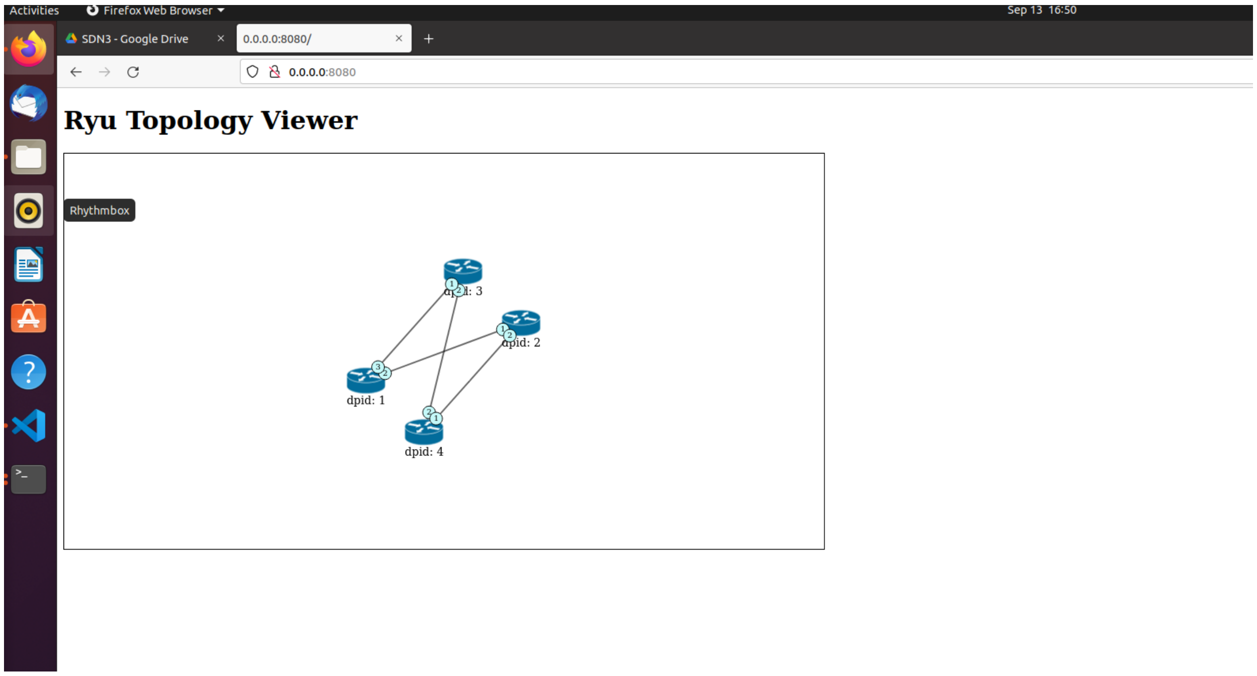
