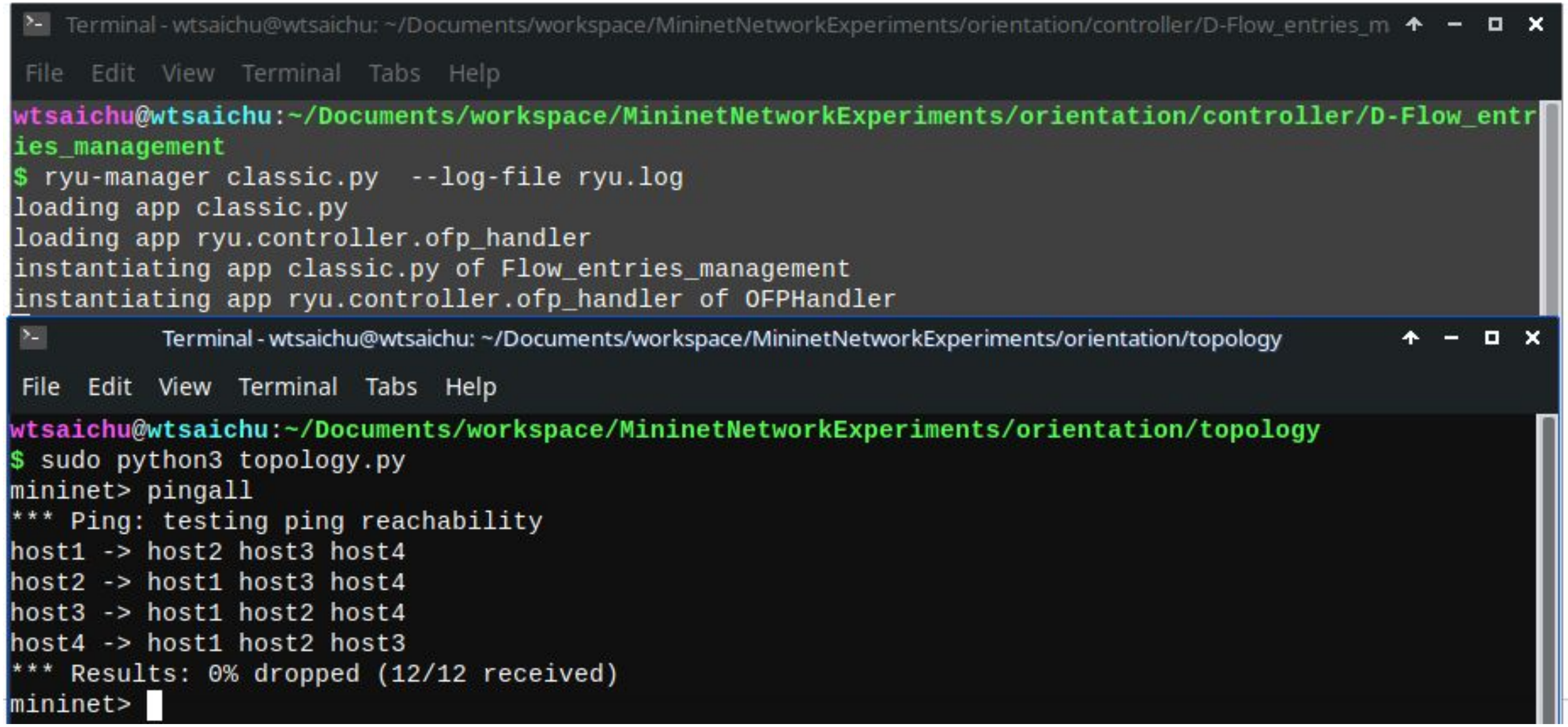
Flow entries management

Consequent

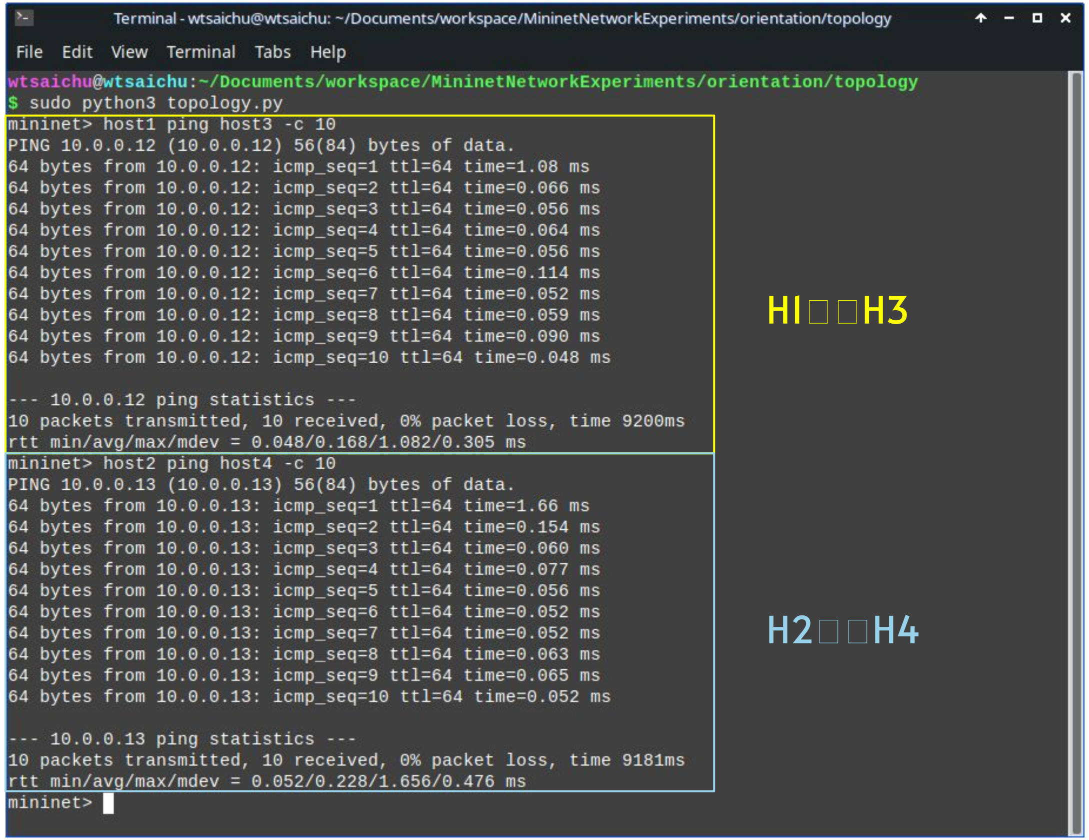
Ping test



```
Terminal - wtsaichu@wtsaichu: ~/Documents/workspace/MininetNetworkExperiments/orientation/controller/D-Flow_entries_m
File  Edit  View  Terminal  Tabs  Help
wtsaichu@wtsaichu:~/Documents/workspace/MininetNetworkExperiments/orientation/controller/D-Flow_entr
ies_management
$ ryu-manager classic.py  --log-file ryu.log
loading app classic.py
loading app ryu.controller.ofp_handler
instantiating app classic.py of Flow_entries_management
instantiating app ryu.controller.ofp_handler of OFPHandler
```

```
Terminal - wtsaichu@wtsaichu: ~/Documents/workspace/MininetNetworkExperiments/orientation/topology
File  Edit  View  Terminal  Tabs  Help
wtsaichu@wtsaichu:~/Documents/workspace/MininetNetworkExperiments/orientation/topology
$ sudo python3 topology.py
mininet> pingall
*** Ping: testing ping reachability
host1 -> host2 host3 host4
host2 -> host1 host3 host4
host3 -> host1 host2 host4
host4 -> host1 host2 host3
*** Results: 0% dropped (12/12 received)
mininet> 
```

