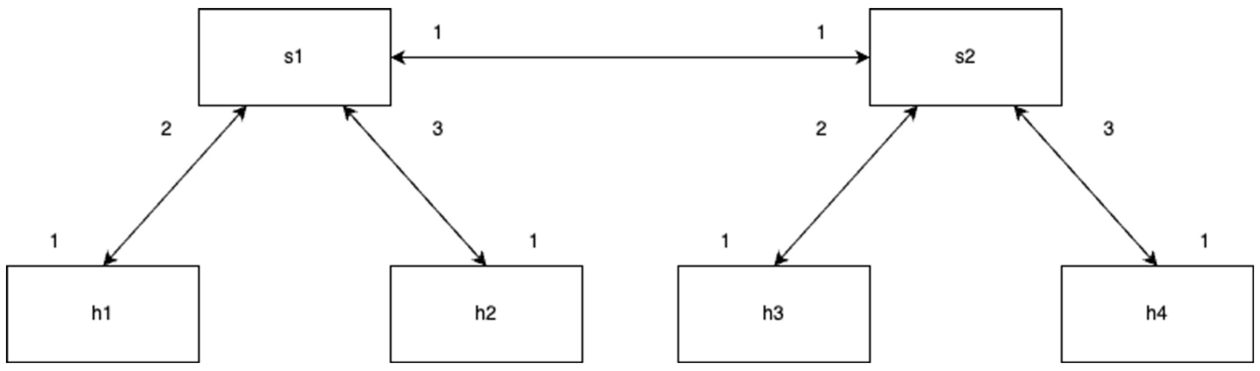
Homework 2

Question 1

Assume that the network topology is as follows:



s1 and s2 are switches, while h1 - h4 are hosts. The numbers represent the physical ports for the links. Complete the routing tables for s1 and s2. All hosts should be able to ping each other.

Let the IP address of h1 be '10.0.0.1', h2 be ‘10.0.0.2’, h3 be ‘10.0.0.3’, and h4 be ‘10.0.0.4’, then we have

The routing table for s1:

Match	Action
Ip=10.0.0.1, protocol=icmp	Output to 2
Ip=10.0.0.2, protocol=icmp	Output to 3
Ip=10.0.0.3, protocol=icmp	Output to 1
Ip=10.0.0.4, protocol=icmp	Output to 1

The routing table for s2:

Match	Action
Ip=10.0.0.1, protocol=icmp	Output to 1
Ip=10.0.0.2, protocol=icmp	Output to 1
Ip=10.0.0.3, protocol=icmp	Output to 2
Ip=10.0.0.4, protocol=icmp	Output to 3

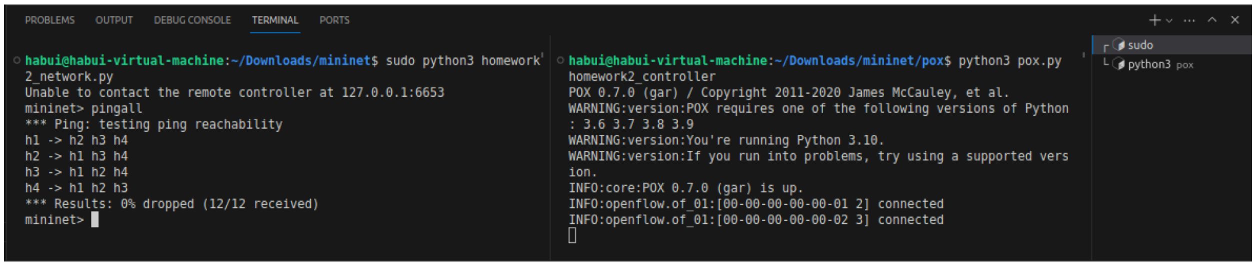
Question 2

Implement a remote network controller using the POX library for the network mentioned in Question 1. Submit all your code including mininet network and controller along with a screenshot of the **pingall** command execution result. All nodes should be able to ping each other.

Hint

1. You can distinguish different switch with the variable: `event.dpid`. When it is 1, it is s1. Otherwise, it is s2.
2. When you add link between nodes, you can decide which port to use.
3. You could use virtual port provided by POX library.

screenshot of the “pingall” command



```
habui@habui-virtual-machine:~/Downloads/mininet$ sudo python3 homework
2_network.py
Unable to contact the remote controller at 127.0.0.1:6653
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

```
habui@habui-virtual-machine:~/Downloads/mininet/pox$ python3 pox.py
homework2_controller
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:POX requires one of the following versions of Python
: 3.6 3.7 3.8 3.9
WARNING:version:You're running Python 3.10.
WARNING:version:If you run into problems, try using a supported vers
ion.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
```

homework2_network.py

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.node import RemoteController

class Homework2Network(Topo):
    def build(self):
        switch1 = self.addSwitch('s1')
        switch2 = self.addSwitch('s2')
        host1 = self.addHost('h1', ip = '10.0.0.1')
        host2 = self.addHost('h2', ip = '10.0.0.2')
        host3 = self.addHost('h3', ip = '10.0.0.3')
        host4 = self.addHost('h4', ip = '10.0.0.4')

        self.addLink(switch1, switch2, port1 = 1, port2 = 1)
        self.addLink(switch1, host1, port1 = 2, port2 = 1)
        self.addLink(switch1, host2, port1 = 3, port2 = 1)
        self.addLink(switch2, host3, port1 = 2, port2 = 1)
        self.addLink(switch2, host4, port1 = 3, port2 = 1)

if __name__ == '__main__':
    net = Mininet(Homework2Network(), controller = RemoteController)
    net.start()
    CLI(net)
    net.stop()
```

pox/homework2_controller.py

```
from pox.core import core
import pox.lib.packet as pkt
import pox.openflow.libopenflow_01 as of

class Homework2Controller:
    def __init__(self) -> None:
        core.openflow.addListeners(self)

    def _handle_ConnectionUp(self, event):
        connection = event.connection
        if event.dpid == 1:
            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.1'),
                    action = of.ofp_action_output(port = 2),
                )
            )

            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.2'),
                    action = of.ofp_action_output(port = 3),
                )
            )

            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.3'),
                    action = of.ofp_action_output(port = 1),
                )
            )

            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.4'),
                    action = of.ofp_action_output(port = 1),
                )
            )

        else:
            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.1'),
                    action = of.ofp_action_output(port = 1),
                )
            )

            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.2'),
                    action = of.ofp_action_output(port = 1),
                )
            )

            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.3'),
                    action = of.ofp_action_output(port = 2),
                )
            )

            connection.send(
                of.ofp_flow_mod(
                    match = of.ofp_match(dl_type=pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.ICMP_PROTOCOL, nw_dst = '10.0.0.4'),
                    action = of.ofp_action_output(port = 3),
                )
            )

        connection.send(
            of.ofp_flow_mod(
                match = of.ofp_match(dl_type=pkt.ethernet.ARP_TYPE),
                action = of.ofp_action_output(port = 65531),
            )
        )

def launch():
    core.registerNew(Homework2Controller)
```