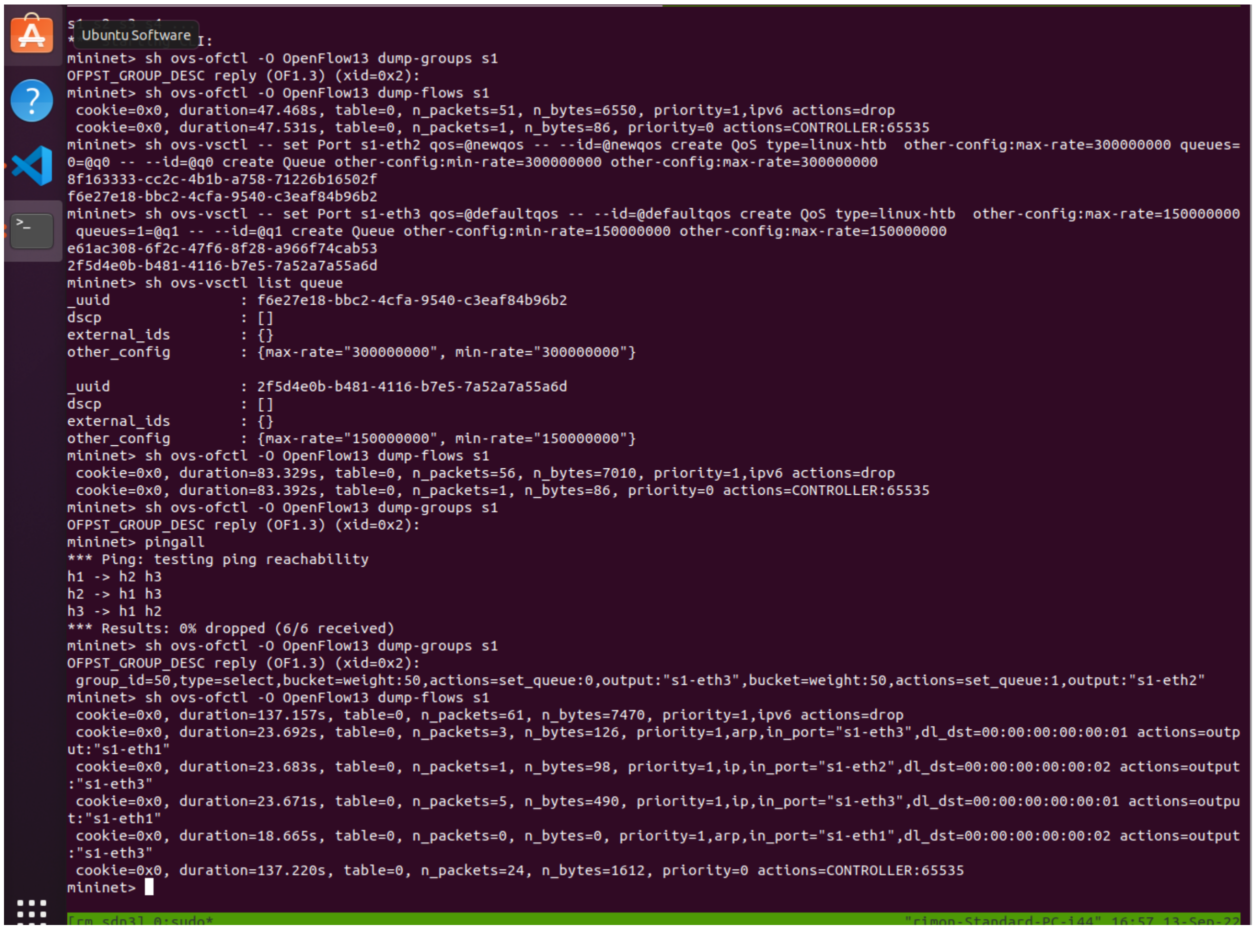
Workshop 3 SDN Report:

1. The topology using the visualization tool:



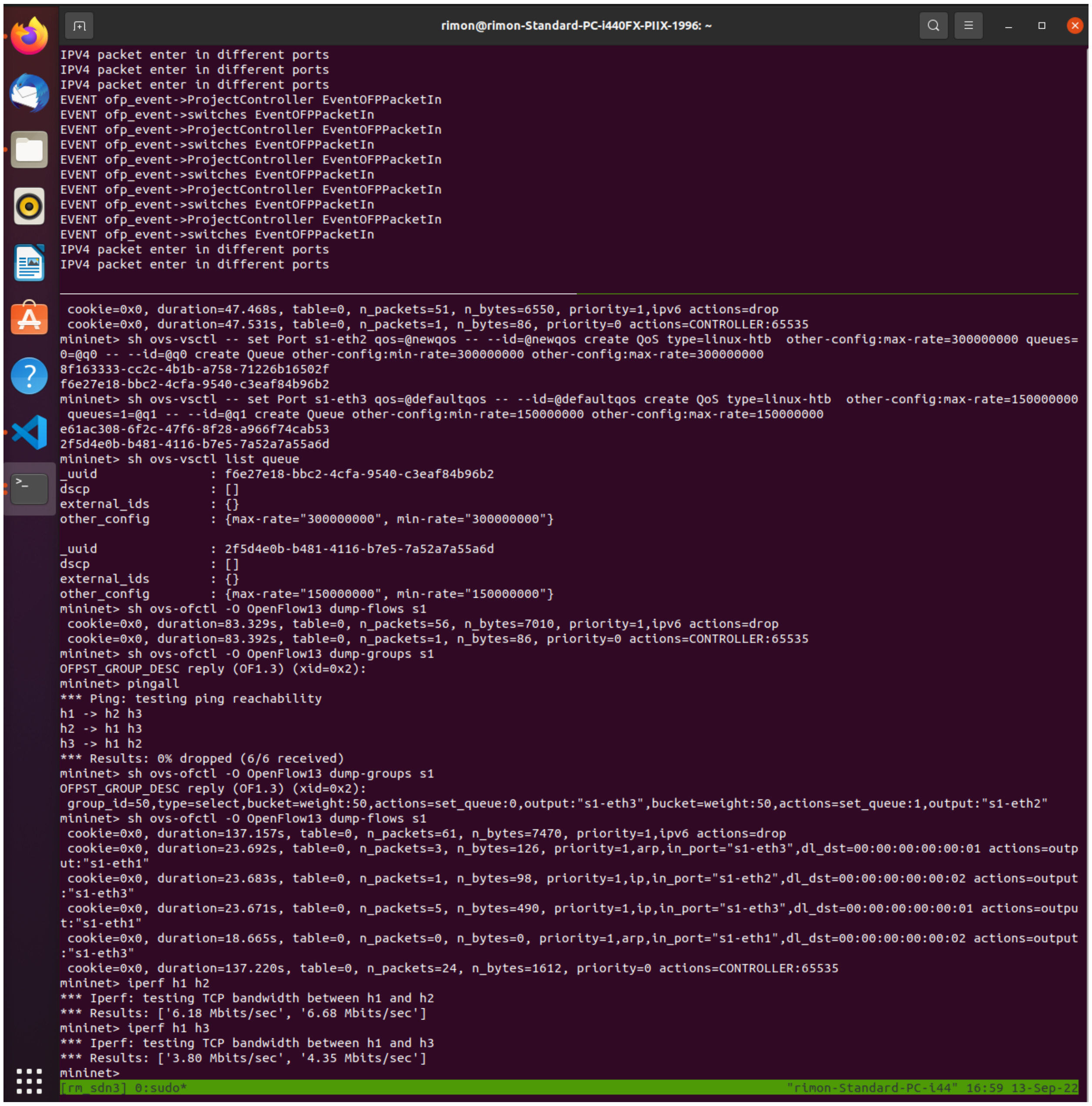
2. The results of the rule dump on the switch showing clearly the packets flowing through both paths, for the corresponding rules:



```
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> sh ovs-ofctl -O OpenFlow13 dump-groups s1
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=47.468s, table=0, n_packets=51, n_bytes=6550, priority=1,ipv6 actions=drop
 cookie=0x0, duration=47.531s, table=0, n_packets=1, n_bytes=86, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-vsctl -- set Port s1-eth2 qos=@newqos -- --id=@newqos create QoS type=linux-htb  other-config:max-rate=300000000 queues=
0=@q0 -- --id=@q0 create Queue other-config:min-rate=300000000 other-config:max-rate=300000000
8f163333-cc2c-4b1b-a758-71226b16502f
f6e27e18-bbc2-4cfa-9540-c3eaf84b96b2
mininet> sh ovs-vsctl -- set Port s1-eth3 qos=@defaultqos -- --id=@defaultqos create QoS type=linux-htb  other-config:max-rate=150000000
 queues=1=@q1 -- --id=@q1 create Queue other-config:min-rate=150000000 other-config:max-rate=150000000
e61ac308-6f2c-47f6-8f28-a966f74cab53
2f5d4e0b-b481-4116-b7e5-7a52a7a55a6d
mininet> sh ovs-vsctl list queue
_uuid               : f6e27e18-bbc2-4cfa-9540-c3eaf84b96b2
dscp                : []
external_ids        : {}
other_config        : {max-rate="300000000", min-rate="300000000"}

_uuid               : 2f5d4e0b-b481-4116-b7e5-7a52a7a55a6d
dscp                : []
external_ids        : {}
other_config        : {max-rate="150000000", min-rate="150000000"}
