Virtual Cluster Embeddings for Software-Defined Networks

Akshay(CS12B034)

Ranjan(CS12B050)

Acknowledgements

- We would like to heartily thank Prof Marco Canini of UCL Italy for always helping us with all the doubts we had related to Frenetic VM and guiding us when all else had failed.
- We would also like to thank the Frenetic team who have been vociferous and enthusiastic in response to all our queries and bugs.

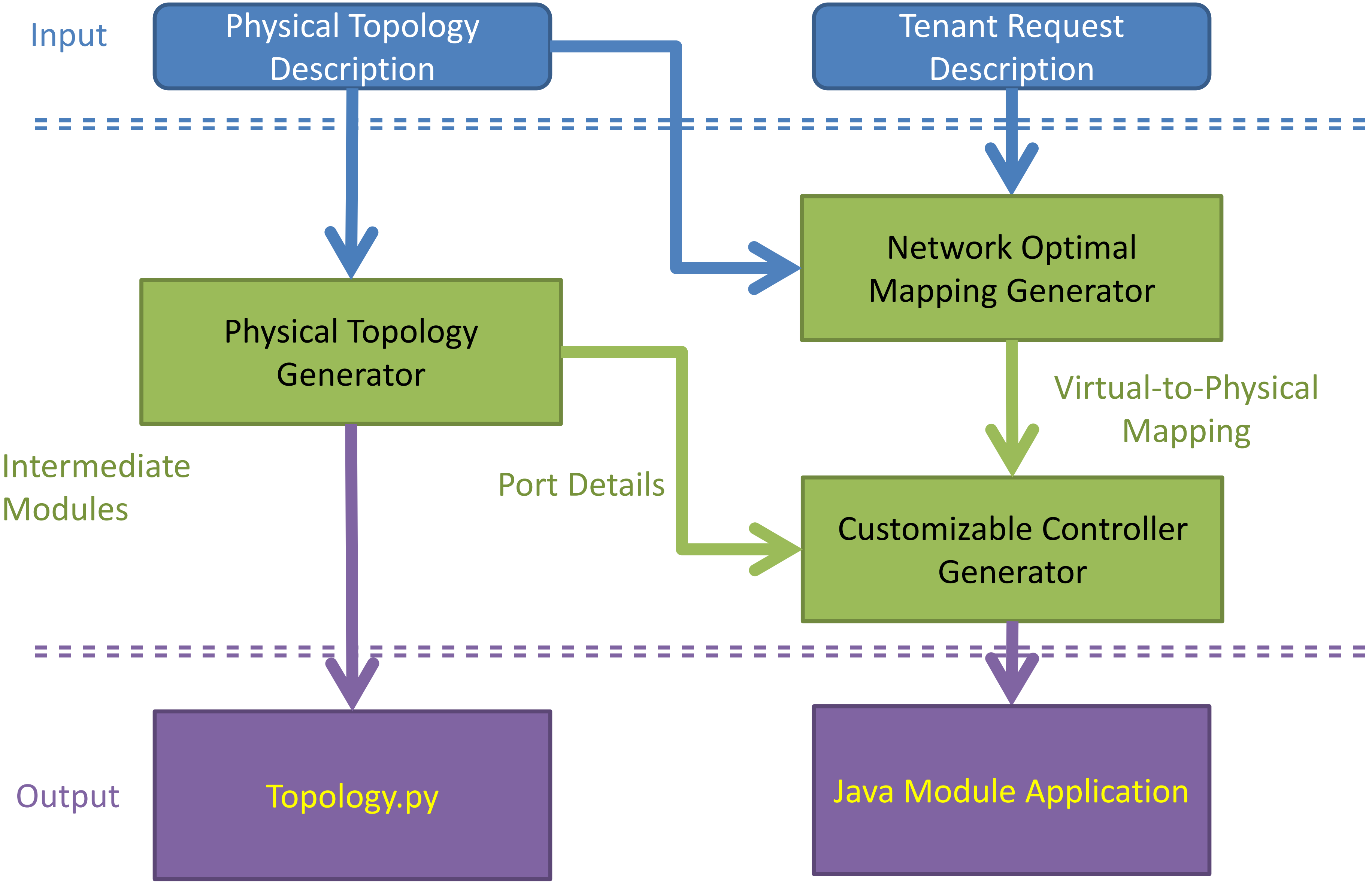
What is it all about?

- **Servicing requests** of virtualization from tenants
- Minimizing cost of allocation while guaranteeing **quality-of-service(QOS)**
- **Generating network flows** for the virtual network for its functionality as a SDN
- **Faster and more optimal allocation** mechanisms to speed up the update process

What we have done?

- **Developing a Embed Compiler**
  - **Input:** Tenant Request + Physical Topology Descriptor File
  - **Output:** A working Mininet topology + Optimal Virtualisation Java Module for Floodlight Controller
- **Developed a modular toolkit called “VirNet” to facilitate creation of physical topologies and implement virtualization.**
- **Proposed a segmented architecture which allows you plug-in your own replacement modules and customize the virtualization.**

VirNet Architecture



Input

Physical Topology Description

Tenant Request Description

Intermediate Modules

Physical Topology Generator

Network Optimal Mapping Generator

Virtual-to-Physical Mapping

Port Details

Customizable Controller Generator

Output

Topology.py

Java Module Application

This file will generate the physical topology with virtual OpenFlow switches using Mininet

This will generate flow rules using REST API to be accepted by the Floodlight controller

