Report submission

Introduction to Software Defined Networking

Winter Semester 2014/15

Seshagiri Prabhu Narasimha

Matriculation no: 20410690

Applied Computer Science

Department of Informatics

University of Goettingen

Germany

Table of Contents

1 Exercise 4 (Python SDN Simulator)

1.1 Functions (20P)

In a file **functions.py**:

– (5P) Define a function that accepts two strings as input, concatenates them and then prints out the result.
– **Solution:** Main source file *functions.py* is found under directory *Exercise*−4.

```
def two_string_input_output_concatinate():
    """
    A function which takes two string inputs
    gives a concatinated output string
    """
    string1 = raw_input("Enter the first string: ")
    string2 = raw_input("Enter the second string: ")
    print string1 + string2
```

– (5P) Define a function that accepts an integer number as input and prints out whether the number is even or odd.
– **Solution:**

```
def odd_or_even():
    """
    A function which takes integer input and
    computes whether its odd or even
    """
    input = int(raw_input("Enter the integer: "))
    if input%2 == 0:
        print "Given integer is even"
    elif input%2 == 1:
        print "Given integer is odd"
    else:
        print "Given input is not an integer"
```

– (10P) Define a function to compute a division by zero and use try/except to catch the exceptions
– **Solution:**

```
def division_by_zero():
    """
    A function to test try-catch during
    a division_by_zero
    """
    try:
        x = 10/0
    except ZeroDivisionError:
        print "Oops! Division by zero found"
```

Output:

```
$ python functions.py
Enter the first string: as
Enter the second string: as
asas
Enter the integer: 10
Given integer is even
Oops! Division by zero found
```

1.2 Classes and their methods (50P)

1. (10P) In a file persons.py: Define a class Person and its two child classes: Male and Female. All classes have a method $get_gender()$ which should return their respective gender
Solution: The source file *person.py* could be found under $Exercise - 4$ directory.

```
#! /usr/bin/env python2
"""
person.py: A program to create a parent class and two child classes of the same
"""
__author__      = "Seshagiri Prabhu"
__copyright__   = "MIT License"


class Person():
    """
    This is the main parent class for a person
    """
    gender = "Null"
    def get_gender(self):
        return gender


class Male(Person):
    """
    This is a child class of person belongs to all males
    """
    gender = "Male"
    def get_gender(self):
        return gender


class Female(Person):
    """
    This is a child class of person belongs to all females
    """
    gender = "Female"
    def get_gender(self):
        return gender


if __name__=="__main__":
    """
    Main function
    """
```

Exercise–4/person.py

2. (40P) In cryptography, a Caesar cipher is a very simple encryption technique in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals. ROT-13 ("rotate by 13 places") is a widely used example of a Caesar cipher where the shift is 13. In a file rot13.py: Implement an Class to encode and decode messages with ROT-13, and use it to decipher the message: Vagebqhpvat FQAf Note: In Python, the key for ROT-13 may be represented by means of the following dictionary:

```
key = {'a':'n', 'b':'o', 'c':'p', 'd':'q', 'e':'r', 'f':'s', 'g':'t', 'h':'u',
 'i':'v', 'j':'w', 'k':'x', 'l':'y', 'm':'z', 'n':'a', 'o':'b', 'p':'c',
 'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k',
 'y':'l', 'z':'m', 'A':'N', 'B':'O', 'C':'P', 'D':'Q', 'E':'R', 'F':'S',
 'G':'T', 'H':'U', 'I':'V', 'J':'W', 'K':'X', 'L':'Y', 'M':'Z', 'N':'A',
 'O':'B', 'P':'C', 'Q':'D', 'R':'E', 'S':'F', 'T':'G', 'U':'H', 'V':'I',
 'W':'J', 'X':'K', 'Y':'L', 'Z':'M'}
```

– **Solution:** Main source file *rot13.py* could be found under directory *Exercise* – 4.

```
#! /usr/bin/env python2
"""
rot13.py: Ceaser cipher
"""
__author__      = "Seshagiri Prabhu"
__copyright__   = "MIT License"

```

```
key_map = {'a':'n', 'b':'o', 'c':'p', 'd':'q', 'e':'r', 'f':'s', 'g':'t', 'h':'u',
        'i':'v', 'j':'w', 'k':'x', 'l':'y', 'm':'z', 'n':'a', 'o':'b', 'p':'c',
        'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k',
        'y':'l', 'z':'m', 'A':'N', 'B':'O', 'C':'P', 'D':'Q', 'E':'R', 'F':'S',
        'G':'T', 'H':'U', 'I':'V', 'J':'W', 'K':'X', 'L':'Y', 'M':'Z', 'N':'A',
        'O':'B', 'P':'C', 'Q':'D', 'R':'E', 'S':'F', 'T':'G', 'U':'H', 'V':'I',
        'W':'J', 'X':'K', 'Y':'L', 'Z':'M', ' ': ' '}


class Ceaser:
    """
    This is the main parent class of Ceaser cipher
    """
    def __init__(self, input_string):
        self.string = input_string


    def encode(self):
        """
        A function to encode string
        """
        out_string = ''
        for x in self.string:
            out_string += key_map[x]
        return out_string


    def decode(self):
        """
        A function to decode string
        """
        out_string = ''
        for x in self.string:
            for key,value in key_map.iteritems():
                if x == value:
                    out_string += key
        return out_string


if __name__=="__main__":
    """
    Main function
    """
    input_string_encode = raw_input("Enter the string you
   want to encode: ")
    encode = Ceaser(input_string_encode)
    print "Encoded string: " + encode.encode()

    input_string_decode = raw_input("Enter the string you
   want to decode: ")
    decode = Ceaser(input_string_decode)
    print "Decoded string: " + decode.decode()
```

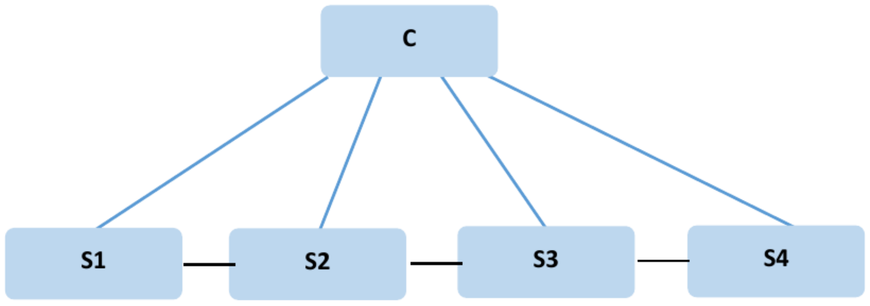
Exercise–4/rot13.py

Output:

```
$ python rot13.py
Enter the string you want to encode: Introducing SDNs
Encoded string: Vagebqhpvat FQAf
Enter the string you want to decode: Vagebqhpvat FQAf
Decoded string: Introducing SDNs
```

1.3 A basic Software-defined Network simulator (100P)

1. (100P) Now that you know all the basics of Python, lets get back to the course content! In this exercise, you will build your very own (VERY basic) SDN simulator. At this point in the lecture you have learned that SDNs basically consist of two elements: switches and controllers. Suppose you have the following topology:



We call this topology, in which the four switches (S1-S4) are connected in a single line, a linear topology. The switches are orchestrated by a single controller C. Your task is to rebuild this topology and simulate the exchange of packets over that topology in a Python simulator. Consider the lifetime of the very first packet in the network that will be send from, for instance, S1 to S4:

The packet will be created at S1, and since it is the first packet, S1 will not have any flow tables installed. Instead, it will have to ask the controller C for a new rule. C has to install a new rule in S1. The rule should be: Forward to S2. S2 has to ask C for a new rule as well. Note that in this case, the rule should be If received from S1, forward to S3. Ultimately, the packet will arrive at S4, which should know that it is the destination of the packet.

Try to figure out how to build such a simulator in Python yourself first. After that, test your system with two flows:

– one flow of five packets originating at S1 and with S4 as destination
– a second flow consisting of three packets from S4 to S1
– you do not need to acknowledge the packets in this basic simulator

Output: The source file *sdn_simulator.py* could be found under *Exercise*−4 directory.

```
$ python sdn_simulator.py

Welcome to SDN Simulator
NAME
         SDN SIMULATOR - an interpreter, interactive SDN
    simulator

OPTIONS
         <SWITCH-A> ping <SWITCH-B>
          -- Send an ICMP like request between two switches

         dump
          -- Dump all the flow tables of the switches

         exit
          -- Exit the simulator

         help
          -- Display the help text

SDN Simulator > Enter the number of switches: 4
Creating Switches  s0...  s1...  s2...  s3...
SDN Simulator > dump
++++++++++++++++++
+Switch   +     s0+
+Left     +    NIL+
+Right    +    NIL+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s1+
+Left     +    NIL+
+Right    +    NIL+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s2+
+Left     +    NIL+
+Right    +    NIL+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s3+
+Left     +    NIL+
+Right    +    NIL+
++++++++++++++++++

SDN Simulator > s0 ping s3
Packet Source: s0 Packet Desination: s3
s0 -> s1 -> s2 -> s3 (Packet delivered)

SDN Simulator > dump
++++++++++++++++++
+Switch   +     s0+
+Left     +    NIL+
+Right    +     s1+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s1+
+Left     +     s0+
+Right    +     s2+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s2+
+Left     +     s1+
+Right    +     s3+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s3+
+Left     +     s2+
+Right    +    NIL+
++++++++++++++++++

SDN Simulator > s3 ping s0
Packet Source: s3 Packet Desination: s0
s3 -> s2 -> s1 -> s0 (Packet delivered)

SDN Simulator > dump
++++++++++++++++++
+Switch   +     s0+
+Left     +    NIL+
+Right    +     s1+
++++++++++++++++++
++++++++++++++++++
+Switch   +     s1+
+Left     +     s0+
+Right     +      s2+
+++++++++++++++++++
+++++++++++++++++++
+Switch    +      s2+
+Left      +      s1+
+Right     +      s3+
+++++++++++++++++++
+++++++++++++++++++
+Switch    +      s3+
+Left      +      s2+
+Right     +     NIL+
+++++++++++++++++++

SDN Simulator > exit
```