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=83.329s, table=0, n_packets=56, n_bytes=7010, priority=1,ipv6 actions=drop
 cookie=0x0, duration=83.392s, table=0, n_packets=1, n_bytes=86, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-ofctl -O OpenFlow13 dump-groups s1
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
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 -O OpenFlow13 dump-groups s1
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
 group_id=50,type=select,bucket=weight:50,actions=set_queue:0,output:"s1-eth3",bucket=weight:50,actions=set_queue:1,output:"s1-eth2"
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=137.157s, table=0, n_packets=61, n_bytes=7470, priority=1,ipv6 actions=drop
 cookie=0x0, duration=23.692s, table=0, n_packets=3, n_bytes=126, priority=1,arp,in_port="s1-eth3",dl_dst=00:00:00:00:00:01 actions=outp
ut:"s1-eth1"
 cookie=0x0, duration=23.683s, table=0, n_packets=1, n_bytes=98, priority=1,ip,in_port="s1-eth2",dl_dst=00:00:00:00:00:02 actions=output
:"s1-eth3"
 cookie=0x0, duration=23.671s, table=0, n_packets=5, n_bytes=490, priority=1,ip,in_port="s1-eth3",dl_dst=00:00:00:00:00:01 actions=outpu
t:"s1-eth1"
 cookie=0x0, duration=18.665s, table=0, n_packets=0, n_bytes=0, priority=1,arp,in_port="s1-eth1",dl_dst=00:00:00:00:00:02 actions=output
:"s1-eth3"
 cookie=0x0, duration=137.220s, table=0, n_packets=24, n_bytes=1612, priority=0 actions=CONTROLLER:65535
mininet> 
[rm_sdn3] 0:sudo*                                                                     "cimon-Standard-PC-i44" 16:57 13-Sep-22
```

