
```

ID == MAC


```
$ sudo mn --mac
...
mininet> h1 ifconfig
h1-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:01
                    inet addr:10.0.0.1 Bcast:10.255.255.255
Mask:255.0.0.0
                    UP BROADCAST RUNNING MULTICAST MTU:1500
Metric:1
                    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                    TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                    collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B)
                    TX bytes:0 (0.0 B)
mininet> exit
```

XTerm

- Start xterm terminal for every host and switch on the network (option -x):
  - *$ sudo mn –x*
  - *mininet> xterm h1*

Running other types of switches

- *$ sudo mn --switch user --test iperf*
- *$ sudo mn --switch ovsk --test iperf*

Python Interpreter

- At the Mininet CLI, run:
  - *mininet> py 'hello ' + 'world'*
- Print the accessible local variables:
  - *mininet> py locals()*
- Next, see the methods and properties available for a node, using the dir() function:
  - *mininet> py dir(s1)*
- You can read the on-line documentation for methods available on a node by using the help() function:
  - *mininet> py help(h1) (Press "q" to quit reading the documentation.)*
- You can also evaluate methods of variables:
  - *mininet> py h1.IP()*

Link Up/Down

- *mininet> link s1 h1 down*
- *mininet> link s1 h1 up*

Connecting to a remote controller

- ***$ sudo mn --controller=remote, ip=[controller IP],port=[controller listening port]***

Например:

- ***$ sudo mn --controller=remote, ip=127.0.0.1,port=6653***

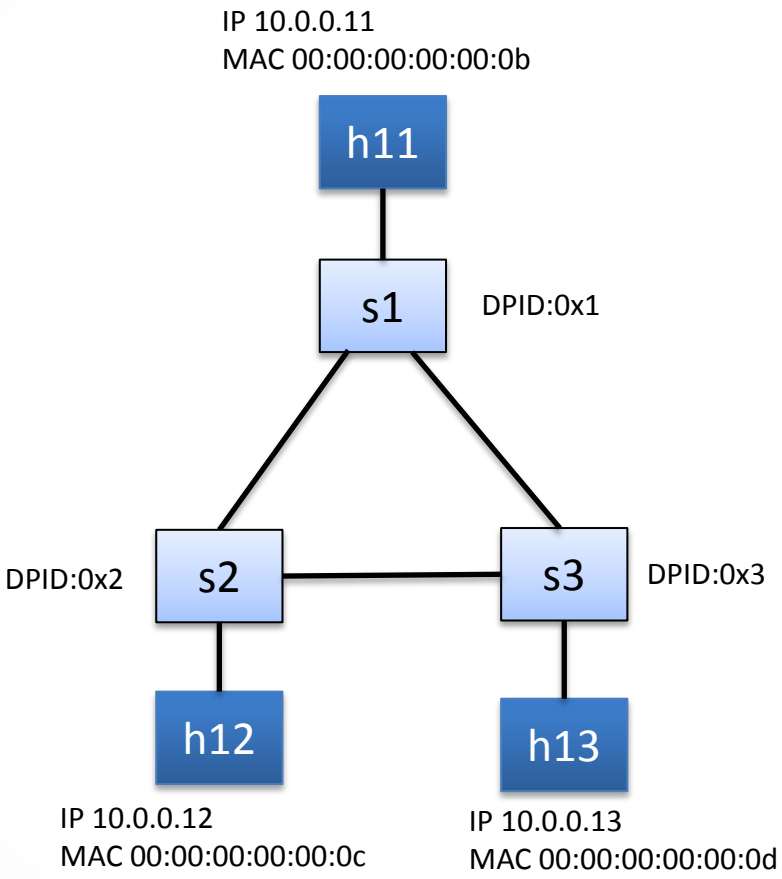
Skoltech
Skolkovo Institute of Science and Technology

Running an arbitrary topology


```
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
                leftHost = self.addHost( 'h1' )
                rightHost = self.addHost( 'h2' )
                leftSwitch = self.addSwitch( 's3' )
                rightSwitch = self.addSwitch( 's4' )
                # Add links
                self.addLink( leftHost, leftSwitch )
                self.addLink( leftSwitch, rightSwitch )
                self.addLink( rightSwitch, rightHost )
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo --test pingall