2 Exercise 7: MN Python API & POX

2.1 The python API and Technologies (50p)

Please explain line by line what happens in the following code:

```
#! /usr/bin/env python2
__author__      = "Seshagiri Prabhu"
__copyright__   = "MIT License"

def runExp():
    """
    A function to run a mininet experiment
    """
    """
    Creates an object of customTopo class with argument n=2,
    used to
    create link between switches and hosts
    """
    topo = customTopo( n=2 )
    """
    Creates a Mininet with with given custom topology. Network
    emulation with hosts spawned in network namespaces.
    Mininet is
    the main class to create and manage network
    """
    net = Mininet ( topo=topo )
    """
    Start controller, switches and network
    """
    net.start()
    """
    The mininet object is passed to simple command line
    interface
    to talk to nodes
    """
    CLI( net )
    """
    Stops the mininet network
    """
    net.stop()


class customTopo( Topo ):
    """
    A class to create custom network topology
    """
    def __init__( self , n, **kwargs ):
        """
        Constructor of the class customTopo
        @args: self , n, kwargs
        """
        """
        Calling the base class for mininet topology
        """
        Topo.__init__( self , **kwargs )
        """
        Add two hosts to the topology and returns the host
    name to the respective objects
        """
        h1, h2 = self.addHost( 'h1' ), self.addHost( 'h2' )
        """
        Adds a switch to the topology and returns the switch
    name to the variable
        """
        s1 = self.addSwitch( 's1' )
        for _ in range( n ):
            """
            Adds bidirectional link between the elements in
    the network
            """
            self.addLink( s1 , h1 )
            self.addLink( s1 , h2 )


if __name__ == '__main__':
    """
    Main function
    """
    """
    To set Mininet s default output level. Informs mininet
    to print
    useful information
    """
    setLogLevel( 'info' )
    """
    Calls the function to run the experiment
    """
    runExp()
```

Exercise–7/question1.py

2.2 The POX controller (100P)

One of the main features of Mininet is that you can easily port your emulated networks to real-world production networks. To do so, you will need to define a mapping of your production network to an emulated Mininet. This also includes the controllers you use.

In this exercise you will have a look at the POX controller. POX is included in the VM image we are using for this course. What we want to do first, is to bring up a very simple controller implementation that acts like a standard hub.

– (Optional) You will read the solution on the next page, but think about which parameters you would have to call mn with to generate a single switch topology with three hosts that assigns nodes with simplified MAC addresses, runs the Open vSwitch implementation on the switch and will link with an external controller. For that, we bring up a new mininet (dont for get to clean up before you do this):

```
$ sudo mn --topo single,3 --mac --switch ovsk --controller
   remote
```

Now, we will bring up the POX controller in a separate terminal:

```
$ ./pox/pox.py log.level --DEBUG misc.of_tutorial
```

– (10P) What do you observe after connecting the controller?
Solution: After the POX controller is enabled, the mininet automatically detects the presence of external controller and able to accept it as the controller for its setup. Hence the hosts are able to ping between each other.

We will now check if our hub is working correctly.

– (20P) Open one xterm window for each host and run tcpdump on hosts 1 and 2:
Solution: Yes, the switch acts as a hub in this case and it broadcasts what ever packet it receives to other devices. Hence every devices connected to hub receives also receives the packet received by the hub.

```
$ tcpdump –XX –n –i <interface>
```

Then, in the xterm of host 3, try to ping the IP address of h1 or h2. Also try to ping a device that is not reachable. Please explain what you observe (Hint: if you recall your basic lecture on computer networks, you should know that a hub is generally a dumb device.)

– (Optional) Recap how a learning switch is operating.
– (70P) Open up the file *pox/pox/misc/of*$_t$*utorial.py* in the editor of your choice. In the Python code you will see that we are currently operating our controller with the help of the $act_like_hub()$ method. Your task in this exercise is to

- (10P) modify the controller to use $act_like_switch()$ instead.
- (40P) implement $act_like_switch()$ so that your controller actually acts like a switch. There are some helpful hints in the code.
- (20P) It is somewhat inefficient to let the controller decide the fate of every single packet. Implement an $act_like_flow_switch()$ method in which your controller instead installs flow-rules into the switch for all packets that are initiually flow-table misses. This will improve the performance of the forwarding on subsequent packets. You can use the POX API and the POX documentation regarding OpenFlow here (hint: have a very close look at ofp_flow_mod).

Solution: Source file *of_tutorial.py* could be found inside $Exercise - 7$ directory.

3 Exercise 8: Mininet & FlowVisor

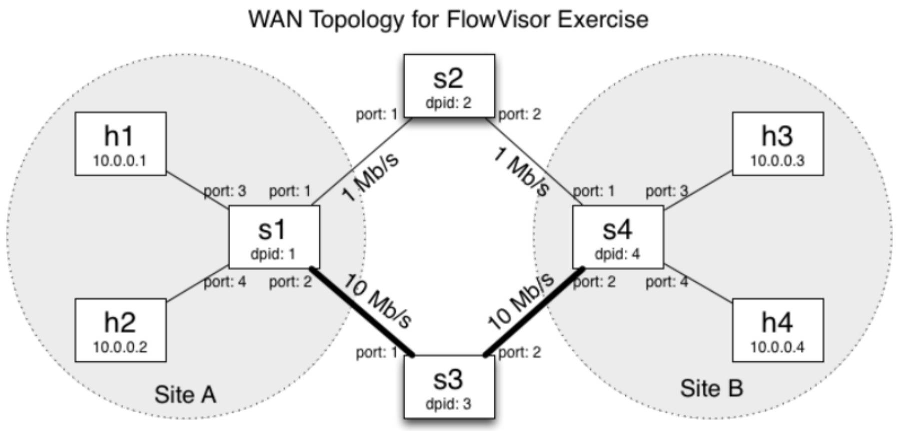
3.1 Install FlowVisor (0P)

(0P) To install FlowVisor, please first download it here. Now, transfer the downloaded .deb file to your VM (e.g., using scp or WinSCP), login to your VM and install the required packages for FlowVisor:

```
$ sudo apt−get −−install PATH/TO/DEB/FILE
```

3.2 Create your own FlowVisor topology (50P)

(50P) Using the Mininet Python API, create the FlowVisor WAN topology (which you may know from earlier exercises) in a file $mini - fw - topo.py$:



After you have defined it start your topology

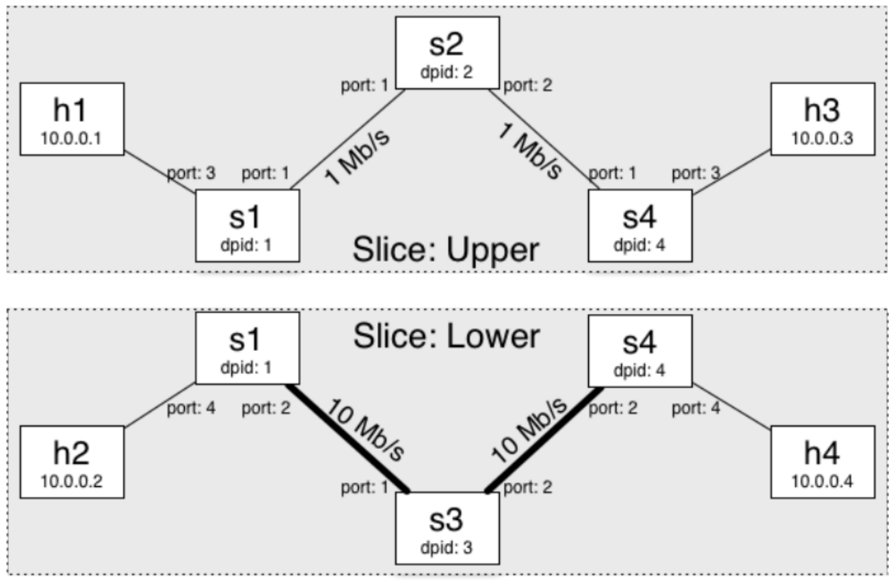
```
$ sudo mn --custom
```

Solution:

```
$ sudo mn --custom mini-fw-topo.py --topo fvtopo --link tc
```

3.3 Slice the Network (100p)

Now, slice your network so that it supports the following slices:



In

short, this slice arrangement allows traffic to be sent from h1 to h3 and h2 to h4 (and vice-versa) only, even though the topology itself (i.e., without slicing) would allow sending traffic between arbitrary pairs of hosts.