3. Test runs of iperf verifying the max rate available for each path:



```
rimon@rimon-Standard-PC-i440FX-PIIX-1996: ~
IPV4 packet enter in different ports
IPV4 packet enter in different ports
IPV4 packet enter in different ports
EVENT ofp_event->ProjectController EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->ProjectController EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->ProjectController EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->ProjectController EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->ProjectController EventOFPPacketIn
EVENT ofp_event->switches EventOFPPacketIn
IPV4 packet enter in different ports
IPV4 packet enter in different ports

 cookie=0x0, duration=47.468s, table=0, n_packets=51, n_bytes=6550, priority=1,ipv6 actions=drop
 cookie=0x0, duration=47.531s, table=0, n_packets=1, n_bytes=86, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-vsctl -- set Port s1-eth2 qos=@newqos -- --id=@newqos create QoS type=linux-htb  other-config:max-rate=300000000 queues=
0=@q0 -- --id=@q0 create Queue other-config:min-rate=300000000 other-config:max-rate=300000000
8f163333-cc2c-4b1b-a758-71226b16502f
f6e27e18-bbc2-4cfa-9540-c3eaf84b96b2
mininet> sh ovs-vsctl -- set Port s1-eth3 qos=@defaultqos -- --id=@defaultqos create QoS type=linux-htb  other-config:max-rate=150000000
 queues=1=@q1 -- --id=@q1 create Queue other-config:min-rate=150000000 other-config:max-rate=150000000
e61ac308-6f2c-47f6-8f28-a966f74cab53
2f5d4e0b-b481-4116-b7e5-7a52a7a55a6d
mininet> sh ovs-vsctl list queue
_uuid               : f6e27e18-bbc2-4cfa-9540-c3eaf84b96b2
dscp                : []
external_ids        : {}
other_config        : {max-rate="300000000", min-rate="300000000"}

_uuid               : 2f5d4e0b-b481-4116-b7e5-7a52a7a55a6d
dscp                : []
external_ids        : {}
other_config        : {max-rate="150000000", min-rate="150000000"}
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=83.329s, table=0, n_packets=56, n_bytes=7010, priority=1,ipv6 actions=drop
 cookie=0x0, duration=83.392s, table=0, n_packets=1, n_bytes=86, priority=0 actions=CONTROLLER:65535
mininet> sh ovs-ofctl -O OpenFlow13 dump-groups s1
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
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 -O OpenFlow13 dump-groups s1
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
 group_id=50,type=select,bucket=weight:50,actions=set_queue:0,output:"s1-eth3",bucket=weight:50,actions=set_queue:1,output:"s1-eth2"
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
 cookie=0x0, duration=137.157s, table=0, n_packets=61, n_bytes=7470, priority=1,ipv6 actions=drop
 cookie=0x0, duration=23.692s, table=0, n_packets=3, n_bytes=126, priority=1,arp,in_port="s1-eth3",dl_dst=00:00:00:00:00:01 actions=outp
ut:"s1-eth1"
 cookie=0x0, duration=23.683s, table=0, n_packets=1, n_bytes=98, priority=1,ip,in_port="s1-eth2",dl_dst=00:00:00:00:00:02 actions=output
:"s1-eth3"
 cookie=0x0, duration=23.671s, table=0, n_packets=5, n_bytes=490, priority=1,ip,in_port="s1-eth3",dl_dst=00:00:00:00:00:01 actions=outpu
t:"s1-eth1"
 cookie=0x0, duration=18.665s, table=0, n_packets=0, n_bytes=0, priority=1,arp,in_port="s1-eth1",dl_dst=00:00:00:00:00:02 actions=output
:"s1-eth3"
 cookie=0x0, duration=137.220s, table=0, n_packets=24, n_bytes=1612, priority=0 actions=CONTROLLER:65535
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['6.18 Mbits/sec', '6.68 Mbits/sec']
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['3.80 Mbits/sec', '4.35 Mbits/sec']
mininet>
[rm_sdn3] 0:sudo*                                                      "rimon-Standard-PC-i44" 16:59 13-Sep-22
```