Create your topology



- Cd mininet/examples

- **sudo ./miniedit.py --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo**

2. dpctl

dpctl command

- Unix command for remote interaction with an OpenFlow switch using the OpenFlow protocol
  - They work only with switches with an open “listener”:
  - Stanford's reference switch, OpenVswitch, HP Procurve, Indigo
  - Commercial switches based on OpenVswitch
- Based on dpctl using the OpenFlow protocol, you can:
  - Show switch status
  - Show flow table (flow stats)
  - Set flows in flow table (flow_mod)
  - Send out packets (pkt_out)
- A useful tool for debugging

dpctl command

listener port

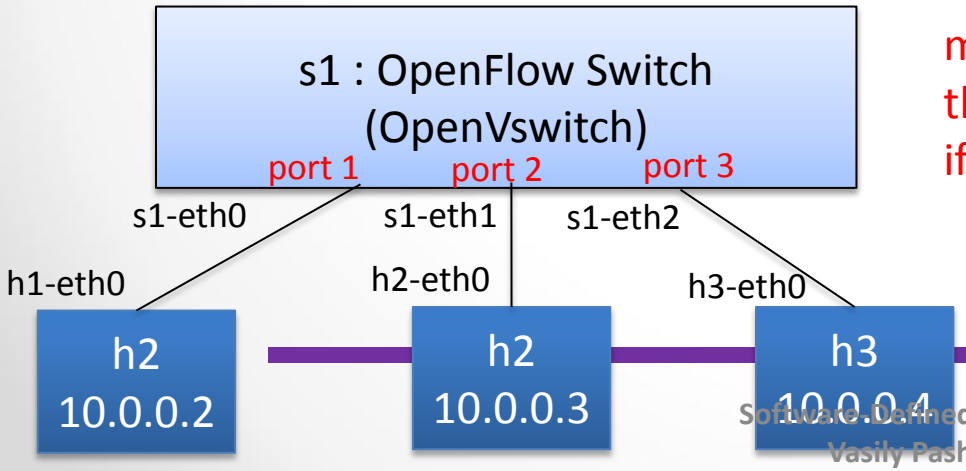
Show command

```
openflow@openflowtutorial:~$ dpctl show tcp:127.0.0.1:6634
features_reply (xid=0x850ce65c): ver:0x1, dpid:1
n_tables:2, n_buffers:256
features: capabilities:0x87, actions:0xfff
 1(s1-eth1): addr:9a:06:01:b7:03:ef, config: 0, state:0
     current:    10GB-FD COPPER
 2(s1-eth2): addr:76:bf:6c:25:2d:26, config: 0, state:0
     current:    10GB-FD COPPER
 3(s1-eth3): addr:16:5c:74:bf:fa:6f, config: 0, state:0
     current:    10GB-FD COPPER
 LOCAL(dp0): addr:00:23:20:c1:40:30, config: 0x1, state:0x1
get_config_reply (xid=0x20214c67): miss_send_len=0
```

Datapath ID of the switch

List of switch interfaces

miss_send_len := length of packet (from the beginning) to send to the controller if there's no flow table (0 = all packet)



dpctl command

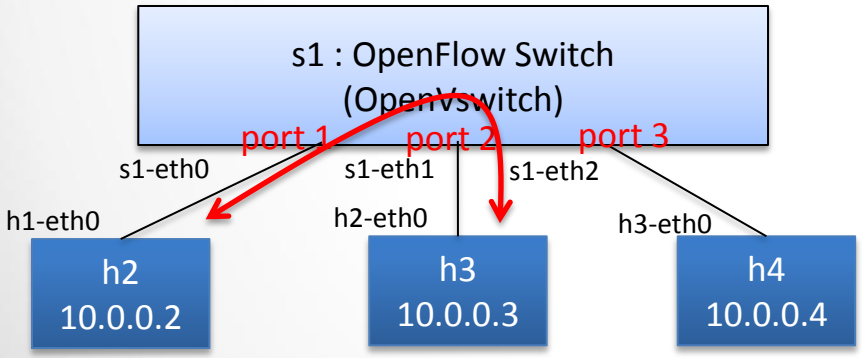
Dump-flow команда

```
openflow@openflowtutorial:~$ dpctl dump-flow tcp:127.0.0.1:6634
stats_reply (xid=0x93cfb59a): flags=none type=1(flow)
```