Physical Topology Description

- Description of the underlying physical topology in a specified format.
- Attributes right now restricted to BW for the project’s purposes, though can be trivially extended to many others.
- An example dumbbell topology description file:
  - **4** <- No. of Hosts
  - **2** <- No. of Switches
  - **5** <- No. of Links

    s1,**s2**,**2** <- Description of each link in format <Node1,Node2,BW(in Mbps)>

    s1,**h1**,**1**

    s1,**h2**,**1**

    s2,**h3**,**1**

    s2,**h4**,**1**

Tenant Request Description

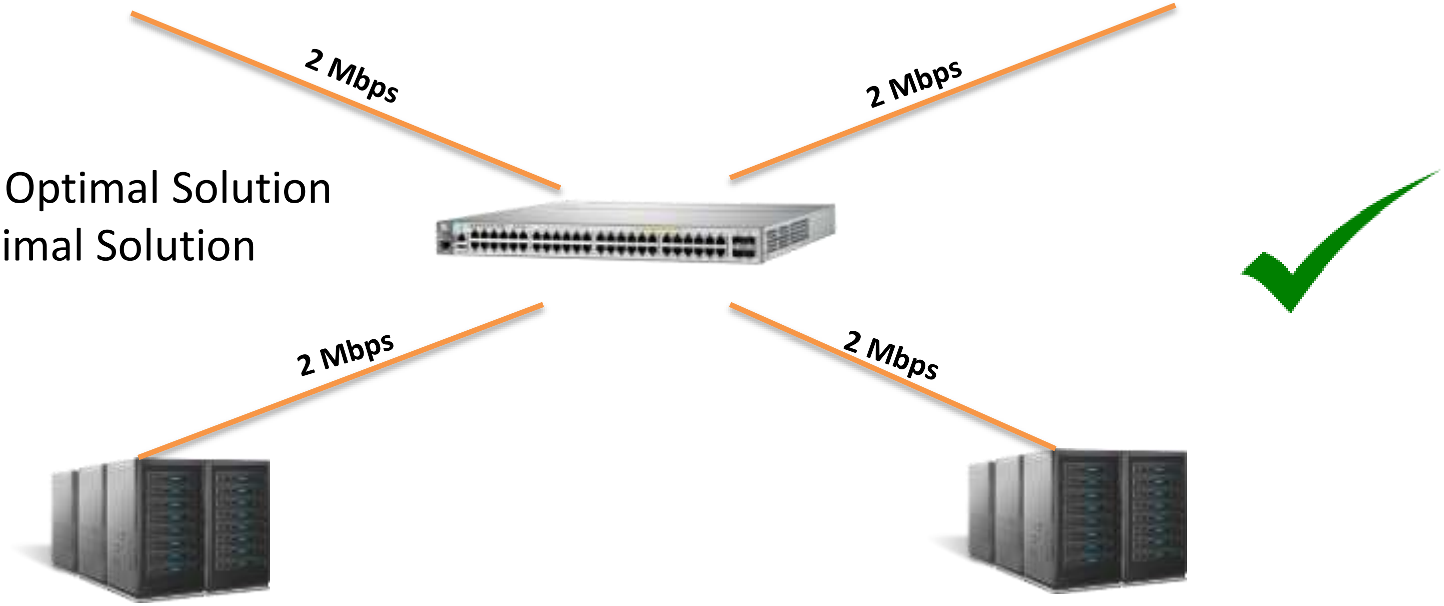
- Description of the tenant’s request in a specified format
- We have enabled various customizations for this such as the tenant can
  - Request different bandwidth requests for different hosts
  - Constraint the central switch to be one in the specified set
- A sample tenant request description file looks like
  - **4** <- No. of hosts
    h1,**1** <- Request description in the format <HostId,BW>
    h2,**1**
    h3,**1**
    h4,**1**

Physical Topology Generator

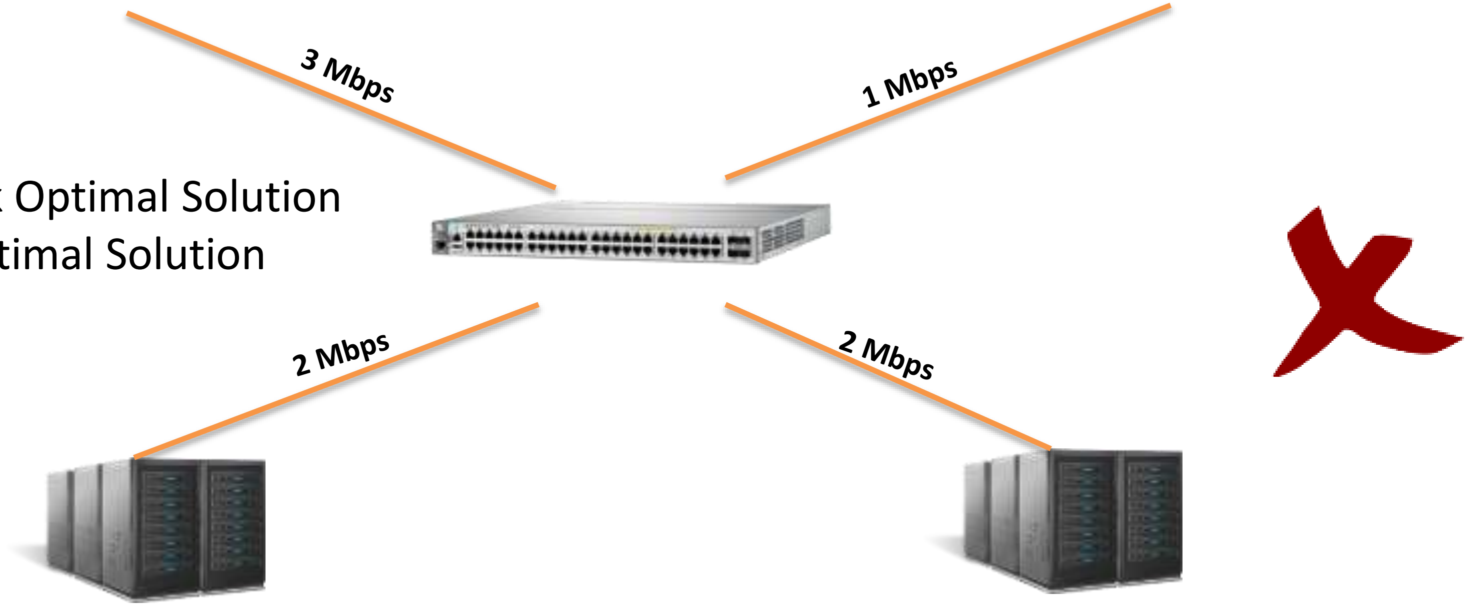
- Given a physical topology, it generates a python file which imports Mininet libraries to instantiate the physical topology using virtual OVF switches in Mininet.
- This stage is also critical in generating details specific to the instantiation of the physical topology like port details which are used by other modules.