For slicing a network with FlowVisor in general, you need to take the following steps. First, make sure you set up the flowvisor package correctly. To configure FlowVisor, use:

```
$ sudo -u flowvisor fvconfig generate /etc/flowvisor/config.
    json
```

Then, start flowvisor in a new terminal:

```
$ sudo /etc/init.d/flowvisor start
```

We have to enable topology control for flowvisor as well:

```
$ fvctl -f /dev/null set-config --enable-topo-ctrl
```

Similar to ovs-ofctl, fvctl is the control channel that we will use for flowvisor. The option f refers to the flowvisor password file. Since we have set the password to be empty, we can hand it /dev/null. This part will be present in all the following fvctl calls. Restart flowvisor:

Now, have a look at FlowVisor configuration

```
$ fvctl -f /dev/null get-config
```

This also has the purpose of making sure that flowvisor is actually running and that all the switches have indeed a connection to flowvisor. The configuration should show this.

– (5P) Which part of the configuration file tells you that all four switches have connected to flowvisor?
Solution:

```
"flowmod-limit": {
    "fvadmin": {
  "00:00:00:00:00:00:00:01": -1,
  "00:00:00:00:00:00:00:02": -1,
  "00:00:00:00:00:00:00:03": -1,
  "00:00:00:00:00:00:00:04": -1,
        "any": null
    }
},
```

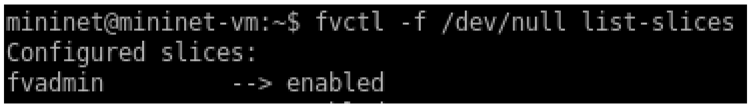
In the lecture, you also got a brief overview over the major flowvisor commands. Now, make use of these commands to

– (5P) List the currently existing slices.
Solution:

```
mininet@mininet—vm:~$ fvctl —f /dev/null list—slices
Configured slices: fvadmin —> enabled
```

– (5P) List the currently existing flowspaces.
Solution:

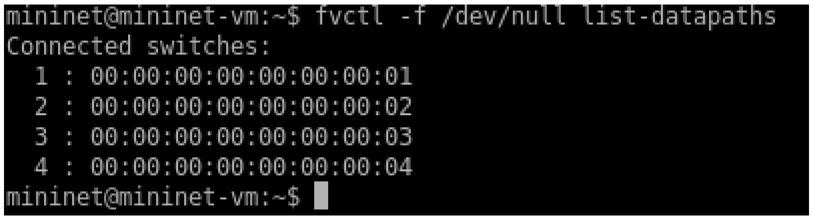
```
mininet@mininet—vm:~$ fvctl —f /dev/null list—flowspace
Configured Flow entries:
  None
```



```
mininet@mininet-vm:~$ fvctl -f /dev/null list-slices
Configured slices:
fvadmin          --> enabled
```

– (5P) List the currently connected switches.
Solution:

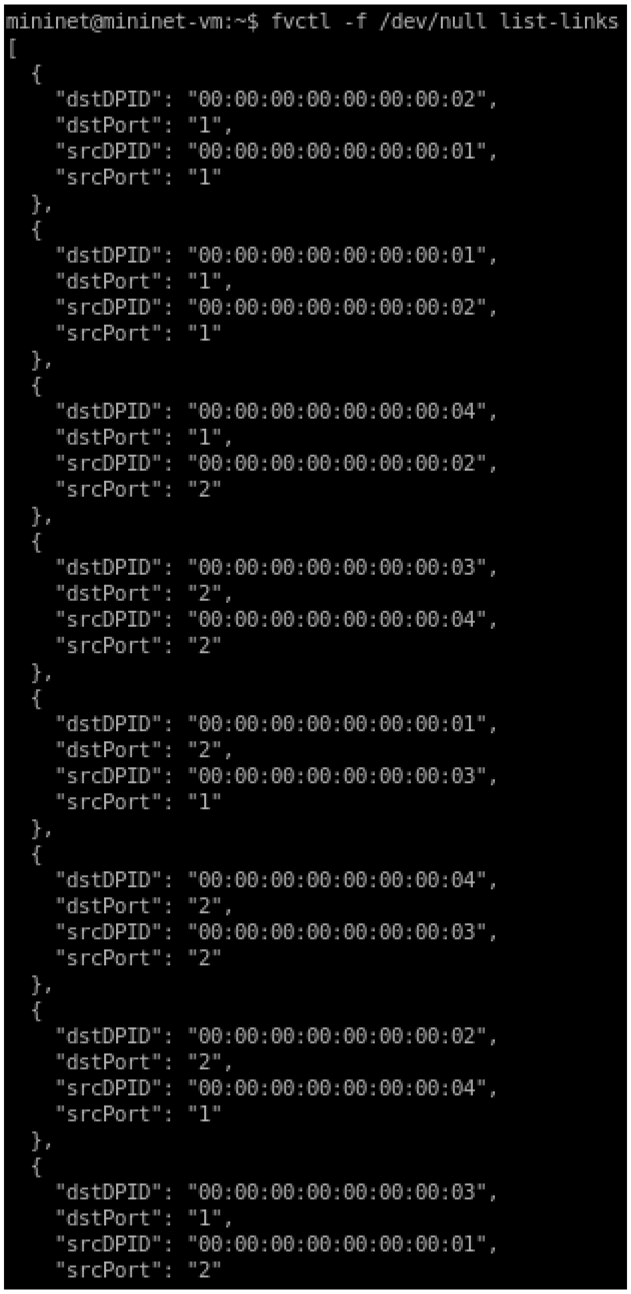
```
mininet@mininet—vm:~$ fvctl —f /dev/null list—datapaths
Connected switches:
  1 : 00:00:00:00:00:00:00:01
  2 : 00:00:00:00:00:00:00:02
  3 : 00:00:00:00:00:00:00:03
  4 : 00:00:00:00:00:00:00:04
```



```
mininet@mininet-vm:~$ fvctl -f /dev/null list-datapaths
Connected switches:
  1 : 00:00:00:00:00:00:00:01
  2 : 00:00:00:00:00:00:00:02
  3 : 00:00:00:00:00:00:00:03
  4 : 00:00:00:00:00:00:00:04
mininet@mininet-vm:~$
```

– (5P) List the currently existing links.
Solution:

```
mininet@mininet—vm:~$ fvctl —f /dev/null list—links
[
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "1"
  },
    {
        "dstDPID": "00:00:00:00:00:00:00:01",
        "dstPort": "1",
        "srcDPID": "00:00:00:00:00:00:00:02",
        "srcPort": "1"
    },
    {
        "dstDPID": "00:00:00:00:00:00:00:04",
        "dstPort": "1",
        "srcDPID": "00:00:00:00:00:00:00:02",
        "srcPort": "2"
    },
    {
        "dstDPID": "00:00:00:00:00:00:00:03",
        "dstPort": "2",
        "srcDPID": "00:00:00:00:00:00:00:04",
        "srcPort": "2"
    },
    {
        "dstDPID": "00:00:00:00:00:00:00:01",
        "dstPort": "2",
        "srcDPID": "00:00:00:00:00:00:00:03",
        "srcPort": "1"
    },
    {
        "dstDPID": "00:00:00:00:00:00:00:04",
        "dstPort": "2",
        "srcDPID": "00:00:00:00:00:00:00:03",
        "srcPort": "2"
    },
    {
        "dstDPID": "00:00:00:00:00:00:00:02",
        "dstPort": "2",
        "srcDPID": "00:00:00:00:00:00:00:04",
        "srcPort": "1"
    },
    {
        "dstDPID": "00:00:00:00:00:00:00:03",
        "dstPort": "1",
        "srcDPID": "00:00:00:00:00:00:00:01",
        "srcPort": "2"
    }
]
```



```
mininet@mininet-vm:~$ fvctl -f /dev/null list-links
[
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:01",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:02",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:04",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:02",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:03",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:04",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:01",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:03",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:04",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:03",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "2",
    "srcDPID": "00:00:00:00:00:00:00:04",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:03",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "2"
```

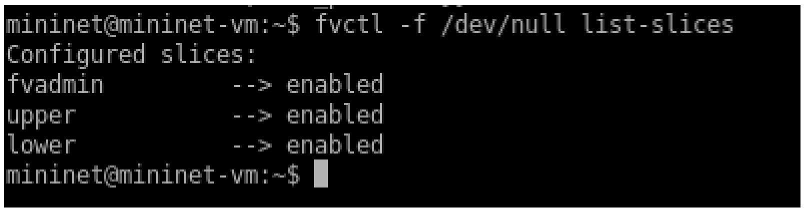
Afterwards, proceed with slicing your topology:

– (10P) Create the appropriate slices.

