SDN Tutorial

SDN Tutorial

- Topics:
- Prerequisites

1. Create topology, Modify topology
2. Simple switch, Add flow
3. Packet-in handler, Packet-out
4. Packet filter in packet-in handler
5. Add actions to push, pop MPLS labels
6. Multiple tables

Prerequisites

- Required tools and versions:
  - Open vSwitch 2.3.0
  - Mininet 2.1.0p
  - Ryu 3.17
- Install:
  - Please follow the instruction in files under SDN_Tutorial/Installation to install the OpenvSwitch, Mininet, and Ryu

Terminology

Terminology	Open vSwitch	OpenFlow	Ryu
Switch	Bridge	OpenFlow Switch	Datapath
Port	Port/Interface	OpenFlow Port	Port
Flow Entry	Flow/Flow Entry	Flow Entry	Flow Entry

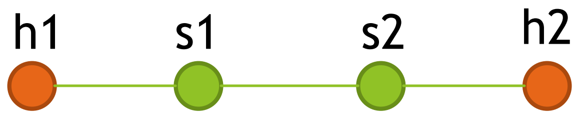
Task 1

- Topics:
  Create topology, Modify topology

- Using tool:
  Mininet

Task1 – Mininet Custom Topology

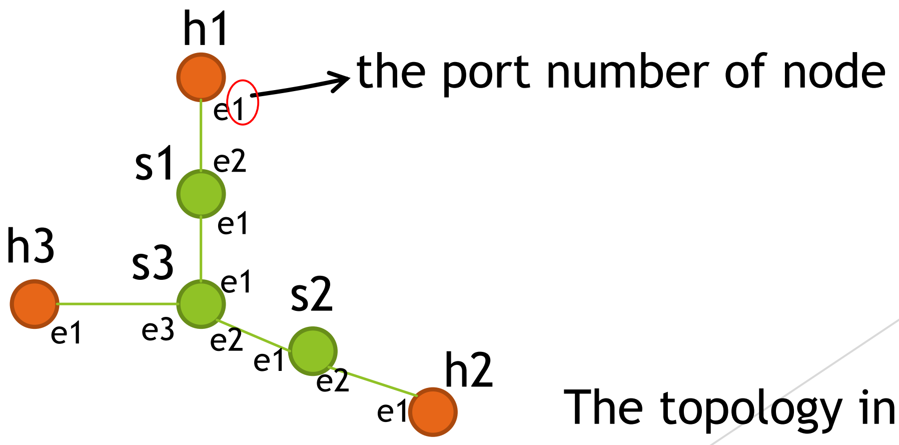
- topo-2sw-2host.py (by mininet)
  - The source code is in Mininet's source tree: mininet/custom/topo-2sw-2host.py
- To start up a mininet with the custom topology
  - sudo mn --custom topo-2sw-2host.py --topo mytopo --controller remote



  - Use 'exit' to close the mininet

Task1 – Read topology from JSON and create it

- mytopo.py (in Exercise_1)
  - A sample code modified from mininet custom topology
  - Read topology description from json file then use custom topology to create it
- Run mytopo.py:
  - sudo mn --custom mytopo.py --topo mytopo --switch ovsk --controller remote --mac --link tc



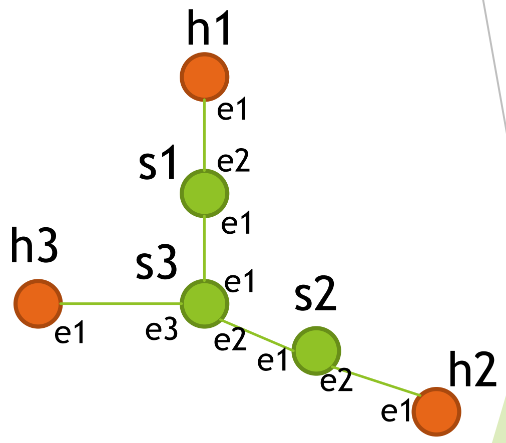
The topology in sample file

Task1 – Options in mininet command (mn)