Ping test

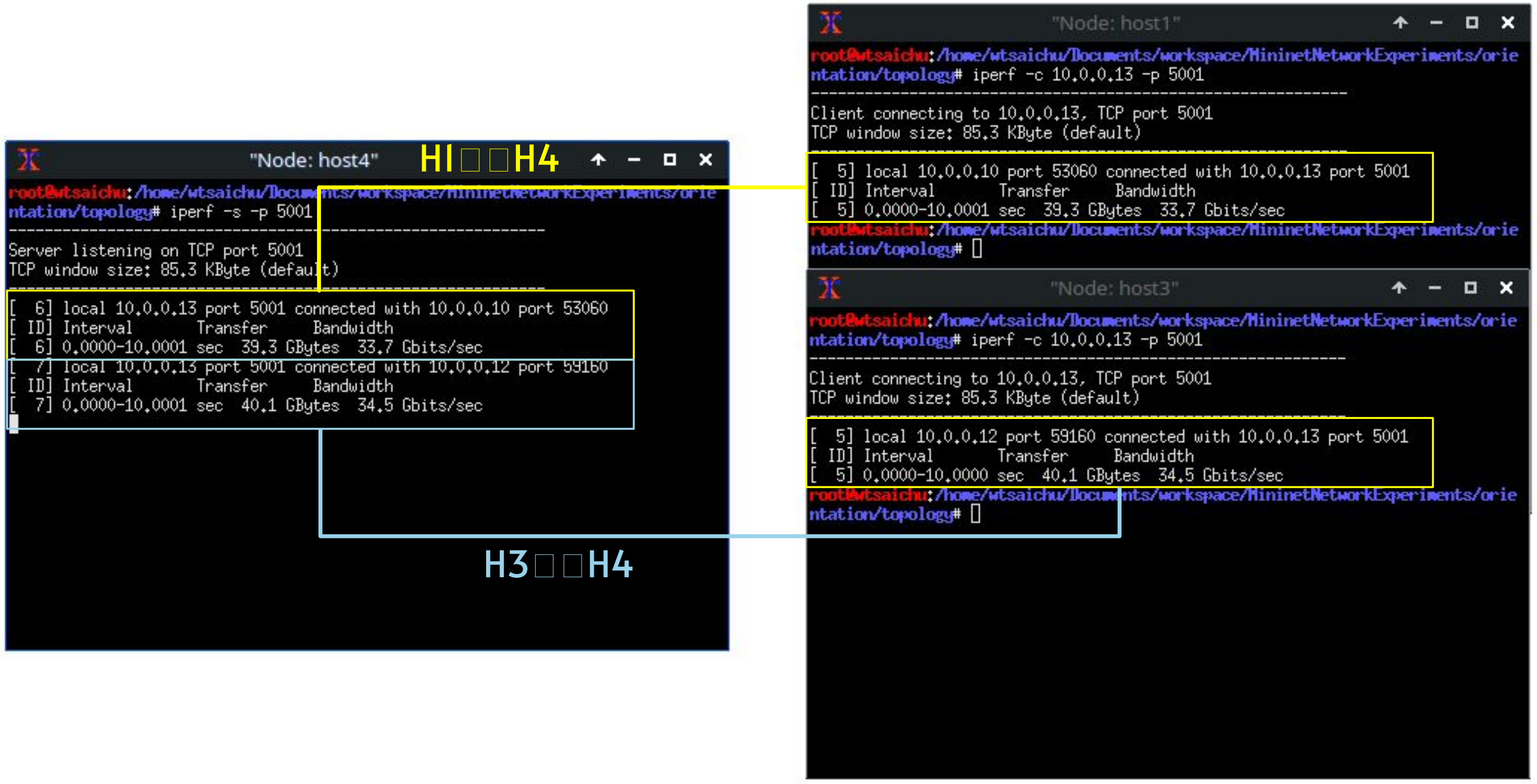

Terminal - wtsaichu@wtsaichu: ~/Documents/workspace/MininetNetworkExperiments/orientation/topology
File Edit View Terminal Tabs Help

```
wtsaichu@wtsaichu:~/Documents/workspace/MininetNetworkExperiments/orientation/topology
$ sudo python3 topology.py
mininet> host1 ping host3 -c 10
PING 10.0.0.12 (10.0.0.12) 56(84) bytes of data.
64 bytes from 10.0.0.12: icmp_seq=1 ttl=64 time=1.08 ms
64 bytes from 10.0.0.12: icmp_seq=2 ttl=64 time=0.066 ms
64 bytes from 10.0.0.12: icmp_seq=3 ttl=64 time=0.056 ms
64 bytes from 10.0.0.12: icmp_seq=4 ttl=64 time=0.064 ms
64 bytes from 10.0.0.12: icmp_seq=5 ttl=64 time=0.056 ms
64 bytes from 10.0.0.12: icmp_seq=6 ttl=64 time=0.114 ms
64 bytes from 10.0.0.12: icmp_seq=7 ttl=64 time=0.052 ms
64 bytes from 10.0.0.12: icmp_seq=8 ttl=64 time=0.059 ms
64 bytes from 10.0.0.12: icmp_seq=9 ttl=64 time=0.090 ms
64 bytes from 10.0.0.12: icmp_seq=10 ttl=64 time=0.048 ms