Solution:

```
mininet@mininet-vm:~$ fvctl -f /dev/null add-slice upper
    tcp:localhost:10001 admin@upperslice
```

```
mininet@mininet-vm:~$ fvctl -f /dev/null add-slice lower
    tcp:localhost:10002 admin@lowerslice
mininet@mininet-vm:~$ fvctl -f /dev/null list-slices
Configured slices:
fvadmin          --> enabled
upper            --> enabled
lower            --> enabled
```

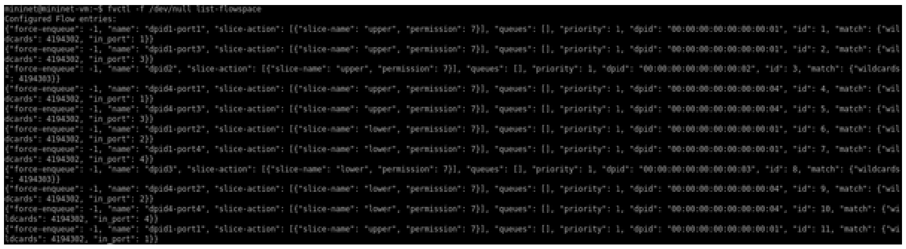


```
mininet@mininet-vm:~$ fvctl -f /dev/null list-slices
Configured slices:
fvadmin          --> enabled
upper            --> enabled
lower            --> enabled
mininet@mininet-vm:~$
```

– (40P) Create the appropriate flowspaces.
Solution:

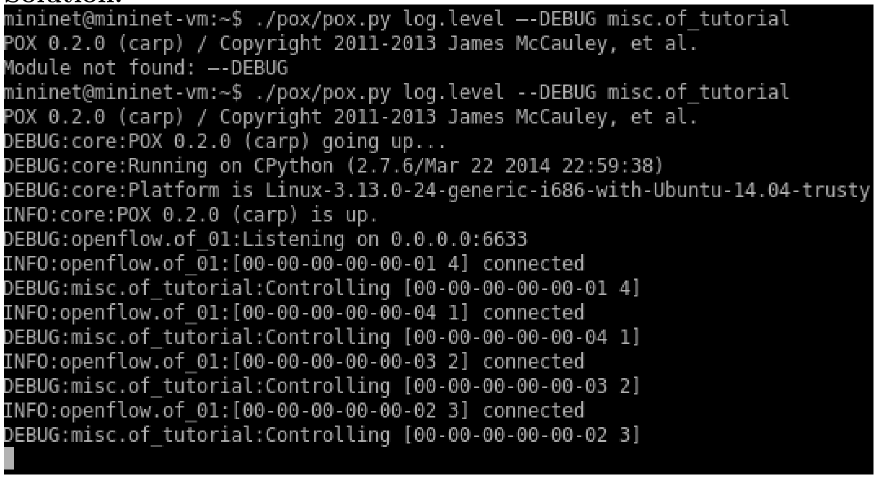
```
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid1-port3 1 1 in_port=3 upper=7
FlowSpace dpid1-port3 was added with request id 1.
mininet@mininet-vm:~$ ^C
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid2 2 1 any upper=7
FlowSpace dpid2 was added with request id 2.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid4-port1 4 1 in_port=1 upper=7
FlowSpace dpid4-port1 was added with request id 3.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid4-port3 4 1 in_port=3 upper=7
FlowSpace dpid4-port3 was added with request id 4.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid1-port2 1 1 in_port=2 lower=7
FlowSpace dpid1-port2 was added with request id 5.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid1-port4 1 1 in_port=4 lower=7
FlowSpace dpid1-port4 was added with request id 6.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid3 3 1 any lower=7
FlowSpace dpid3 was added with request id 7.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid4-port2 4 1 in_port=2 lower=7
FlowSpace dpid4-port2 was added with request id 8.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid4-port4 4 1 in_port=4 lower=7
FlowSpace dpid4-port4 was added with request id 9.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace
    dpid1-port1 1 1 in_port=1 upper=7
FlowSpace dpid1-port1 was added with request id 10.
```

```
mininet@mininet-vm:~$ fvctl -f /dev/null list-flowspace
Configured Flow entries:
{"force-enqueue": -1, "name": "dpid1-port1", "slice-action"
    : [{"slice-name": "upper", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01",
    "id": 1, "match": {"wildcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid1-port3", "slice-action"
    : [{"slice-name": "upper", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01",
    "id": 2, "match": {"wildcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid2", "slice-action": [{"
    slice-name": "upper", "permission": 7}], "queues": [],
    "priority": 1, "dpid": "00:00:00:00:00:00:00:02", "id":
     3, "match": {"wildcards": 4194303}}
{"force-enqueue": -1, "name": "dpid4-port1", "slice-action"
    : [{"slice-name": "upper", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04",
    "id": 4, "match": {"wildcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid4-port3", "slice-action"
    : [{"slice-name": "upper", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04",
    "id": 5, "match": {"wildcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid1-port2", "slice-action"
    : [{"slice-name": "lower", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01",
    "id": 6, "match": {"wildcards": 4194302, "in_port": 2}}
{"force-enqueue": -1, "name": "dpid1-port4", "slice-action"
    : [{"slice-name": "lower", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01",
    "id": 7, "match": {"wildcards": 4194302, "in_port": 4}}
{"force-enqueue": -1, "name": "dpid3", "slice-action": [{"
    slice-name": "lower", "permission": 7}], "queues": [],
    "priority": 1, "dpid": "00:00:00:00:00:00:00:03", "id":
     8, "match": {"wildcards": 4194303}}
{"force-enqueue": -1, "name": "dpid4-port2", "slice-action"
    : [{"slice-name": "lower", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04",
    "id": 9, "match": {"wildcards": 4194302, "in_port": 2}}
{"force-enqueue": -1, "name": "dpid4-port4", "slice-action"
    : [{"slice-name": "lower", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04",
    "id": 10, "match": {"wildcards": 4194302, "in_port":
    4}}
{"force-enqueue": -1, "name": "dpid1-port1", "slice-action"
    : [{"slice-name": "upper", "permission": 7}], "queues":
     [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01",
    "id": 11, "match": {"wildcards": 4194302, "in_port":
    1}}
```



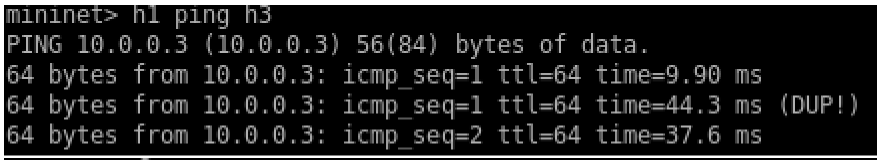
```
mininet@mininet-vm:~$ fvctl -f /dev/null list-flowspace
Configured Flow entries:
{"force-enqueue": -1, "name": "dpid1-port1", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 1, "match": {"wil
dcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid1-port3", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 2, "match": {"wil
dcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid2", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:02", "id": 3, "match": {"wildcards
": 4194303}}
{"force-enqueue": -1, "name": "dpid4-port1", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 4, "match": {"wil
dcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid4-port3", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 5, "match": {"wil
dcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid1-port2", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 6, "match": {"wil
dcards": 4194302, "in_port": 2}}
{"force-enqueue": -1, "name": "dpid1-port4", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 7, "match": {"wil
dcards": 4194302, "in_port": 4}}
{"force-enqueue": -1, "name": "dpid3", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:03", "id": 8, "match": {"wildcards
": 4194303}}
{"force-enqueue": -1, "name": "dpid4-port2", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 9, "match": {"wil
dcards": 4194302, "in_port": 2}}
{"force-enqueue": -1, "name": "dpid4-port4", "slice-action": [{"slice-name": "lower", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:04", "id": 10, "match": {"wi
ldcards": 4194302, "in_port": 4}}
{"force-enqueue": -1, "name": "dpid1-port1", "slice-action": [{"slice-name": "upper", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:01", "id": 11, "match": {"wi
ldcards": 4194302, "in_port": 1}}
```

– (10P) Connect an instance of the POX controller to each of your slices
Solution:

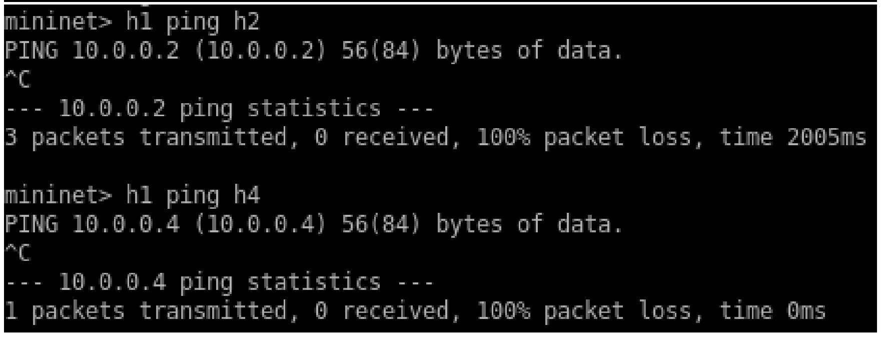


```
mininet@mininet-vm:~$ ./pox/pox.py log.level —-DEBUG misc.of_tutorial
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
Module not found: —-DEBUG
mininet@mininet-vm:~$ ./pox/pox.py log.level --DEBUG misc.of_tutorial
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Mar 22 2014 22:59:38)
DEBUG:core:Platform is Linux-3.13.0-24-generic-i686-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 4]
INFO:openflow.of_01:[00-00-00-00-00-04 1] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-04 1]
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-03 2]
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 3]
```

– (10P) In Mininet, verify that your slicing works properly, i.e., h1 can reach h3 but not h2 and h4, and h2 can reach h4, but not h1 and h3.
Solution:



```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=9.90 ms
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=44.3 ms (DUP!)
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=37.6 ms
```



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2005ms

mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
^C
--- 10.0.0.4 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

4 Paper Review

Paper Review by

Seshagiri Prabhu Narasimha

Paper Information

Paper Title: OpenVirteX: Make Your Virtual SDNs Programmable
Type of Paper

	Full paper (9 or more pages)	➻	Short paper (Less than 9 pages)

Summary of the Paper

OpenVirteX (OVX), is a network virtualization platform. It enables operators to create and maintain virtual Software Defined Networks (vSDN). Its the first framework which enables tenants to specify their own Network Operating System (NOS) and control it through a programmable API. OVX serves as an Infrastructure as a Service (IaaS) in the context of SDN. Some of the previous research works including FlowVisor, VeRTIGO and FlowN are the inspiration for the OVX. FlowVisor and FlowN provides shared address space among the tenant networks and hence any misconfiguration would lead to overlapping or security concerns. By introducing virtualization, OVX enables complete modularity between tenant networks and thus it supports any arbitrary topology and addressing schemes, configured as per tenant request. Network embedder tool is used to convey the tenant requests via API calls to OVX. A user could specify a virtual network's addressing scheme, topology and NOS link, this generates virtual to physical mapping. The internal OVX architecture does a loose decoupling of virtual elements from physical counterparts.