There are no rules for streams now

Setting up simple OpenFlow rules:

```
openflow@openflowtutorial:~$ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
openflow@openflowtutorial:~$ dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
```



Any packet arriving port 1 is forwarded to port 2
Any packet arriving port 2 is forwarded to port 1

dpctl command

We check that the rules have been established:

```
openflow@openflowtutorial:~$ dpctl dump-flow tcp:127.0.0.1:6634
stats_reply (xid=0xbe98f29b): flags=none type=1(flow)
  cookie=0, duration_sec=13s, duration_nsec=364000000s, table_id=0, priority=32768, n_packets=0, n_bytes=0,
idle_timeout=60,hard_timeout=0,in_port=1,actions=output:2
  cookie=0, duration_sec=5s, duration_nsec=734000000s, table_id=0, priority=32768, n_packets=0, n_bytes=0,
idle_timeout=60,hard_timeout=0,in_port=2,actions=output:1
```

(idle_timeout is set to 60 sec by default)

Ping check

```
mininet> h2 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_req=1 ttl=64 time=0.044 ms
64 bytes from 10.0.0.3: icmp_req=2 ttl=64 time=0.091 ms
64 bytes from 10.0.0.3: icmp_req=3 ttl=64 time=0.091 ms
64 bytes from 10.0.0.3: icmp_req=4 ttl=64 time=0.092 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3015ms
rtt min/avg/max/mdev = 0.044/0.079/0.092/0.022 ms
```

dpctl command

Examples of adding rules

- Flood arp packet from port 1 with mac src = 00:00:00:00:00:03 w/ idle_timeout 120 sec

```
openflow@openflowtutorial:~$ dpctl add-flow tcp:127.0.0.1:6634
    in_port=1,idle_timeout=120,dl_src=00:00:00:00:00:03,dl_type=0x806,actions=flood
```

- Specifying MAC src & dst and IP src & dst addresses

```
openflow@openflowtutorial:~$ dpctl add-flow tcp:127.0.0.1:6634
    in_port=2,idle_timeout=120,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,dl_type=0x800,nw_s
    rc=10.0.0.3,nw_dst=10.0.0.2,actions=output:1
```

- Specifying TCP flow

```
openflow@openflowtutorial:~$ dpctl add-flow tcp:127.0.0.1:6634
    in_port=1,idle_timeout=120,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,dl_type=0x800,nw_sr
    c=10.0.0.2,nw_dst=10.0.0.3,nw_proto=6,tp_dst=80,actions=output:2
```

Skoltech
Skolkovo Institute of Science and Technology

dpctl command

Removing Flow Rules

- Removing all rules

```
openflow@openflowtutorial:~$ dpctl del-flows tcp:127.0.0.1:6634
```

- Removing all rules that have in_port = 1

```
openflow@openflowtutorial:~$ dpctl del-flows tcp:127.0.0.1:6634 in_port=1
```

- Removing all rules that match on port in_port = 1

```
openflow@openflowtutorial:~$ dpctl --strict del-flows tcp:127.0.0.1:6634 in_port=1
```

Dpctl help

```
openflow@openflowtutorial:~$    man dpctl
and/or
openflow@openflowtutorial:~$    dpctl -h
```