--- 10.0.0.12 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9200ms
rtt min/avg/max/mdev = 0.048/0.168/1.082/0.305 ms
mininet> host2 ping host4 -c 10
PING 10.0.0.13 (10.0.0.13) 56(84) bytes of data.
64 bytes from 10.0.0.13: icmp_seq=1 ttl=64 time=1.66 ms
64 bytes from 10.0.0.13: icmp_seq=2 ttl=64 time=0.154 ms
64 bytes from 10.0.0.13: icmp_seq=3 ttl=64 time=0.060 ms
64 bytes from 10.0.0.13: icmp_seq=4 ttl=64 time=0.077 ms
64 bytes from 10.0.0.13: icmp_seq=5 ttl=64 time=0.056 ms
64 bytes from 10.0.0.13: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 10.0.0.13: icmp_seq=7 ttl=64 time=0.052 ms
64 bytes from 10.0.0.13: icmp_seq=8 ttl=64 time=0.063 ms
64 bytes from 10.0.0.13: icmp_seq=9 ttl=64 time=0.065 ms
64 bytes from 10.0.0.13: icmp_seq=10 ttl=64 time=0.052 ms

--- 10.0.0.13 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9181ms
rtt min/avg/max/mdev = 0.052/0.228/1.656/0.476 ms
mininet>
```