- Switch type:
  - --switch=ovsk
    - Use openvswitch kernel switch
  - --switch=user
    - Use openvswitch user-space switch
  - --custom=CUSTOM
    - Read custom topo and node params from .pyfile
  - --mac
    - Automatically set host MACs for hosts
  - --controller=remote
    - Connect to remote controller (Default IP is 127.0.0.1)
  - --link tc
    - Use tc link in mininet (Set bandwidth, delay, jitter, loss… for links)
- Type “mn –help” to see more

Task1 – Topology JSON file

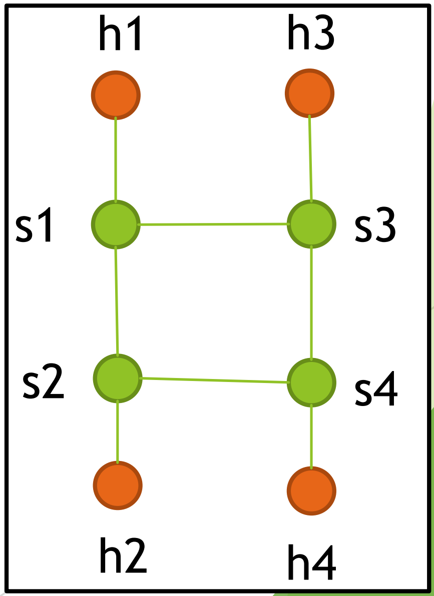
- Define “node” and “link” in topology
- node:
  - The node name in mininet
- link:
  - “src”: the src node of this link
  - “dst”: the dst node of this link
  - “bw”: the bandwidth of this link
  - “delay”: the delay value of this link
  - “p1”: the port number of src node
  - “p2”: the port number of dst node



```
{"node": [ "s1", "s2", "s3", "h1", "h2", "h3"],
 "link": [
        {"src": "h1", "dst": "s1", "bw": 10, "delay": "100ms", "p1": 1, "p2": 2},
        {"src": "h2", "dst": "s2", "bw": 10, "delay": "100ms", "p1": 1, "p2": 2},
        {"src": "h3", "dst": "s3", "bw": 10, "delay": "100ms", "p1": 1, "p2": 3},
        {"src": "s1", "dst": "s3", "bw": 10, "delay": "100ms", "p1": 1, "p2": 1},
        {"src": "s2", "dst": "s3", "bw": 10, "delay": "100ms", "p1": 1, "p2": 2}
                ]}
```

Exercise1 – Modify JSON file and create a new topology

- Modify topo_sample.json to create a new topology like the following picture
- Assign the port number for each interface
- Type ‘net’ in mininet to show your topology



Task 2

- Topics:
  Simple switch, Add flow

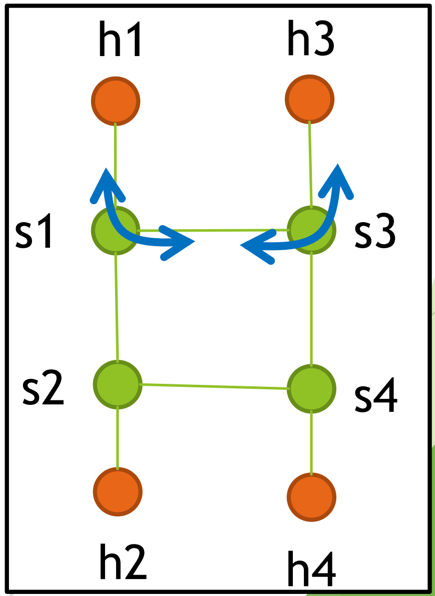
- Using tools:
  Ryu, tcpdump(wireshark), ovs-vsctl & ovs-ofctl

Task2 – Simple switch in Ryu

- Material:
  - The switch hub implementation tutorial in Ryu book (Please read it and follow the execution steps.) http://osrg.github.io/ryu-book/en/html/switching_hub.html#id1
- The source code of simple switch is in Ryu's source tree: ryu/app/simple_switch_13.py

Exercise2 – Add flow

- Delete the function "_packet_in_handler" and modify the function "switch_features_handler" in simple_switch_13.py (start from exercise2.py)
- After finishing the installation of table-miss flow on switches, add flow-entries to build up the communications between h1 and h3 in exercise1's topology.