3. OpenVSwitch

Topology launch

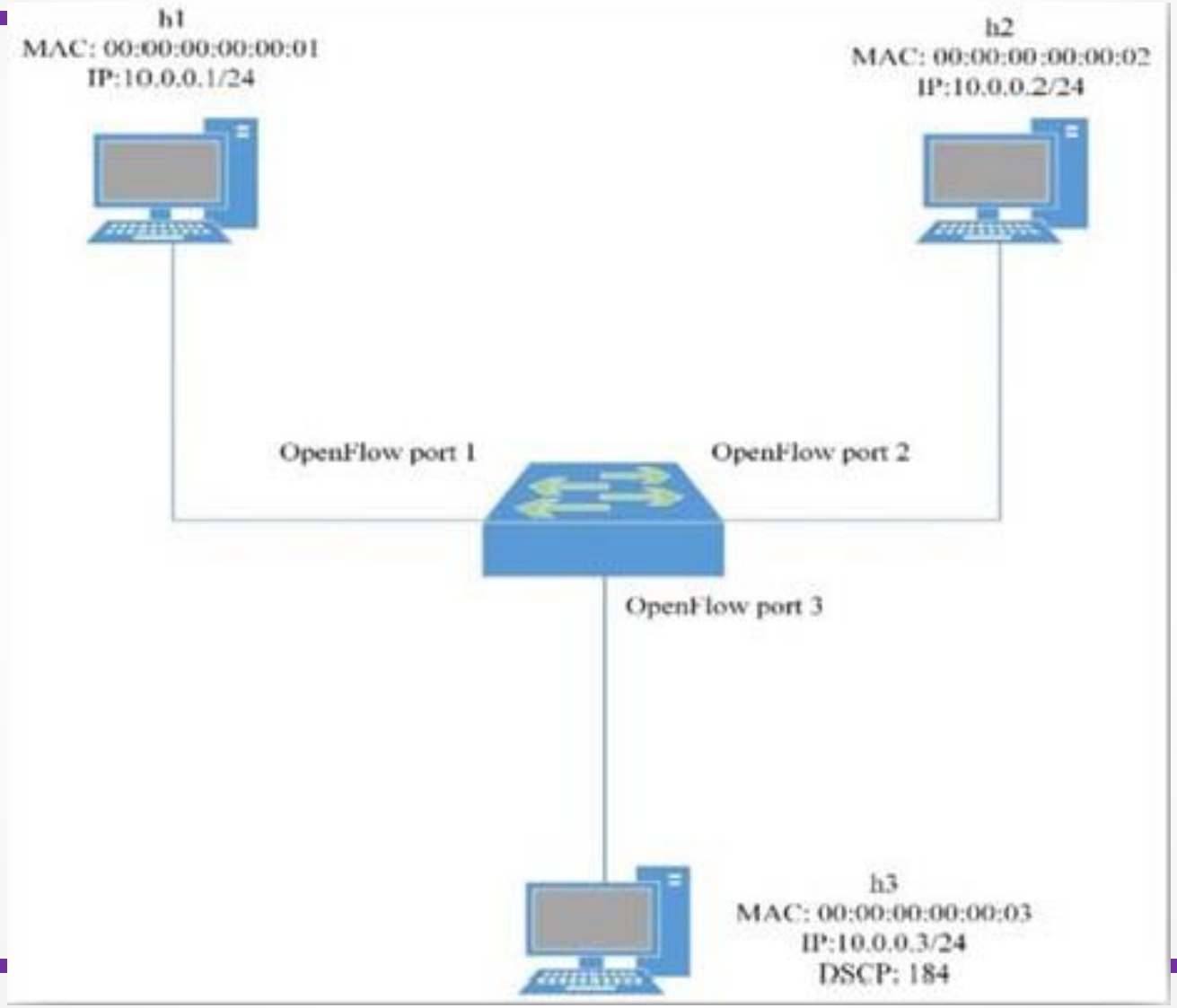
sudo mn --topo=single,3 --controller=none –mac

mininet> dump

mininet> net

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5091>
<Host h2: h2-eth0:10.0.0.2 pid=5093>
<Host h3: h3-eth0:10.0.0.3 pid=5095>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
```


h1
MAC: 00:00:00:00:00:01
IP:10.0.0.1/24
h2
MAC: 00:00:00:00:00:02
IP:10.0.0.2/24
OpenFlow port 1
OpenFlow port 2
OpenFlow port 3
h3
MAC: 00:00:00:00:00:03
IP:10.0.0.3/24
DSCP: 184

Show

mininet> sh ovs-ofctl show s1