H1□□H3

H2□□H4

IPERF test

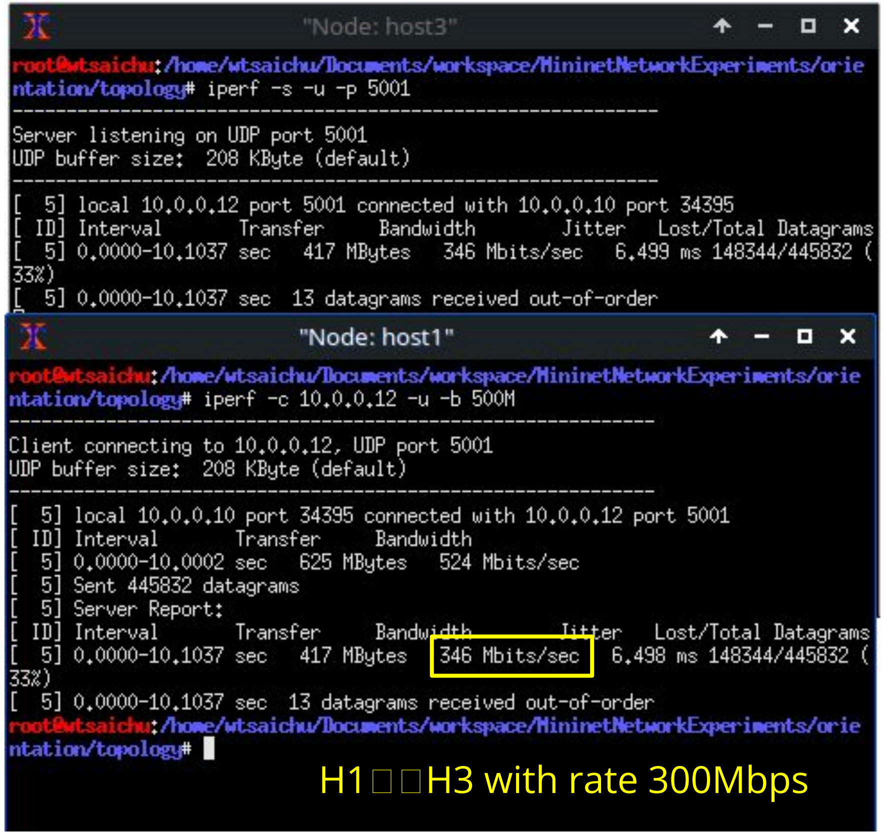


```
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -s -p 5001
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.13 port 5001 connected with 10.0.0.10 port 53060
[ ID] Interval       Transfer     Bandwidth
[  6] 0.0000-10.0001 sec  39.3 GBytes  33.7 Gbits/sec
[  7] local 10.0.0.13 port 5001 connected with 10.0.0.12 port 59160
[ ID] Interval       Transfer     Bandwidth
[  7] 0.0000-10.0001 sec  40.1 GBytes  34.5 Gbits/sec
```

```
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -c 10.0.0.13 -p 5001
------------------------------------------------------------
Client connecting to 10.0.0.13, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.10 port 53060 connected with 10.0.0.13 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5] 0.0000-10.0001 sec  39.3 GBytes  33.7 Gbits/sec
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# 
```

```
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -c 10.0.0.13 -p 5001
------------------------------------------------------------
Client connecting to 10.0.0.13, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.12 port 59160 connected with 10.0.0.13 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5] 0.0000-10.0000 sec  40.1 GBytes  34.5 Gbits/sec
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# 
```