OVX enforces the tenant traffic isolation using the physical address fields of virtual components: Flow ID (looks up the particular flow that entered the virtual link and restore its values when the traffic exits the virtual link), Tenant ID (Isolates the different tenants on the virtual link) and Virtual link ID (keeps tracks of virtual links). Hence whenever a link layer packet arrives with a certain outport specified, OVX “knows” where exactly the other end of that link is.

The only limitation put by the OVX is while mapping the switches. No single physical switch cannot be partitioned. One virtual component can be mapped to one or more physical components. This increases the resilience of the network. As the virtual to physical mapping is logically centralized in the global map, these can be changed at runtime enabling tenants to do dynamic vSDN reconfigurations. Another minor feature is the ability to port the virtual networks under two conditions given that operator network support the same control protocol as OVX and the vSDN topology can be mapped onto the network without partitioning the physical switches.

The results shown by the OVX are remarkable when compared to previous related works. The control channel latency to create virtual networks through API has been reduced by half when compared to the existing solutions. The OVX was deployed and tested for a production scale on Internet 2, a nation wide research network in United States.

Contributions of the Paper

(What are the major issues addressed in the paper? Do you consider them important? Comment on the degree of novelty, creativity, impact, and technical depth in the paper)

[100-150 words]

The major issues addressed by this research work is inability to create customizable topology, isolated addressing schemes and control his/her virtual SDN with NOS of his/her choice. The importance of this research work is that it gives more freedom to the tenants to create on demand, re-configurable and customizable virtual networks with resilience. The security of the tenant network is quite high as there is complete network isolation.

Programmable API for tenants with customizable topology & full address space and resiliency of the topology network are some of the new techniques introduced by this paper.

Even though the core concepts of network virtualization, complete network isolation and resilience of tenant network are inspired from the standard OS virtualization, the impact of developing these are something great as other related works couldn’t succeed in achieving the same.

The paper does describe the atomic details of their implementation of architecture. It has mentioned very high level details about the internal and external OVX architecture.

Strengths of the Paper

(What are the main reasons to assess this paper as high-quality? e.g. innovative ideas/technologies, paper presentation, detailed performance evaluation.)
[at least 2-3 strengths, more are better]

1. The ideas put forward by the authors for the virtual networks gives a new dimension to the traditional networking.
2. The paper is presented in such a way that even a bachelor degree computer science student with little or no networking knowledge can understand the concept.

Weaknesses of the Paper

(What are the most important reasons to assess this paper as low-quality? e.g. writing skill, technical errors, unrealistic assumptions, unanswered questions, limited measurements or evaluation)
[at least one weakness, more are better]

1. Runtime load has not be measured and included in the results. The paper only describes the latency of the control channel during the creation of virtual networks.
2. The paper mentions at several places that physical components cannot be partitioned and keeps it is as a strong assumption. However, they do not mention the reason of keeping such a assumption/rule.

Open Issues and Future Work

(What are the options for future work that you think are important? If you are asked to work on the problem investigated in this paper, what will you do differently?)
[try to find something]

1. Snapshotting of the running VM is not enabled in the current version of OVX, once enabled, then the system would allow the tenants to port their virtual topology to work under different network operator.
2. The current system uses `openflowj` library which is strictly tied with `OpenFlow v1.0`. Once upgraded to v1.3 with `LOXI`, a marshalling and unmarshalling engine that generates it generates version-agonistic `OF` libraries in multiple languages in the southbound interface, thus allowing common open source `OFv1.0` controllers like NOX and Floodlight to run on an `OFv1.3` network.
3. OVX with `OF1.3` uses flow-based meters to enforce QoS for the virtual networks. This would enable a fully isolated environment with performance guarantees.

5 Presentation

Network Virtualization

----Group 10----

"Advances in Network-Slicing"

Introduction to Software-defined Networking

Presented by
Hari Raghavendar Rao 11334055
Seshagiri Prabhu 21410690

OpenVirteX

Ali Al-Shabibi, Marc De Leenheer, Matteo Gerola, Ayaka Koshibe, Guru Parulkar, William Snow

Network Virtualization

- Enable multi-tenancy
- Decouple the physical network from the virtual network
- Topology virtualization
- Allow security and user traffic isolation

Related works or Existing solutions

Proprietary

- Some use overlay based approaches
- Network core only for simple forwarding
- Use SDN to use NV but take SDN away from tenants

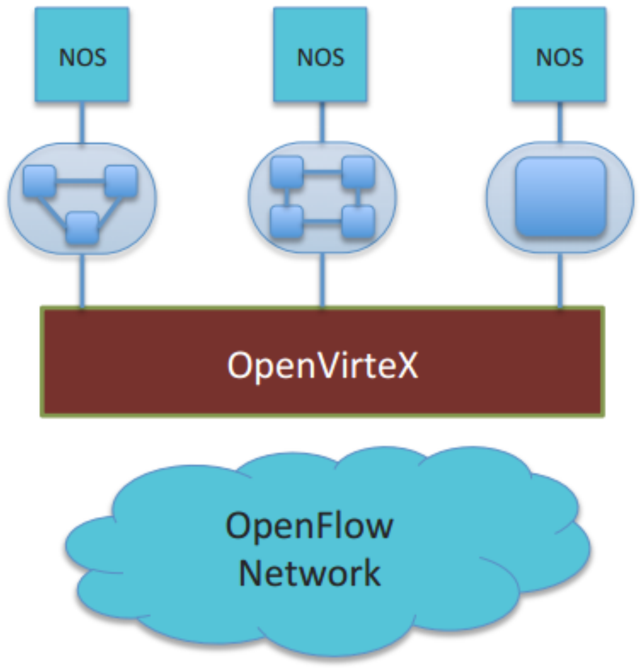
Open Source

- Flow Space Slicing
- Header space shared amongst tenants
- Configuration complexity increases exponentially with number of tenants

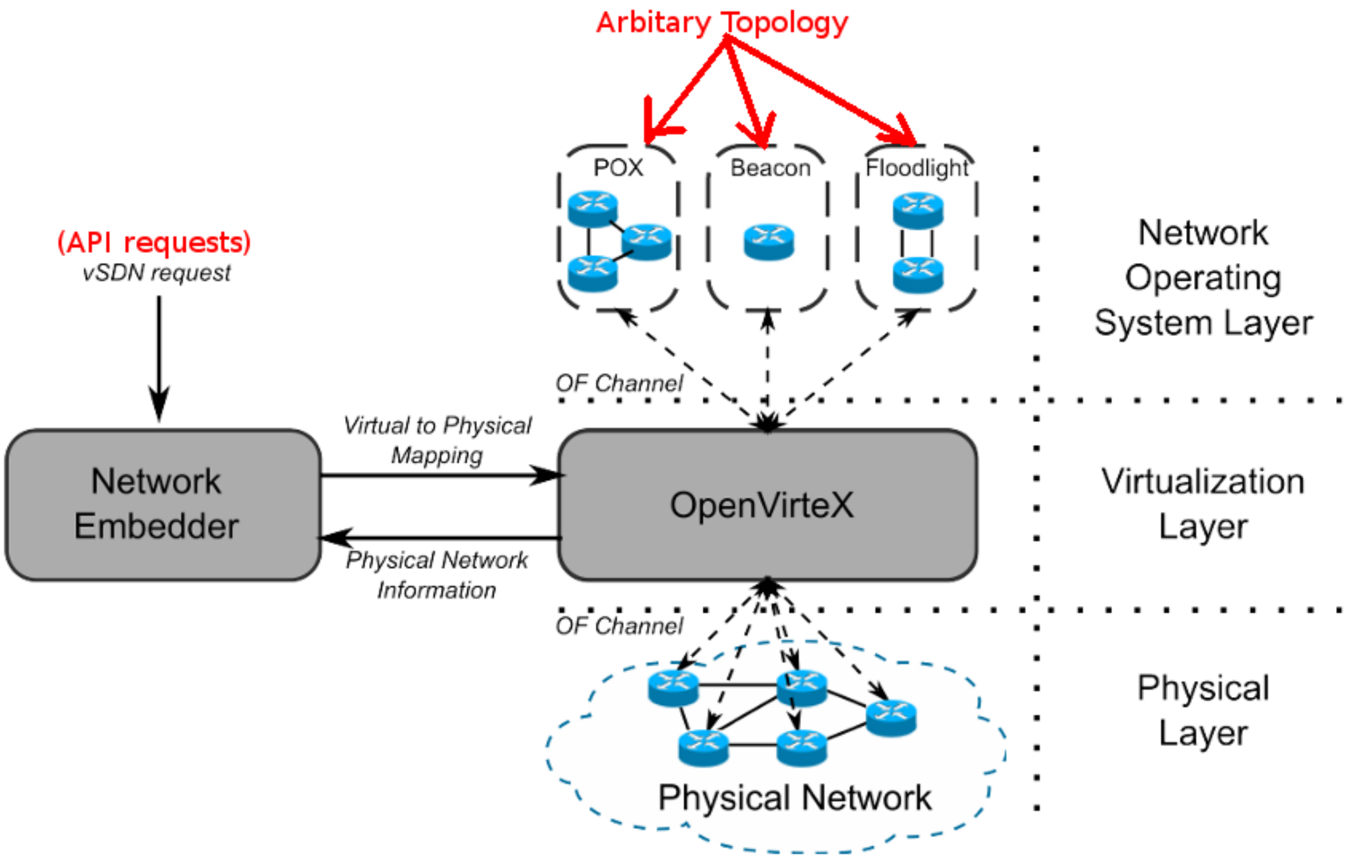
OpenVirteX

OpenVirteX enables the virtualization of OpenFlow networks

- Address Space Virtualization
- Topology Virtualization
- Programmability through OF
- Resiliency
- Portability