Network Optimal Mapping Generator

- Given a physical topology description and a tenant request, this module executes our network optimal embedding algorithm to generate the mapping
- This generates a set of paths fro each of the requested hosts to the central switch which has been selected
- We even have an additional feature which will tell the tenant the best possible allocation that can be serviced if the requested virtualization cannot be performed.



✓ Network Optimal Solution
✓ Flow Optimal Solution

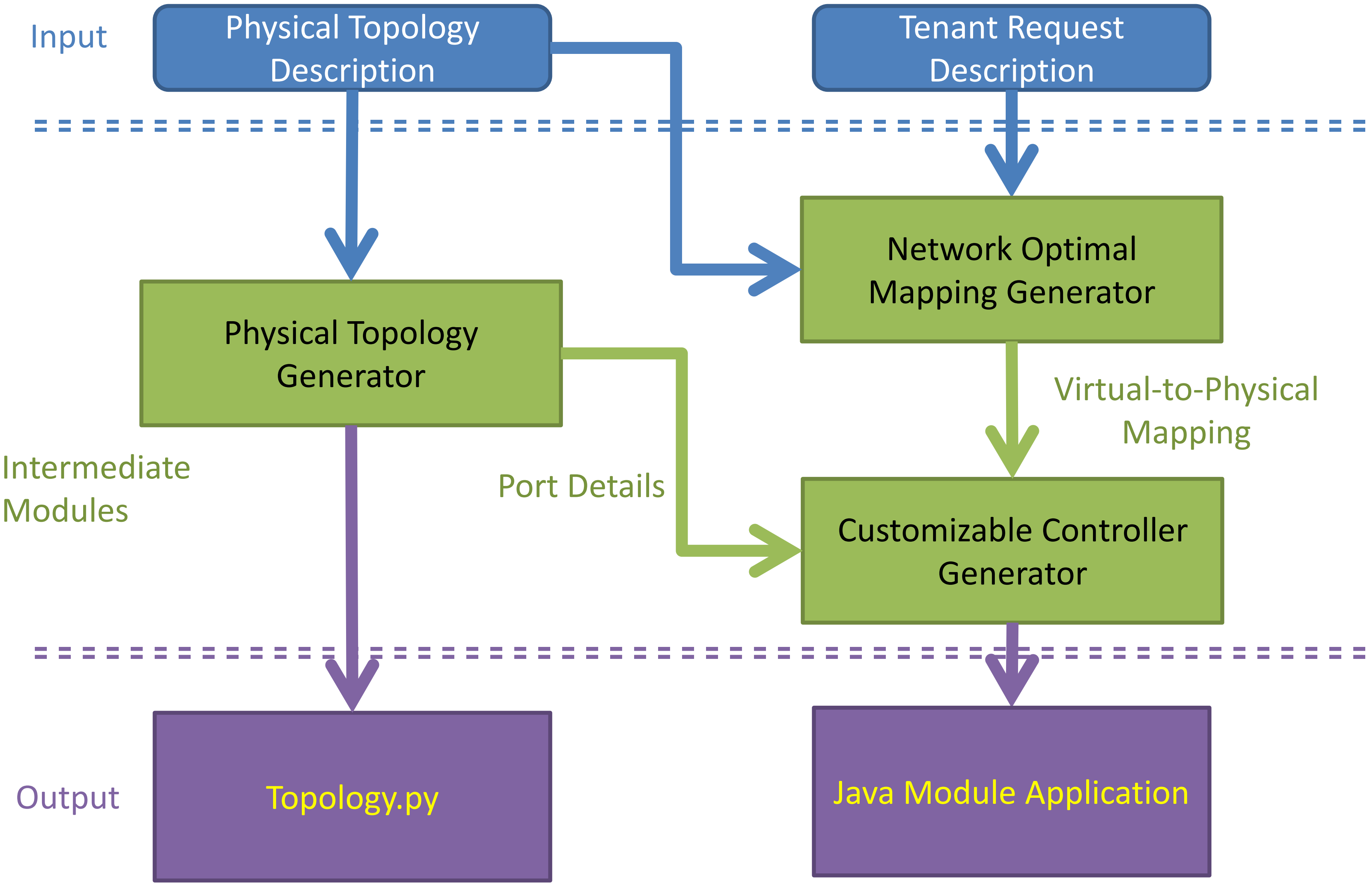


✗ Network Optimal Solution
✓ Flow Optimal Solution

Customizable controller generator

- It receives the paths from the previous module and the port details from the physical topology generator and generates a Floodlight Java module that installs the appropriate flow rules using the REST API
- It uses source MAC and in_port to route the packet to the central switch. Then from the central switch to the destination it installs the rules using in_port and destination MAC.