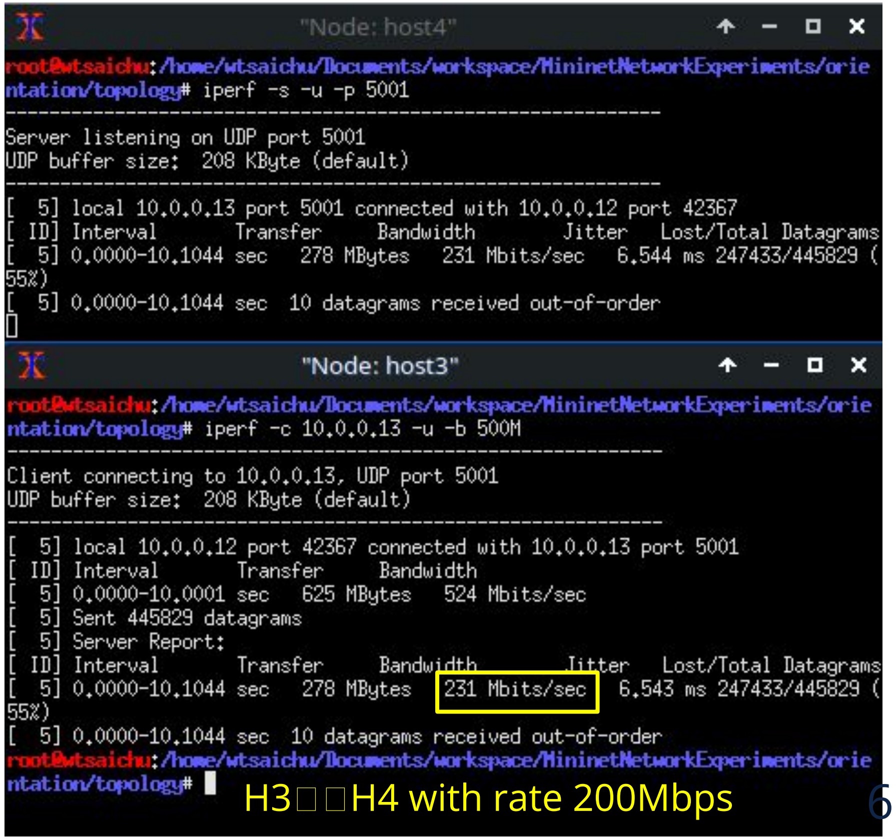
Meter test



```
"Node: host3"
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -s -u -p 5001
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.12 port 5001 connected with 10.0.0.10 port 34395
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  5] 0.0000-10.1037 sec   417 MBytes   346 Mbits/sec   6.499 ms 148344/445832 (
33%)
[  5] 0.0000-10.1037 sec  13 datagrams received out-of-order
```

```
"Node: host1"
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -c 10.0.0.12 -u -b 500M
------------------------------------------------------------
Client connecting to 10.0.0.12, UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.10 port 34395 connected with 10.0.0.12 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5] 0.0000-10.0002 sec   625 MBytes   524 Mbits/sec
[  5] Sent 445832 datagrams
[  5] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  5] 0.0000-10.1037 sec   417 MBytes   346 Mbits/sec   6.498 ms 148344/445832 (
33%)
[  5] 0.0000-10.1037 sec  13 datagrams received out-of-order
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# 
```

H1□□H3 with rate 300Mbps



```
"Node: host4"
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -s -u -p 5001
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.13 port 5001 connected with 10.0.0.12 port 42367
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  5] 0.0000-10.1044 sec   278 MBytes   231 Mbits/sec   6.544 ms 247433/445829 (
55%)
[  5] 0.0000-10.1044 sec  10 datagrams received out-of-order
```