```
mininet> sh ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS AR
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan m
rc mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(s1-eth1): addr:6e:a6:70:f0:32:31
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:e2:3d:89:b0:99:92
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:22:85:b1:3c:2f:3c
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:7e:fd:5b:4e:b7:4d
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
mininet>
```

Rule installation

mininet> Pingall

mininet> sh ovs-ofctl add-flow s1 action= normal

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X ^C
Interrupt
stopping h1
mininet> sh ovs-ofctl add-flow s1 action=normal
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

- **mininet>** sh ovs-ofctl dump-flows s1

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=355.983s, table=0, n_packets=24, n_bytes=1680, idle_a
MAL
mininet>
```

- **mininet>** sh ovs-ofctl del-flows s1

Skoltech
Skolkovo Institute of Science and Technology

Adding rules

- mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,action=output:2
- mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,action=output:1

```
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,actions=output:2
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,actions=output:1
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.15 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.033 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2221ms
rtt min/avg/max/mdev = 0.033/0.750/2.154/0.992 ms
mininet> h3 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2035ms

mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=76.800s, table=0, n_packets=5, n_bytes=378, idle_age=58, priorit
_port=1 actions=output:2
 cookie=0x0, duration=66.386s, table=0, n_packets=5, n_bytes=378, idle_age=58, priorit
_port=2 actions=output:1
mininet>
```

Priority rules

mininet> sh ovs-ofctl add-flow s1 priority=32768, action=drop
mininet> h1 ping –c3 h2
mininet> sh ovs-ofctl dump-flows s1

```
mininet> sh ovs-ofctl add-flow s1 priority=32768,actions=drop
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2030ms

mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=20.085s, table=0, n_packets=6, n_bytes=420, idle_age=7, actions=drop
 cookie=0x0, duration=1100.334s, table=0, n_packets=5, n_bytes=378, idle_age=1082, priority=500,in_port=1 actions=output
:2
 cookie=0x0, duration=1089.920s, table=0, n_packets=5, n_bytes=378, idle_age=1082, priority=500,in_port=2 actions=output
:1
mininet> sh ovs-ofctl del-flows --strict
ovs-ofctl: 'del-flows' command requires at least 1 arguments
mininet> sh ovs-ofctl del-flows s1 --strict
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1151.669s, table=0, n_packets=5, n_bytes=378, idle_age=1133, priority=500,in_port=1 actions=output
:2
 cookie=0x0, duration=1141.255s, table=0, n_packets=5, n_bytes=378, idle_age=1133, priority=500,in_port=2 actions=output
:1
mininet>
```

Layer2 matching

mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02, action=output:2

mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02, action=output:2

=output:1

mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood

mininet> pingall

```
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,actions=ou
tput:2
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,actions=ou
tput:1
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 X
h3 -> X X
*** Results: 66% dropped (2/6 received)
mininet>
```

Layer 3 matching

```
mininet> sh ovs-ctl add-flow s1 priority=500,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal
/bin/sh: 1: ovs-ctl: not found
mininet> sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=norm
al
mininet> sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.3,actions=mod_nw_tos:184,normal
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.1,actions=output:1
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.2,actions=output:2
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.3,actions=output:3
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=48.844s, table=0, n_packets=4, n_bytes=168, idle_age=13, arp,arp_tpa=10.0.0.1
actions=output:1
 cookie=0x0, duration=41.844s, table=0, n_packets=4, n_bytes=168, idle_age=13, arp,arp_tpa=10.0.0.2
actions=output:2
 cookie=0x0, duration=23.328s, table=0, n_packets=4, n_bytes=168, idle_age=13, arp,arp_tpa=10.0.0.3
actions=output:3
 cookie=0x0, duration=123.928s, table=0, n_packets=12, n_bytes=1176, idle_age=18, priority=500,ip,nw
_src=10.0.0.0/24,nw_dst=10.0.0.0/24 actions=NORMAL
 cookie=0x0, duration=87.500s, table=0, n_packets=0, n_bytes=0, idle_age=87, priority=500,ip,nw_src=
10.0.0.3 actions=mod_nw_tos:184,NORMAL
mininet>
```

Skoltech
Skolkovo Institute of Science and Technology

Thanks for your attention!

Vasily Pashkov
pashkov@lvk.cs.msu.su