VirNet Architecture

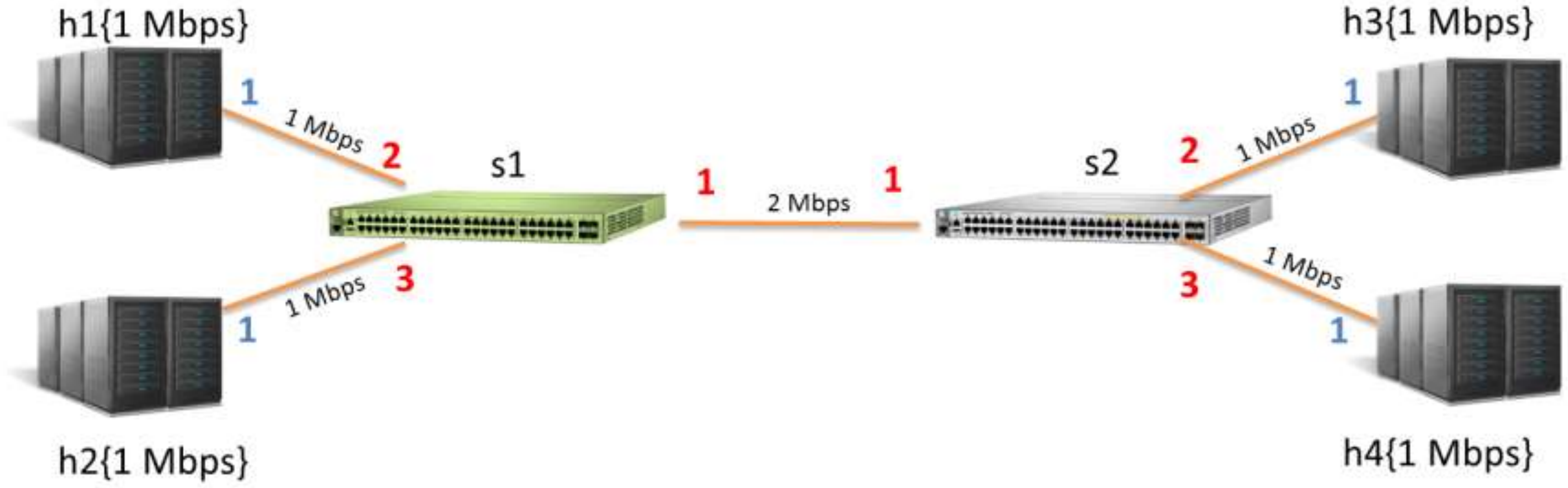


Input
Physical Topology Description
Tenant Request Description
Network Optimal Mapping Generator
Physical Topology Generator
Virtual-to-Physical Mapping
Intermediate Modules
Port Details
Customizable Controller Generator
Output
Topology.py
Java Module Application

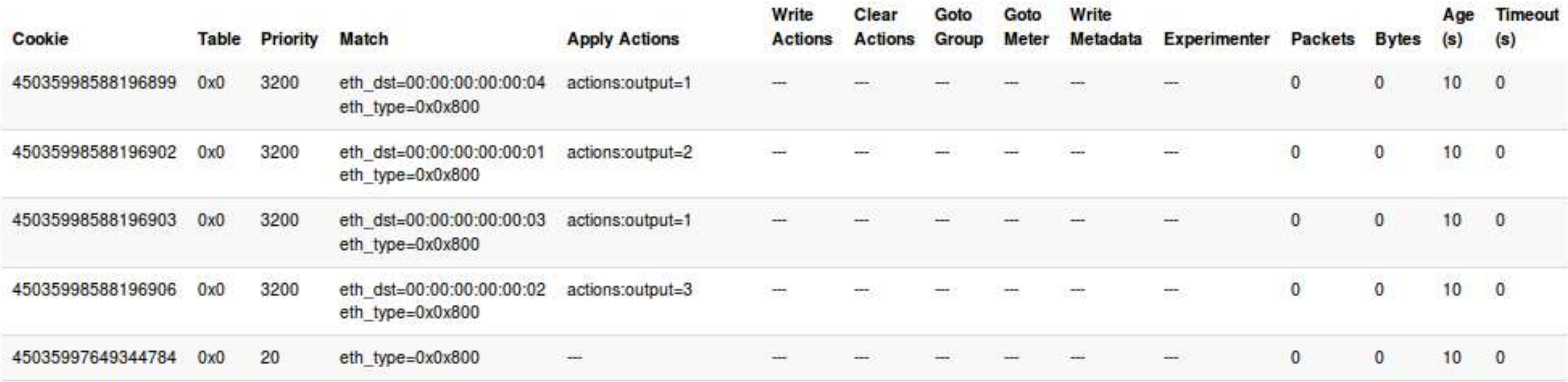
This file will generate the physical topology with virtual OpenFlow switches using Mininet

This will generate flow rules using REST API to be accepted by the Floodlight controller