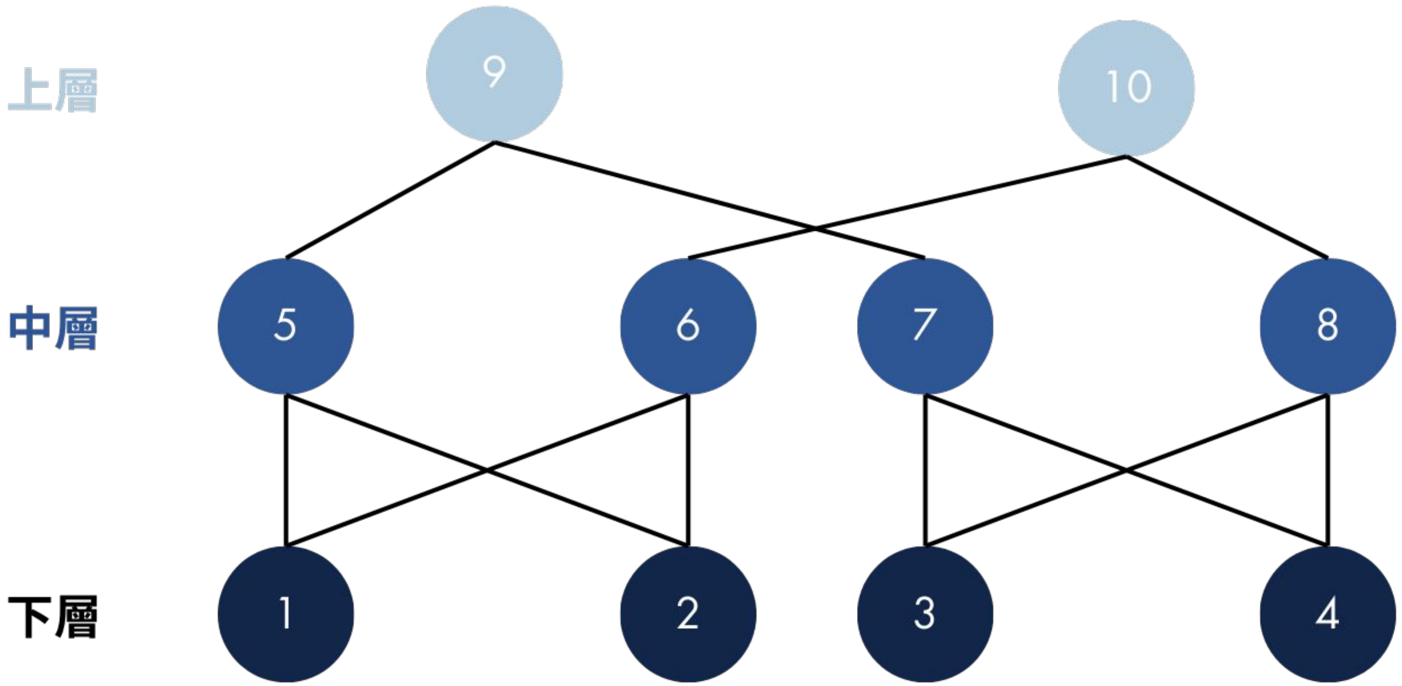
```
"Node: host3"
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# iperf -c 10.0.0.13 -u -b 500M
------------------------------------------------------------
Client connecting to 10.0.0.13, UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.12 port 42367 connected with 10.0.0.13 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5] 0.0000-10.0001 sec   625 MBytes   524 Mbits/sec
[  5] Sent 445829 datagrams
[  5] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  5] 0.0000-10.1044 sec   278 MBytes   231 Mbits/sec   6.543 ms 247433/445829 (
55%)
[  5] 0.0000-10.1044 sec  10 datagrams received out-of-order
root@wtsaichu:/home/wtsaichu/Documents/workspace/MininetNetworkExperiments/orie
ntation/topology# 
```