QUESTIONS:

- **Do you know what a decorator is? What is it used for in a Ryu Controller?**
  A decorator is a design pattern in Python that allows a user to add new functionality to an existing object without modifying its structure. Decorators are usually called before the definition of a function you want to decorate.

- **Why do you think we need the empty Match?**
  install table miss flow entry in case there is no rule in the switch to process the packet.

- **Why is it called "table-miss flow entry"?**
  should be the entry to process packetIN for the controller because where dont have a rule to match on the switch.

- **Why is the priority zero here?**
  miss entry should be the least priority to execute after all other rules are exhausted with higher priority.

- **What is the use of the flood? Why do we need the mac address of a device?**
  In case of miss we flood all other ports supposedly to check if we have an output port associated with it in the openflow table.

- **What is the ARP protocol?**
  Address Resolution Protocol (ARP) is a protocol or procedure that connects an ever-changing Internet Protocol (IP) address to a fixed physical machine address, also known as a media access control (MAC) address, in a local-area network (LAN).

- **What is the eth_dst parameter in the match?**
  Ethernet destination port for the mac address attached.

- **What other parameters could we try to match? What is a wildcard?**

```
/* Maximum number of physical and logical switch ports. */
OFPP_MAX = 0xffffff00,
/* Reserved OpenFlow Port (fake output "ports"). */
OFPP_IN_PORT = 0xfffffff8, /* Send the packet out the input port. This
reserved port must be explicitly used
in order to send back out of the input
port. */
OFPP_TABLE = 0xfffffff9, /* Submit the packet to the first flow table
NB: This destination port can only be
used in packet-out messages. */
OFPP_NORMAL = 0xfffffffa, /* Process with normal L2/L3 switching. */
OFPP_FLOOD = 0xfffffffb, /* All physical ports in VLAN, except input
port and those blocked or link down. */
OFPP_ALL = 0xfffffffc, /* All physical ports except input port. */
OFPP_CONTROLLER = 0xfffffffd, /* Send to controller. */
OFPP_LOCAL = 0xfffffffe, /* Local openflow "port". */
OFPP_ANY = 0xffffffff /* Wildcard port used only for flow mod
(delete) and flow stats requests. Selects
all flows regardless of output port
(including flows with no output port). */
```

- **Why do we need to obtain the information for four different protocols?**
  You can create a Packet class instance with the received raw data. Then the packet library parses the data and creates

protocol class instances included the data. The packet class ’protocols’ has the protocol class instances.

- **What do you think this decorator could be useful for?**
  This decorator tells Ryu when the decorated function
  should be called.
  The first argument of the decorator indicates which type of event this function should be called for. As you might
  expect, every time Ryu gets a packet_in message, this function is called.
  The second argument indicates the state of the switch. You probably want to ignore packet_in messages before the
  negotiation between Ryu and the switch is finished. Using ’MAIN_DISPATCHER’ as the second argument means
  this function is called only after the negotiation completes.
  they are useful for adding functionality while handling events
  set_ev_cls specifies the event class supporting the received message the state of the openflow switch argument

- **Why do we need to create groups and flows in different steps?**
  Flow is to add rules to switches, but groups are to create buckets and perform advanced actions like mirroring and load balancing on the packets by assigning buckets.

- **Where is this group defined?**
  It’s created on the switch by the function send_group_mod()

- **Why do we need the following special cases?**
  Because we are load balancing on the sessions coming from the host on port 1 on switch 1.

- **Why do we need to obtain the information for four different protocols?**
  We need to separate packets depending on the protocol to facilitate topology discovery and separate functionality depending on protocol layer

- **What is the event here and what is the difference with the CONFIG_DISPATCHER in the previous function?**
  EventOFPPacketIn which is packet_in to send the packet to the controller to process it since it’s a miss. The difference is that CONFIG_DISPATCHER is for negotiation of the openflow protocol and getting switch features while this event is for normal connection flow after negotiation success.

Notes from TA Hours for project 1:

- https://buildmedia.readthedocs.org/media/pdf/ryu/latest/ryu.pdf
- eth-> 64.65
- ip4-> 73
- LLDP 79
- Of 83
- Udp 103

- Vlan 104
- Bridge is 135
- OpenFLow Messages v1.3 → 213
- Flow match structure 208
- Shortest path is based on number of hops..
- Igraph, networkx library has shortest path
- gethost(), getswitch and getlink()
- Recalculate based on every certain t time or if topology changes
- Synchronization handling of events
- Install all sessions at once on switches at one time.. U can add many flow entries to as many switches as need per packet_in_handler() on as many switches.. flowmod()
- You should all the dpid for all switches when you scan the topology
- The same packet ending in the same switch.. U will receive duplicates
- TODO questions in workshop3 all are mandatory
- ARP PACKET STORM.. To compensate to use STP algorithm we need to avoid arp storms!!!
    - Install arp flow table entries and help u alot with all
    - right, STP as implemented in RYU will mess with your path algo
    - Event host add() to completely avoid using arp
    - U have to report something reply pack to keep sending arp packet and then formulate ur arp packet.. So instead of flooding sent ur own arp packet back
    - The hosts arent aware that the ryu control.. So when the port comes alive but u need to know what is that port connected to a host or another switch
    - So avoid arp as much as u can
    - Write ur own widespath algorithms python.. Shortest path doesnt work