Example 1- Dumbbell topology

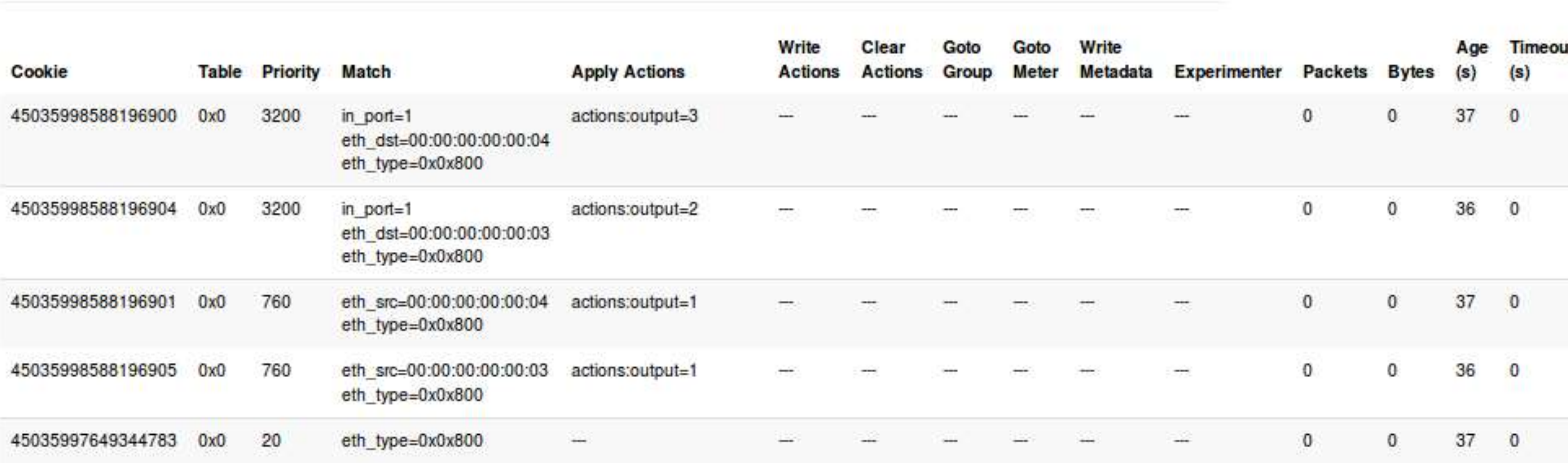


S1 Flowtable



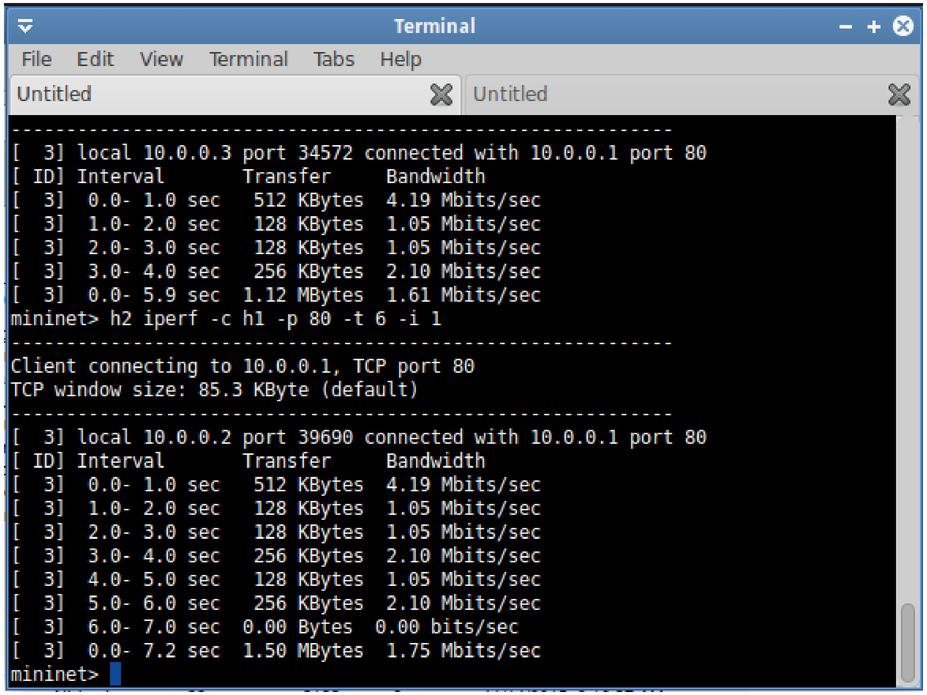
Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
45035998588196899	0x0	3200	eth_dst=00:00:00:00:00:04 eth_type=0x0x800	actions:output=1	—	—	—	—	—	—	0	0	10	0
45035998588196902	0x0	3200	eth_dst=00:00:00:00:00:01 eth_type=0x0x800	actions:output=2	—	—	—	—	—	—	0	0	10	0
45035998588196903	0x0	3200	eth_dst=00:00:00:00:00:03 eth_type=0x0x800	actions:output=1	—	—	—	—	—	—	0	0	10	0
45035998588196906	0x0	3200	eth_dst=00:00:00:00:00:02 eth_type=0x0x800	actions:output=3	—	—	—	—	—	—	0	0	10	0
45035997649344784	0x0	20	eth_type=0x0x800	—	—	—	—	—	—	—	0	0	10	0

S2 Flowtable

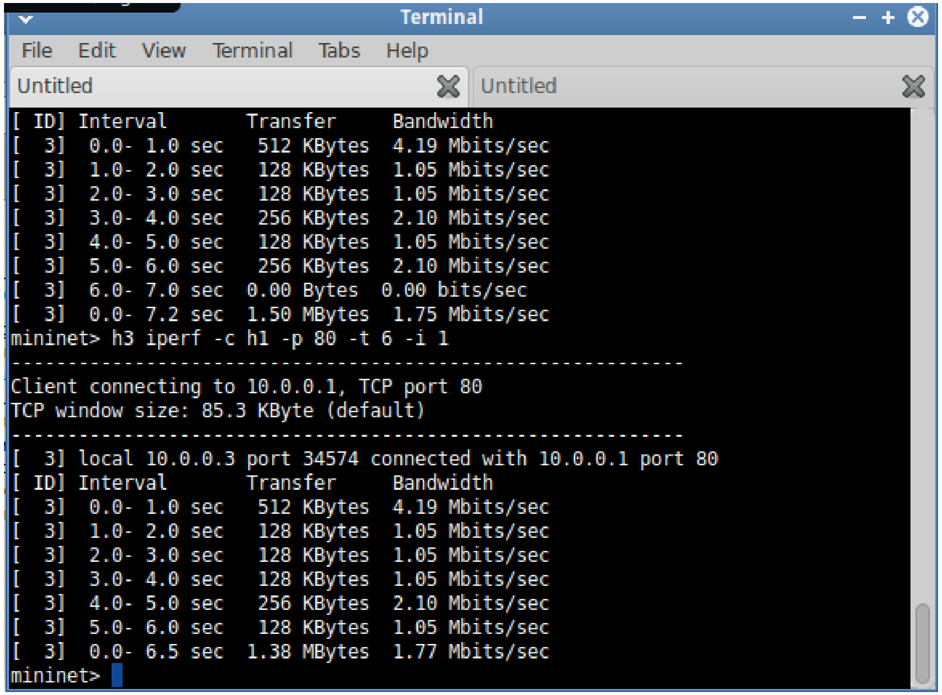


Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeou (s)
45035998588196900	0x0	3200	in_port=1 eth_dst=00:00:00:00:00:04 eth_type=0x0x800	actions:output=3	—	—	—	—	—	—	0	0	37	0
45035998588196904	0x0	3200	in_port=1 eth_dst=00:00:00:00:00:03 eth_type=0x0x800	actions:output=2	—	—	—	—	—	—	0	0	36	0
45035998588196901	0x0	760	eth_src=00:00:00:00:00:04 eth_type=0x0x800	actions:output=1	—	—	—	—	—	—	0	0	37	0
45035998588196905	0x0	760	eth_src=00:00:00:00:00:03 eth_type=0x0x800	actions:output=1	—	—	—	—	—	—	0	0	36	0
45035997649344783	0x0	20	eth_type=0x0x800	—	—	—	—	—	—	—	0	0	37	0