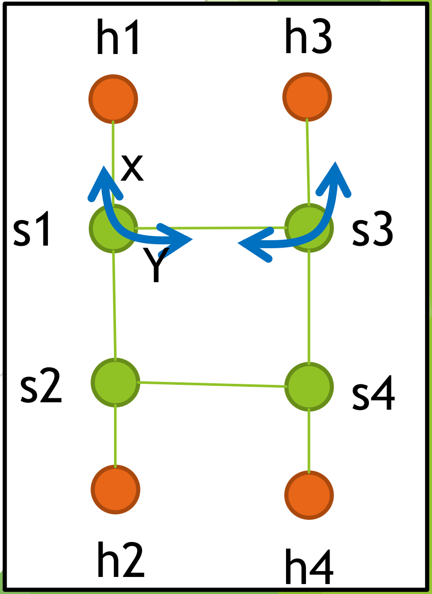


Exercise2 - Hints

- Hint1: You need to add 2 flow-entries on s1 and s3
  - Ex: (for s1)
    - Flow1: Forward packets from port X to port Y
    - Flow2: Forward packets from port Y to port X
- Hint2: Flow's attributes
  match: inport, actions: outport, priority=1

```
ex:
match = parser.OFPMatch(in_port=2)
actions = [parser.OFPActionOutput(1)]
self.add_flow(datapath, 1, match, actions)
```

  - (priority=0 is table-miss flow)
- Hint3: Use datapath.id to get switch index



Task 3

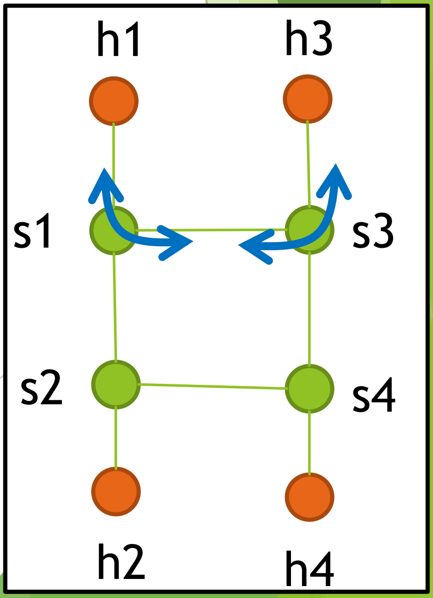
- Topics:
  Packet-in handler, Packet-out

- Using tool:
  Ryu

Exercise3 – Packet-in handler

- Modify the function "_packet_in_handler" in simple_switch_13.py (start from exercise3.py)

- Use topology in exercise1.
  Similar to exercise2, use packet-out to forward packets between h1 and h3.

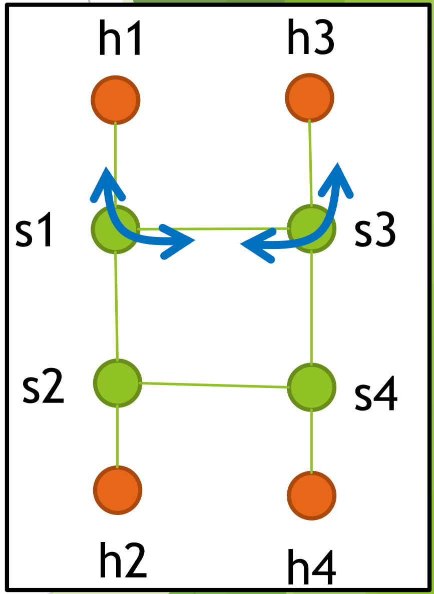


Exercise3 - Hints

- Hint1: Add "--arp" when running mininet to let the hosts get the mac addresses of other hosts