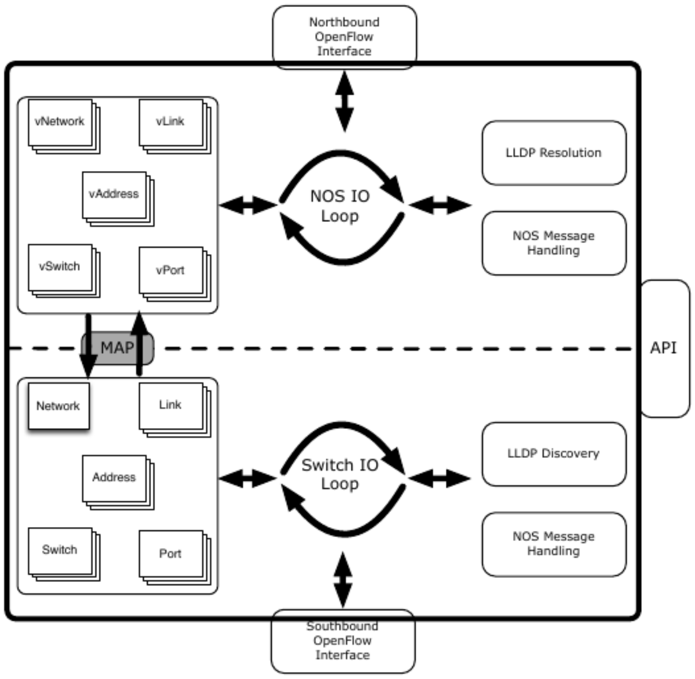
OpenVirteX - System Architecture



OpenVirteX - Internal Architecture

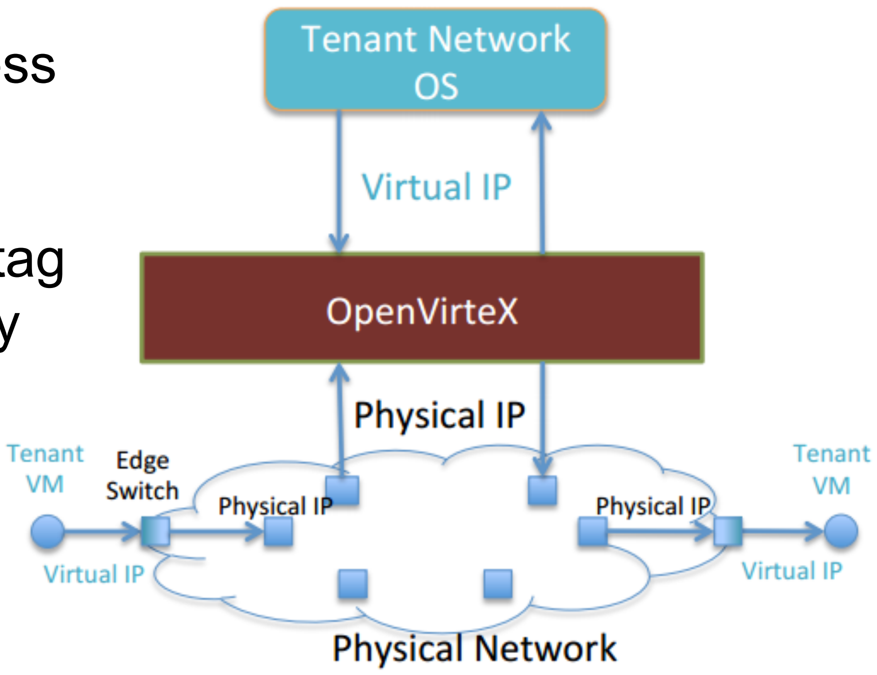
Tenant traffic isolation using MAC address fields

- Flow ID
  - Looks up the particular flow that entered the virtual link and restore its values when the traffic exits the virtual link
- Tenant ID
  - Isolates the different tenants on the virtual link
- Virtual Link ID
  - Keeps tracks of virtual links



Address Virtualization

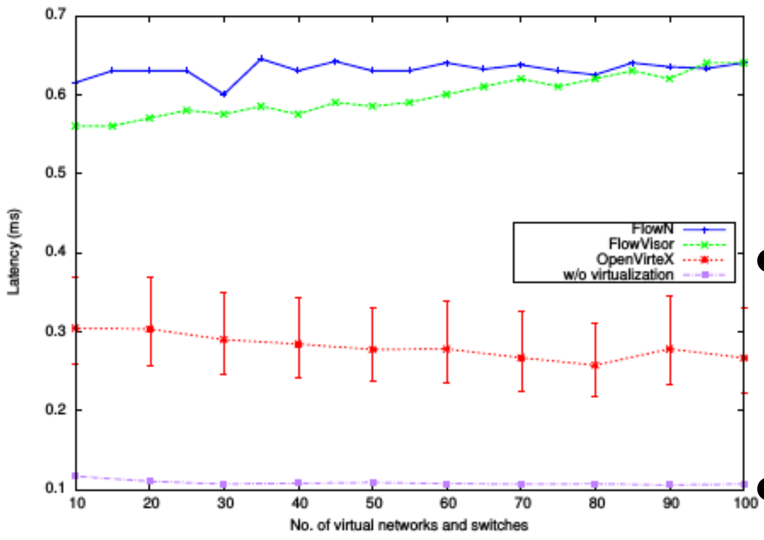
- Multiple virtual networks can use the same address space
- The rewriting inserts a tag to enable OVX to identify the packets owner
- Rewriting is completely transparent to NOS and end hosts.



Topology Virtualization

- OVX allows tenants to specify their own arbitrary topology
- OVX maps the vSDN topologies to the actual physical network. No need of isomorphism in the tenant design.
- When an LLDP message arrives at a virtual switch element with a certain outport specified, OVX “knows” where the other end of that link is.
- LIMITATION: Single physical switch cannot be partitioned

Results



Future works

- Snapshotting and migrations of vSDN:
  - Ability to preserve the state and data for fast recovery and ease of migration and duplication
- Evolving beyond OF*v1.0*:
  - Integrate OpenFlowJ-Loxi engine that generates version-agnostic OF libs for multi-languages
- vSDN based QoS:
  - In OF*v1.0*, bandwidth limit can be enforced by by statically assign a specific port queue to VN. Migration to OF*v1.3* will fix this