Running iperf to test the system

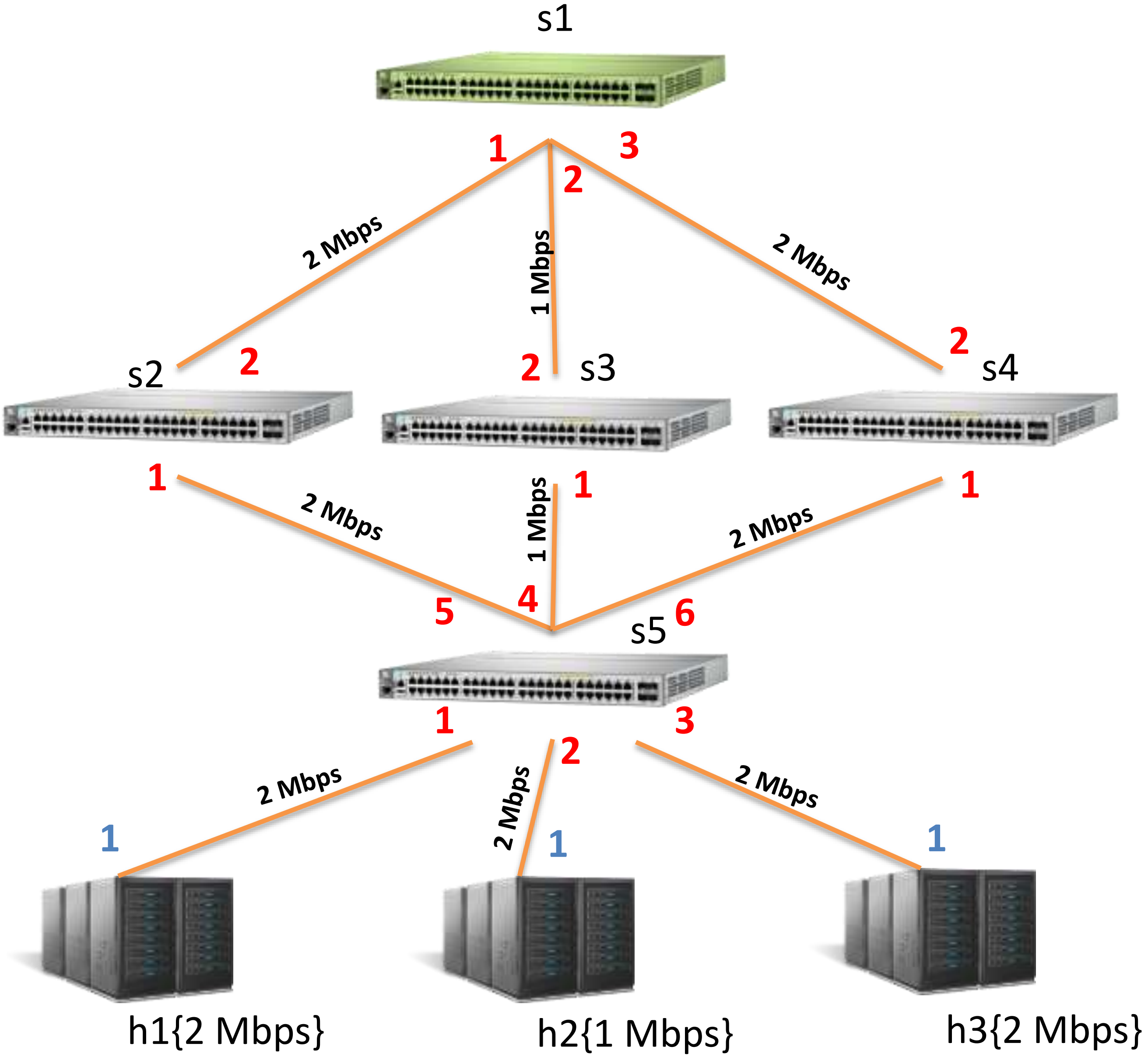


```
------------------------------------------------------------
[  3] local 10.0.0.3 port 34572 connected with 10.0.0.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  0.0- 5.9 sec  1.12 MBytes  1.61 Mbits/sec
mininet> h2 iperf -c h1 -p 80 -t 6 -i 1
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 39690 connected with 10.0.0.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  6.0- 7.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  0.0- 7.2 sec  1.50 MBytes  1.75 Mbits/sec
mininet>
```

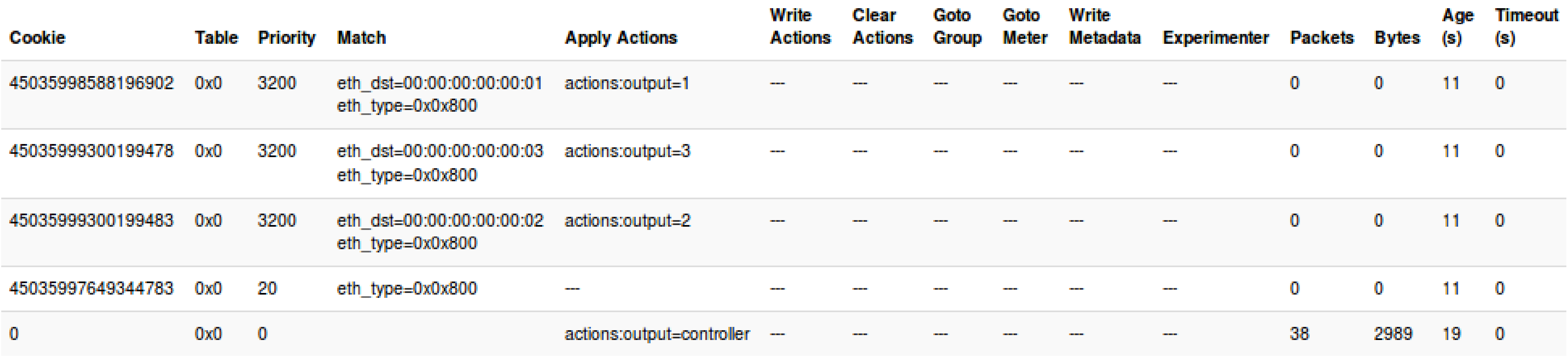


```
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  6.0- 7.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  0.0- 7.2 sec  1.50 MBytes  1.75 Mbits/sec
mininet> h3 iperf -c h1 -p 80 -t 6 -i 1
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.3 port 34574 connected with 10.0.0.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0- 6.5 sec  1.38 MBytes  1.77 Mbits/sec
mininet>
```