- Hint2: Use mac address dst (destination) to decide the outport of switch

```
ex:
if dst == '00:00:00:00:00:01':
    actions = [parser.OFPActionOutput(1)]
    out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                              in_port=in_port, actions=actions, data=data)
    datapath.send_msg(out)
```



Task 4

- Topic:
  Add filter in packet-in handler

- Using tool:
  Ryu

Exercise4 – Packet-in filter

- Modify the function “_packet_in_handler” in simple_switch_13.py (start from exercise4.py)
- Use ethertype to identify different types of captured packets and print their src and dst (IP packets and ARP packets)
  - Material: Ryu Packet library API Reference http://ryu.readthedocs.org/en/latest/library_packet_ref.html
- Exercise4(a): Do not use ‘--arp’ in mininet to automatically learn mac addresses on hosts, and use ping to generate ARP packets
- Exercise4(b): Use ‘--arp’ in mininet to automatically learn mac addresses on hosts, and use ping to gernerate ICMP packets (IP type)

Exercise4 - Hints

- Hint1: You need the library in Ryu's source code

```
from ryu.ofproto.ether import ETH_TYPE_IP, ETH_TYPE_ARP
from ryu.lib.pacekt import ipv4, arp
```

  - Find the required field of protocols in the library

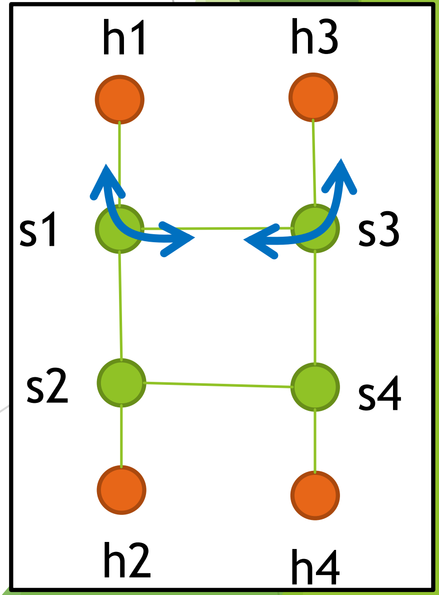
Task 5

- Topic:
  Add actions to push, pop MPLS labels

- Using tool:
  Ryu

Exercise5 – Add MPLS actions

- Delete the function "_packet_in_handler" and modify the function "switch_features_handler" in simple_switch_13.py (start from exercise5.py)
- After finishing the installation of table-miss flow on switches, add flow-entries to build up the communications between h1 and h3 in exercise1's topology.
- Similar to exercise2, but use MPLS label to distinguish packets between switches.
    - Push and pop MPLS label on edge switch
- Use user-space switch(in ovs, kernel switch do not support MPLS)
    - sudo mn --custom mytopo.py --topo mytopo --switch ovsk,datapath=user --controller remote --mac --link tc



Exercise5 - Hints

- Hint1: Required match and actions
  - Push:

```
match = parser.OFPMatch(eth_type=ETH_TYPE_IP,
                                              eth_dst='00:00:00:00:00:03')
actions = [datapath.ofproto_parser.OFPActionPushMpls(ETH_TYPE_MPLS),
              datapath.ofproto_parser.OFPActionSetField(mpls_label=10),
              datapath.ofproto_parser.OFPActionOutput(2)]
```

  - Pop:

```
match = parser.OFPMatch(eth_type=ETH_TYPE_MPLS,
                                    mpls_label=30)
actions = [datapath.ofproto_parser.OFPActionPopMpls(ETH_TYPE_IP),
              datapath.ofproto_parser.OFPActionOutput(1)]
```

