COMN - Computer Communications and Networks
Coursework Introduction: SDN

Michio Honda

School of Informatics

University of Edinburgh

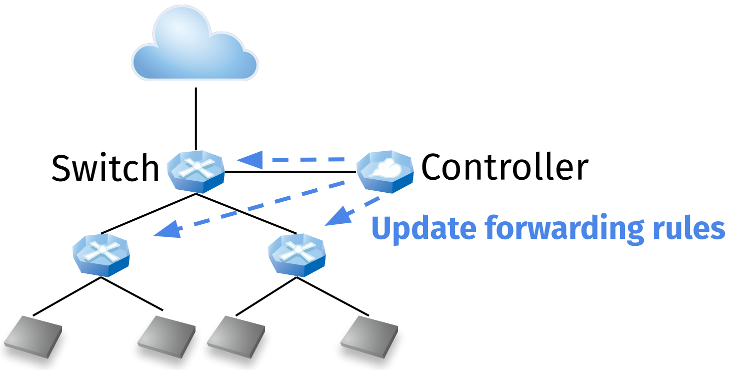
14/03/2025

Text version is also available at

https://git.ecdf.ed.ac.uk/mhonda/comn25cw/blob/master/sdn/introduction.md

SDN and OpenFlow

- SDN
  - Centralised network architecture
  - Decomple *decision making* from *forwarding action*
  - Control plane
    - Policy and management
  - Data (forwarding) plane
    - Packet forwarding



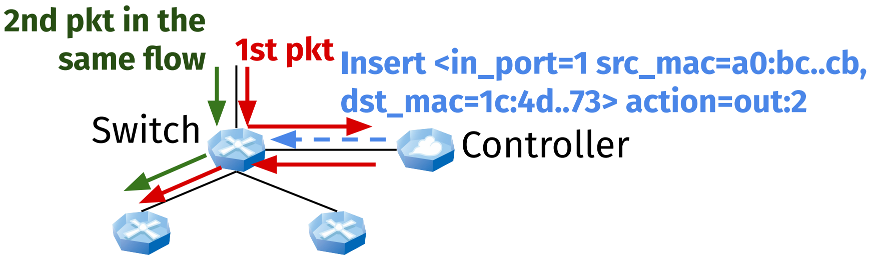
- OpenFlow
  - De facto protocol to achieve SDN

What is a Flow?

- A group of packets in various meaningful form, like
  - Packets with the same source-destination MAC address pair
  - TCP packets with the same four tuple (<src_ip><dst_ip><src_port><dst_port>)

OpenFlow in Action

- A switch forwards packets that do not match any existing flows to the controller
- The controller makes decision what to do for this packet, like
  - Drop
  - Insert the flow to the switch so that the packets that match this flow do not go to the controller anymore



Mininet and Ryu

- Mininet
  - A network emulator that runs in a single (virtual) machine
- Ryu (‘s fork, OS-Ken)
  - Python framework to implement OpenFlow controllers
- Both are pre-installed in our VM

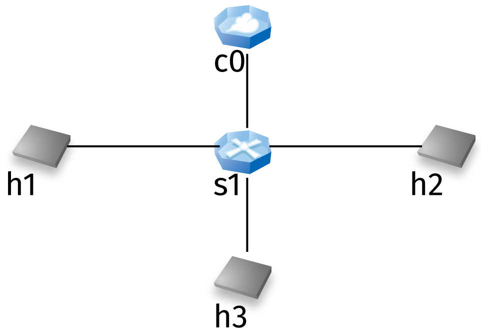
Running Ryu and Mininet

(In the VM)

```
vagrant@ubuntu-jammy:/vagrant/comn25cw/sdn$ osken-manager l2learn.py
loading app l2learn.py
loading app os_ken.controller.ofp_handler
instantiating app l2learn.py of L2Learn14
instantiating app os_ken.controller.ofp_handler of OFPHandler
```

(In another window)

```
vagrant@ubuntu-jammy:/vagrant/comn25cw/sdn$ sudo mn --topo single,3 --mac --controller remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```



Executing a UNIX command

- Executing a UNIX command

```
mininet> h2 ping -c 1 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.175 ms

--- 10.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.175/0.175/0.175/0.000 ms
```

- Some commands do not need to specify the host

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Understanding controler behavior

(stop the OS-Ken controller (ctrl+c) and exit from Mininet (type “exit”)

- Let’s put a print() in the function called every time the controller receives a packet

```
vagrant@ubuntu-jammy:/vagrant/comn25cw/sdn$ vi l2learn.py
...
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        print(in_port, pkt) # add this line
        msg = ev.msg
...
```

- Then run OS-Ken controller and Mininet again, and do `pingall`
  - You will see packets received at the ryu-controller output

Understanding L2 Learning switch in Ryu

`(l2learn.py)`

```
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
1       msg = ev.msg
2       in_port, pkt = (msg.match['in_port'], packet.Packet(msg.data))
3       dp = msg.datapath
4       ofp, psr, did = (dp.ofproto, dp.ofproto_parser, format(dp.id, '016d'))
5       eth = pkt.get_protocols(ethernet.ethernet)[0]
6       dst, src = (eth.dst, eth.src)
7       self.ht.setdefault(did, {})
8       he = self.ht[did] # shorthand
9       he[src] = in_port
10      out_port = he[dst] if dst in he else ofp.OFPP_FLOOD
```

- Line 9: First create a hash table entry for the source MAC
- Line 10: Decide the output port(s) based on the existence of the destination MAC address entry

Understanding L2 Learning switch in Ryu

```
(l2learn.py)
...
11        acts = [psr.OFPActionOutput(out_port)]
12        if out_port != ofp.OFPP_FLOOD:
13            mtc = psr.OFPMatch(in_port=in_port, eth_dst=dst, eth_src=src)
14            self.add_flow(dp, 1, mtc, acts, msg.buffer_id)
15            if msg.buffer_id != ofp.OFP_NO_BUFFER:
16                return
17        data = msg.data if msg.buffer_id == ofp.OFP_NO_BUFFER else None
18        out = psr.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
19                               in_port=in_port, actions=acts, data=data)
20        dp.send_msg(out)
```

- Line 11: Generate a list of action list (just output in this case)
- Line 12-: If the correct output port has been identified (Line 12), insert the flow entry (matches to src/dst MAC addresses and input port, as seen in Line 13) to the switch (Line 14). Then the controller applies the packet action (Line 18-20)

Summary

- We learned Mininet and Ryu
- SDN is a popular networking research topic:
  - Flow management technique [1]
  - Controller fault Tolerance [2]
  - Dataplane abstraction [3]
  - Host switch [4]
- Many interesting papers at ACM SOSR and SIGCOMM, and USENIX NSDI

[1] https://conferences.sigcomm.org/sigcomm/2011/papers/sigcomm/p254.pdf
[2] https://www.cs.princeton.edu/~mfreed/docs/ravana-sosr15.pdf
[3] https://www.sigcomm.org/sites/default/files/ccr/papers/2014/July/0000000-0000004.pdf
[4] https://www.usenix.org/system/files/conference/nsdi17/nsdi17-firestone.pdf