Example 2-Distributed topology

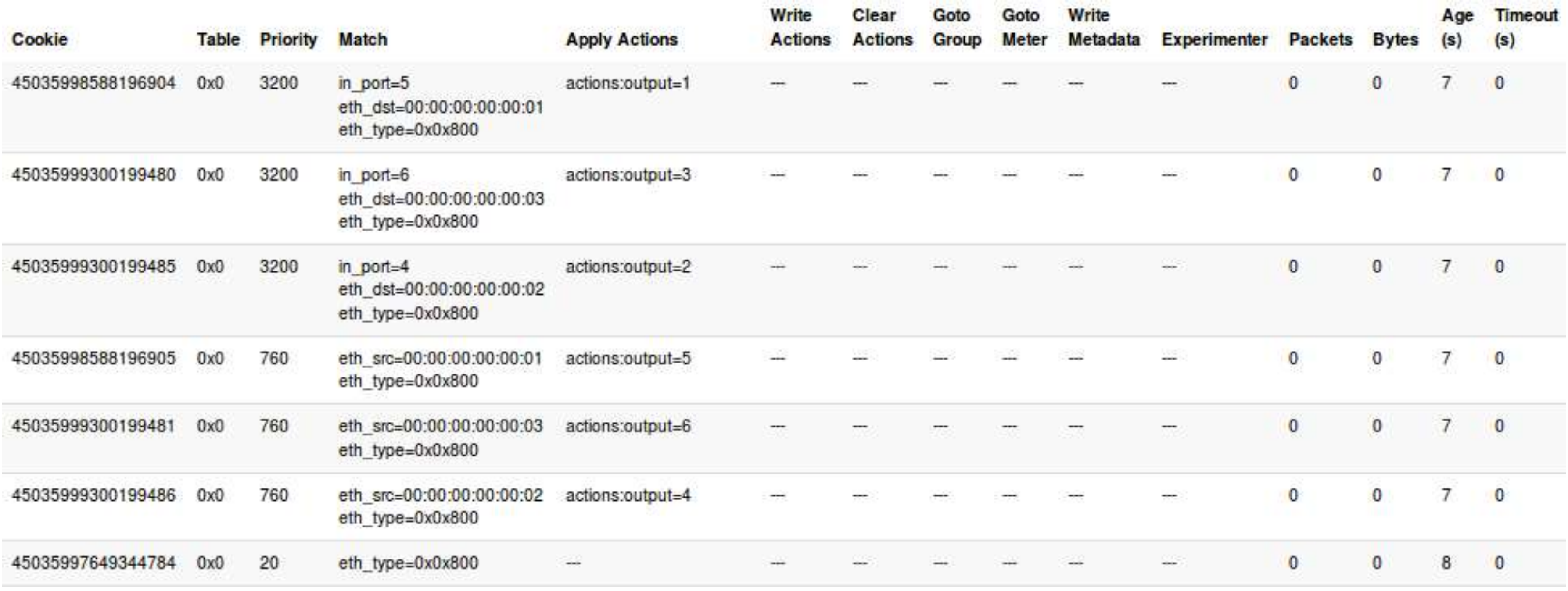


S1 Flowtable



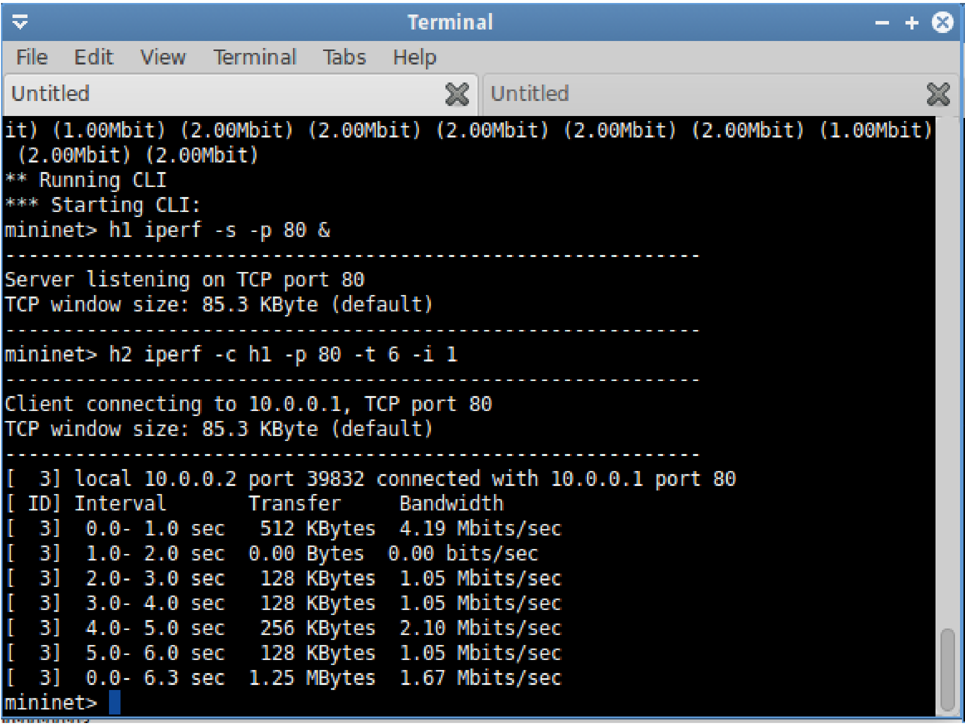
Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
45035998588196902	0x0	3200	eth_dst=00:00:00:00:00:01 eth_type=0x0x800	actions:output=1	---	---	---	---	---	---	0	0	11	0
45035999300199478	0x0	3200	eth_dst=00:00:00:00:00:03 eth_type=0x0x800	actions:output=3	---	---	---	---	---	---	0	0	11	0
45035999300199483	0x0	3200	eth_dst=00:00:00:00:00:02 eth_type=0x0x800	actions:output=2	---	---	---	---	---	---	0	0	11	0
45035997649344783	0x0	20	eth_type=0x0x800	---	---	---	---	---	---	---	0	0	11	0
0	0x0	0		actions:output=controller	---	---	---	---	---	---	38	2989	19	0