Task 6

- Topic:
  Multiple tables

- Using tool:
  Ryu

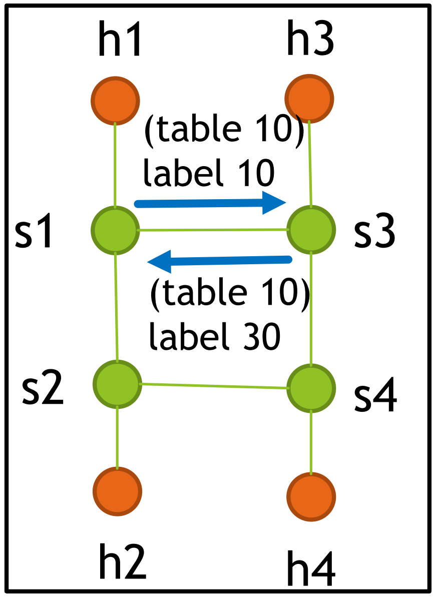
Task6 – Multiple tables

- Use multiple tables to implement pre-build flows of MPLS path
- Find some paths beforehand and assign them with MPLS labels.
  When the traffic comes, add flows to match certain MAC addressed on edge switches.
- Use instruction to adjust actions

Exercise6 – Pre-build flows

1. After finishing the installation of table-miss flow on switches, add flows to match certain MPLS labels.

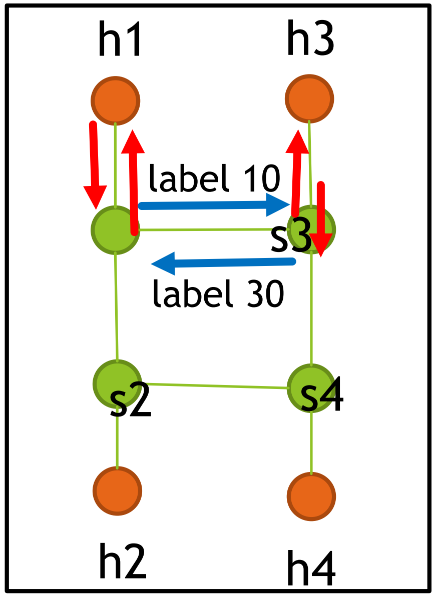
- Add flows: (ex: for the path from s1 to s3)
  - (on s1): table 10, match: label 10, actions: output, instruction: apply actions
  - (on s3): table 0, match: label 10, actions: pop mpls label, instruction: apply actions, goto table 10



Exercise6 – Traffic comes

2. When switches get ARP or IP packets from h1 or h3 to each others, add flows to push MPLS label

- Add flows: (ex: for the path from s1 to s3)
  - (on s1): table 0, match: dst:0x03, actions: push label 10, instruction: apply actions, goto table 10
  - (on s3): table 10, match: dst:0x03, actions: outport 1, instruction: apply actions



Exercise6 - Hints

- Hint1: for flow with lower table-id, use instructions to go to another table
  Material: https://github.com/osrg/ryu-book/blob/master/en/source/openflow_protocol.rst

```
ex:
instruction = [parser.OFPInstructionActions(datapath.ofproto.OFPIT_APPLY_ACTIONS, actions),
               parser.OFPInstructionGotoTable(10)]
```

- Hint2: Add parameter of table_id and instruction in function “add_flow”

```
def add_flow(self, datapath, table_id, priority, match, instruction, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    inst = instruction
    mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                            match=match, table_id=table_id, instructions=inst)
    datapath.send_msg(mod)
```

Exercise6 - Hints

- Hint3: Use self.dpset to get datapath structure in Ryu
  - self.dpset = { dpid: datapath structure of dpid }
  - ex: to get dpid 1

```
datapath = self.dpset[1]
```