Conclusion

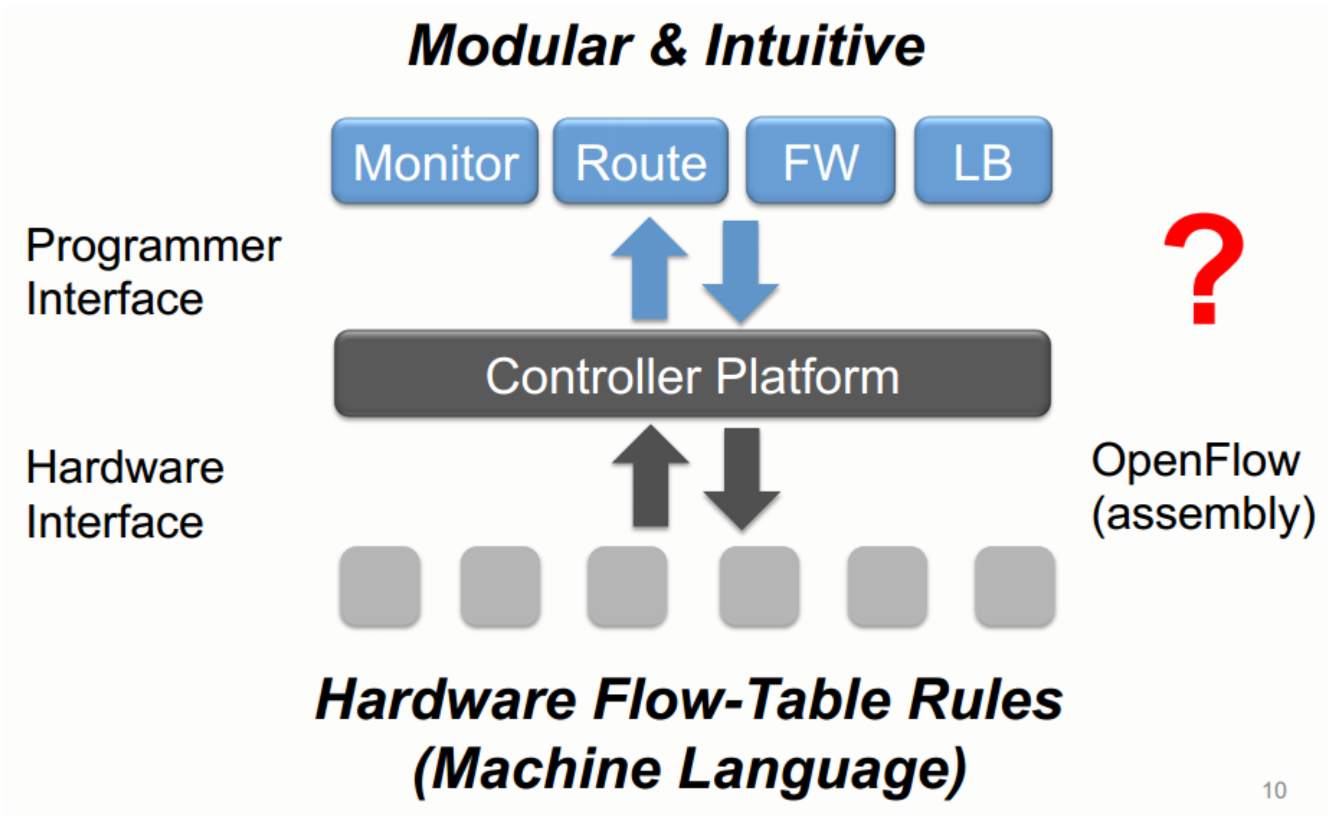
- OpenVirteX is capable of presenting tenants with configurable SDNs (topology customization)
- Full traffic address virtualization of tenant networks achieved (isolation & security)
- Modest performance when compared to the existing solutions.
- OpenVirteX Architecture enables the introduction of features and enhancements such as link resilience to tenant networks

CoVisor: A Compositional Hypervisor for Software-Defined Networks

Xin Jin, Jennifer Gossels, Jennifer Rexford, David Walker

Princeton University

Coordinating Control-plane Modules:



Covisor:

In past hypervisors used to focus on “*slicing*” the network into disjoint parts of “*separate control*” by “*separate entities*”.

CoVisor allows multiple controllers to collaborate on processing the same traffic..

Compositional Network Hypervisor:

Assembly of multiple controllers:

- ❏ Single Network administrator assemble multiple controllers in flexible and configurable manner.
- ❏ Compose Data plane policies :

Parallel: Allow multiple controllers to act independently on same packets at same time.

Sequentially: Allow one controller to choose to process certain traffic before another.

Overriding: Allow one controller to choose to act or defer control to another controller.

Definition of abstract topologies:

- ❑ To protect the physical infrastructure:

“One physical switch to many logical switches”

“Allowing the administrator to provide a custom virtual topology to each controller”

Ex: “Firewall controller” the administrator may abstract the network as “big virtual switch” but the firewall doesn’t need to know the underlying topology to determine if packet should be forwarded or dropped. In contrast a “routing controller” needs the exact topology to perform its task effectively.

Protection against misbehaving controllers:

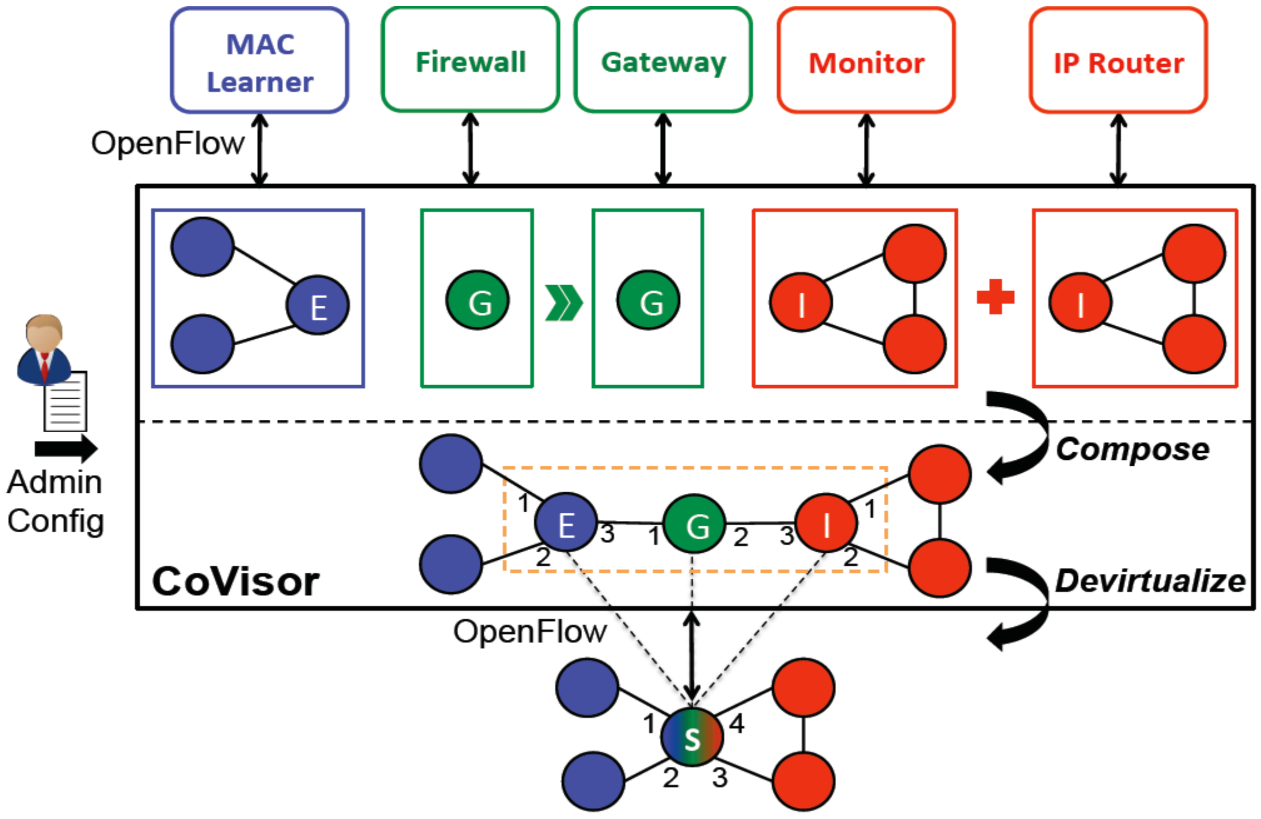
Restriction : what a controller can see of the physical topology.

Control: How a controller can process packets.

"CoVisor *Hypervisor can restrict by limiting functionality of the virtual switches exposed to other controller*"

Ex: firewall controllers should not be allowed to modify packets and MAC learner should not able to inspect IP or TCP headers.

CoVisor overview:



Functionality of CoVisor:

- ❏ Covisor uses a *novel algorithm* to *incrementally compose* applications.
- ❏ Rule priorities:calculate priorities for new rules using convenient Algebra .
- ❏ Removes the need to recompile from scratch for every rule update.
- ❏ It translates the composed policy into rules for physical topology. (one physical switch mapped to multiple virtual switches)
- ❏ After compiling the policy, covisor sends the necessary rule to update to switches.

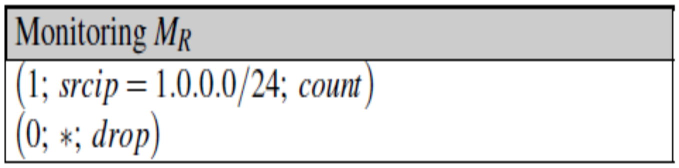
Challenge:

- ❑ Optimise the compilation process
- ❑ Flow table compilation efficiency.

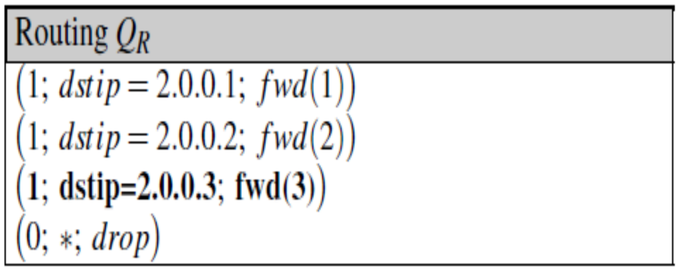
Solution:

- ❏ Incremental compilation
    - ❏ Incremental priority assignment
    - ❏ Advanced indexing for fast rule lookup

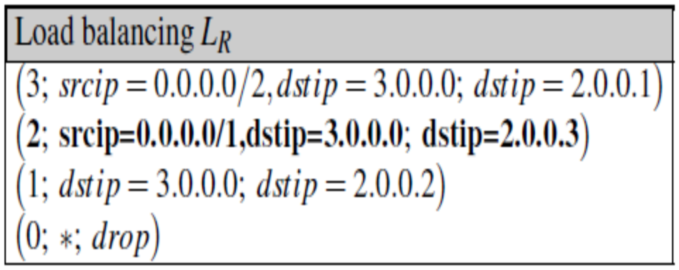
Incremental Compilation:



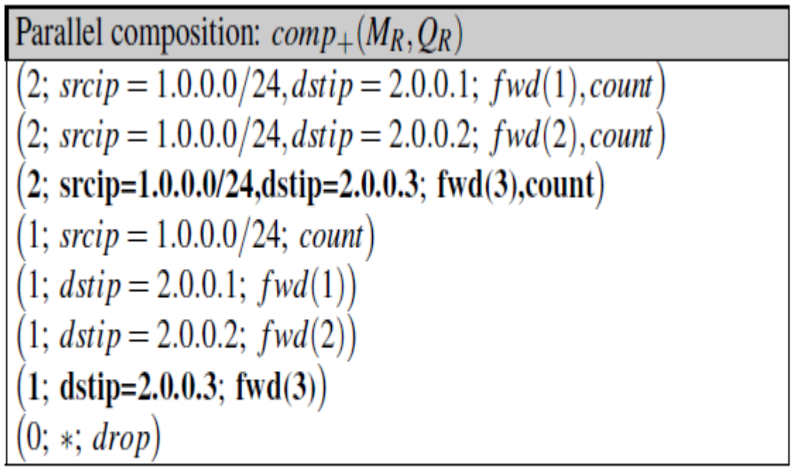
Monitoring M_R
$(1; srcip = 1.0.0.0/24; count)$
$(0; *; drop)$



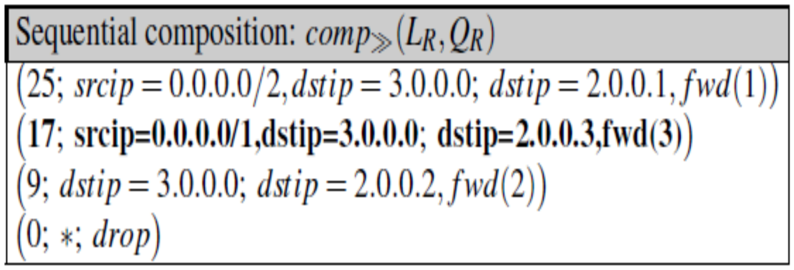
Routing Q_R
$(1; dstip = 2.0.0.1; fwd(1))$
$(1; dstip = 2.0.0.2; fwd(2))$
(1; dstip=2.0.0.3; fwd(3))
$(0; *; drop)$



Load balancing L_R
$(3; srcip = 0.0.0.0/2, dstip = 3.0.0.0; dstip = 2.0.0.1)$
(2; srcip=0.0.0.0/1,dstip=3.0.0.0; dstip=2.0.0.3)
$(1; dstip = 3.0.0.0; dstip = 2.0.0.2)$
$(0; *; drop)$



Parallel composition: $comp_{+}(M_R, Q_R)$
$(2; srcip = 1.0.0.0/24, dstip = 2.0.0.1; fwd(1), count)$
$(2; srcip = 1.0.0.0/24, dstip = 2.0.0.2; fwd(2), count)$
(2; srcip=1.0.0.0/24,dstip=2.0.0.3; fwd(3),count)
$(1; srcip = 1.0.0.0/24; count)$
$(1; dstip = 2.0.0.1; fwd(1))$
$(1; dstip = 2.0.0.2; fwd(2))$
(1; dstip=2.0.0.3; fwd(3))
$(0; *; drop)$



Sequential composition: $comp_{\gg}(L_R, Q_R)$
$(25; srcip = 0.0.0.0/2, dstip = 3.0.0.0; dstip = 2.0.0.1, fwd(1))$
(17; srcip=0.0.0.0/1,dstip=3.0.0.0; dstip=2.0.0.3,fwd(3))
$(9; dstip = 3.0.0.0; dstip = 2.0.0.2, fwd(2))$
$(0; *; drop)$

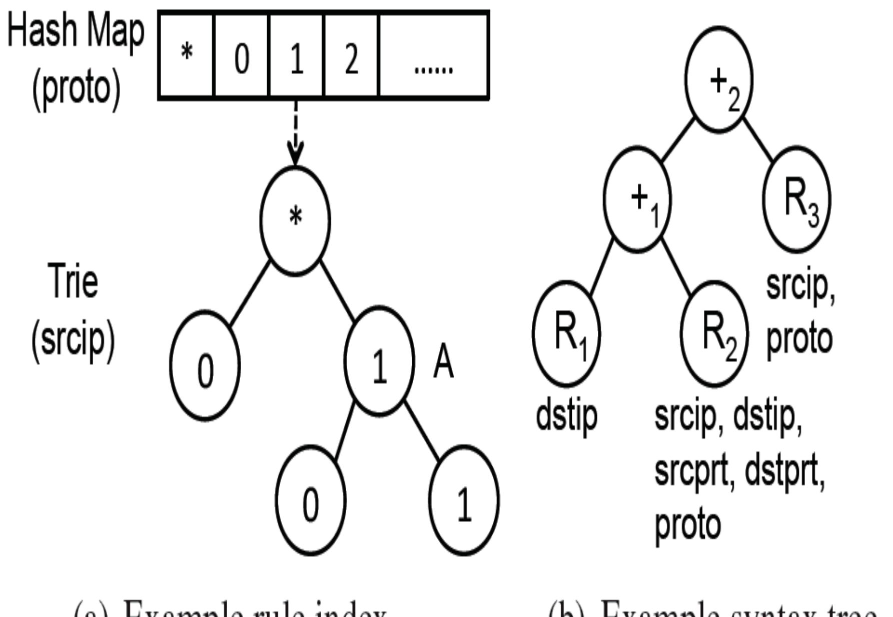
Incremental Priority Assignment:

- Heuristics
  - Parallel: $r_k.priority = r_{1i}.priority + r_{2j}.priority.$
  - Sequential: $r_k.priority = r_{1i}.priority \times MAX_{R_2} + r_{2j}.priority.$
  - Overriding: $r_k.priority = \begin{cases} r_k.mPriority \times MAX_{R_2} & \text{if } r_k \in R_1 \\ r_k.mPriority, & \text{if } r_k \in R_2 \end{cases}$

Advanced Indexing Structure:

- Given a target rule R_t, how to efficiently find the rules in a flow table that overlap with R_t?

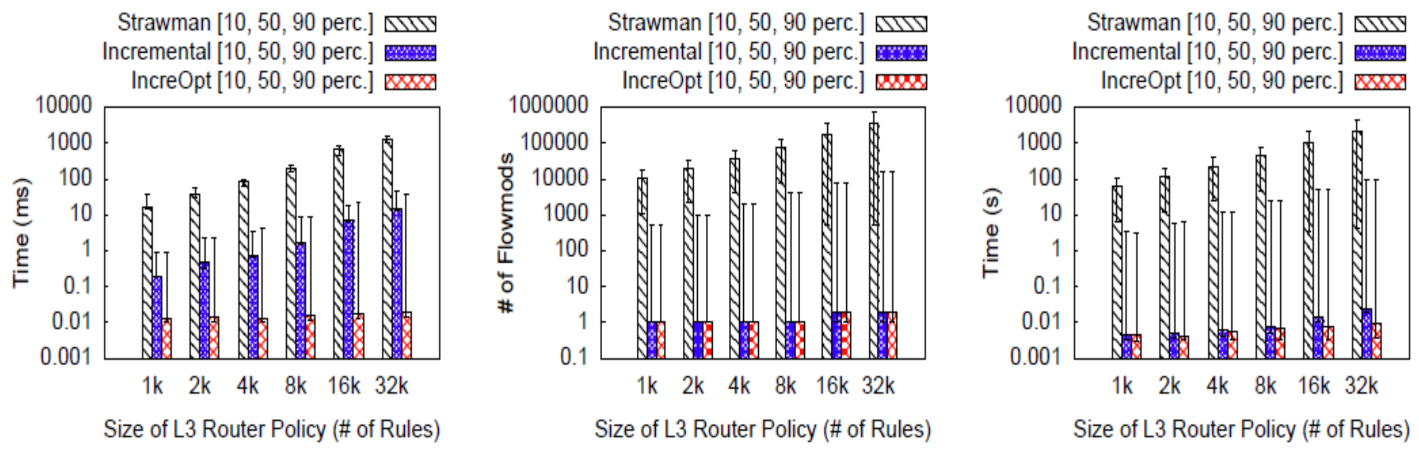
Hashtable : exact match
Trie : prefix-match
list of arbitrary: wildcard match



(a) Example rule index. (b) Example syntax tree.

Implementation and Evaluation:

- Implement with 4k lines of java code on OpenVirteX.
- Simulation experiments against a strawman compiler.



(a) Compilation Time (b) Rule Update Overhead (c) Total Update Time

Figure 10: Per-rule update overhead of L3-L4 Firewall $\gg$ L3 Router (log-log scale).

Conclusion:

- ❑ Administrator can combine multiple controllers to collaboratively process a network’s traffic.
- ❑ Covisor uses a combination of *novel algorithms* and data structures to efficiently compile policies in an incremental manner.
- ❑ Evaluations on covisor prototype shows it faster than a naive implementation.

References:

Ali Al-Shabibi, M.De Leenheer, M. Gerola,A.Koshibe,G.Parulkar, E. Salvadori, and B. Snow , "OpenVirteX: Make Your Virtual SDNs Programmable", *ACM SIGCOMM HotSDN* workshop,August 2014.

Xin Jin ,Jennifer Gossels,Jennifer Rexford,David Walker *Princeton University* "CoVisor: A Compositional Hypervisor for Software-Defined Networks", *USENIX NSDI May 4-6 2015.*

Thank You

Any Questions?