S5 Flowtable

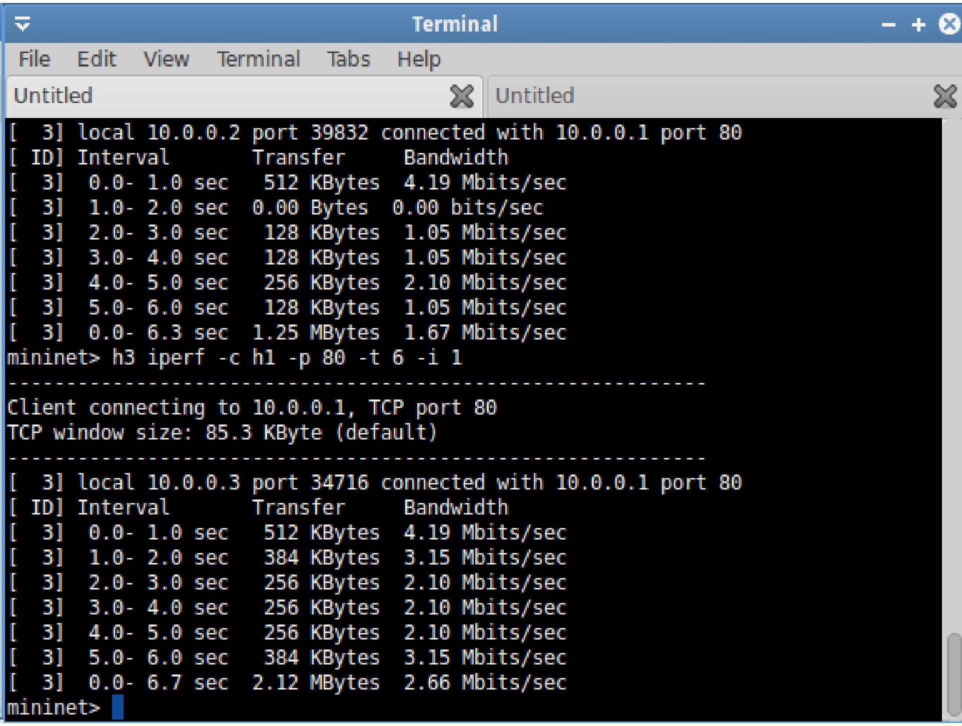


Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
45035998588196904	0x0	3200	in_port=5 eth_dst=00:00:00:00:00:01 eth_type=0x0x800	actions:output=1	---	---	---	---	---	---	0	0	7	0
45035999300199480	0x0	3200	in_port=6 eth_dst=00:00:00:00:00:03 eth_type=0x0x800	actions:output=3	---	---	---	---	---	---	0	0	7	0
45035999300199485	0x0	3200	in_port=4 eth_dst=00:00:00:00:00:02 eth_type=0x0x800	actions:output=2	---	---	---	---	---	---	0	0	7	0
45035998588196905	0x0	760	eth_src=00:00:00:00:00:01 eth_type=0x0x800	actions:output=5	---	---	---	---	---	---	0	0	7	0
45035999300199481	0x0	760	eth_src=00:00:00:00:00:03 eth_type=0x0x800	actions:output=6	---	---	---	---	---	---	0	0	7	0
45035999300199486	0x0	760	eth_src=00:00:00:00:00:02 eth_type=0x0x800	actions:output=4	---	---	---	---	---	---	0	0	7	0
45035997649344784	0x0	20	eth_type=0x0x800	---	---	---	---	---	---	---	0	0	8	0

Running iperf to test the system



```
it) (1.00Mbit) (2.00Mbit) (2.00Mbit) (2.00Mbit) (2.00Mbit) (2.00Mbit) (1.00Mbit)
 (2.00Mbit) (2.00Mbit)
** Running CLI
*** Starting CLI:
mininet> h1 iperf -s -p 80 &
------------------------------------------------------------
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
mininet> h2 iperf -c h1 -p 80 -t 6 -i 1
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 39832 connected with 10.0.0.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0- 6.3 sec  1.25 MBytes  1.67 Mbits/sec
mininet> 
```



```
[  3] local 10.0.0.2 port 39832 connected with 10.0.0.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0- 6.3 sec  1.25 MBytes  1.67 Mbits/sec
mininet> h3 iperf -c h1 -p 80 -t 6 -i 1
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.3 port 34716 connected with 10.0.0.1 port 80
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  2.0- 3.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  3.0- 4.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  4.0- 5.0 sec   256 KBytes  2.10 Mbits/sec
[  3]  5.0- 6.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  0.0- 6.7 sec  2.12 MBytes  2.66 Mbits/sec
mininet> 
```

Conclusion

- Developed a virtualization toolkit for SDN – VirNet v0.1 – for servicing virtualization requests
- Furnished a virtualization wrapper over the state-of-the-art controllers which was the original goal of the project from the start
- Modularized the process of virtualization into cohesive modules which can be replaced by other custom modules to receive a new embedding.

Questions ?

- All suggestions for improvement are also welcome ☺

That's all Folks!