H3□□H4 with rate 200Mbps

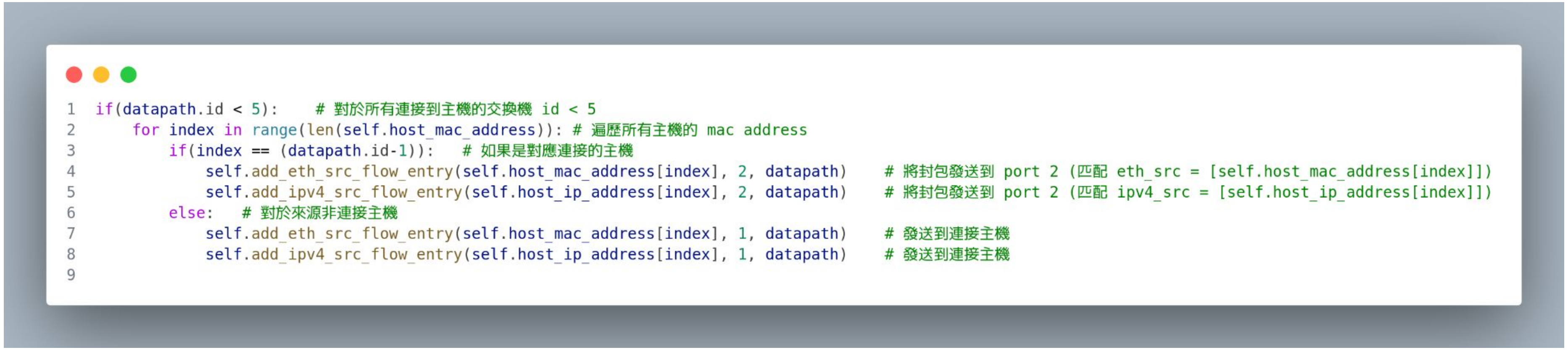
Flow entry

priority	match								datapath	actions	
	eth_type	ip_proto	eth_src	eth_dst	ipv4_src	ipv4_dst	tcp_dst	in_port		output port	meter
0	-								ALL	CONTROLLER	-
1	-	-	✓	-	-	-	-	-	5,6,7,8	2	
1	-	-	-	-	-	-	-	✓	9,10	1	
2	0x0800	6	-	-	✓	-	-	-	5,6,7,8	2	
3	-	-	✓	-	-	-	-	-	1,2,3,4	1	
4	0x0800	6	-	-	✓	-	-	-	1,2,3,4	1	
5	-	-	-	✓	-	-	-	-	5,6,7,8	1	
6	0x0800	6	-	-	-	✓	-	-	5,6,7,8	1	
7	0x0800	6	-	-	-	✓	5001	-	5,6,7,8	1	
8	-	-	✓	✓	-	-	-	-	5,6,7,8	1	
9	0x0800	6	-	-	✓	✓	-	-	5,6,7,8	1	
10	0x0800	6	-	-	✓	✓	5001	-	5,6,7,8	1	
11	0x0800	17	-	-	✓	✓	-	-	*	1	✓

Topology

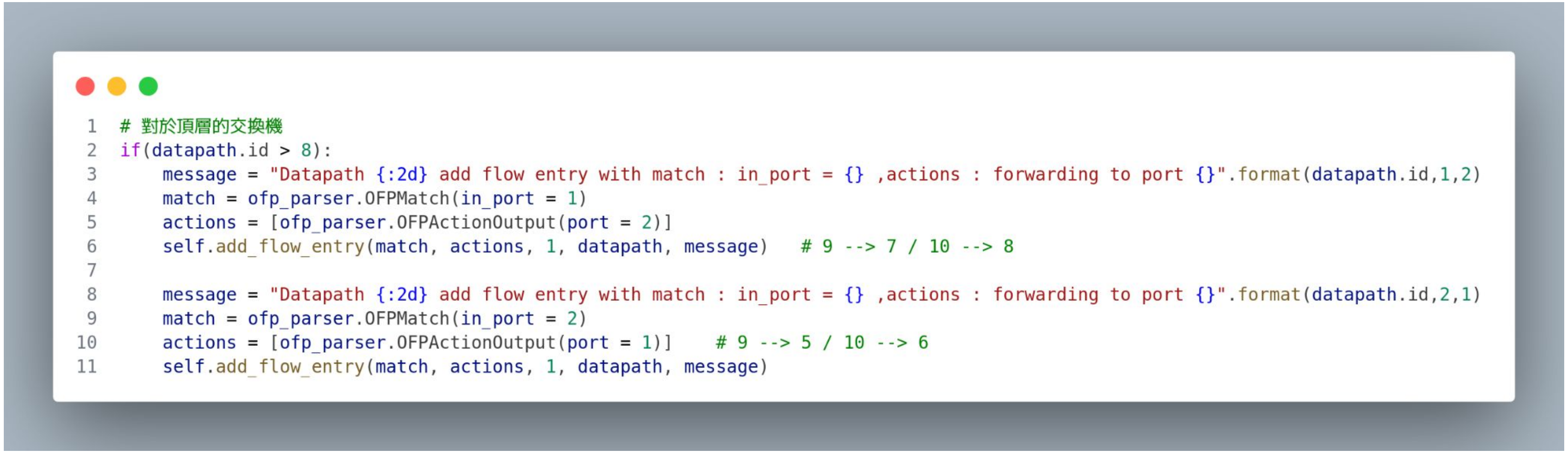


Datapath_id < 5



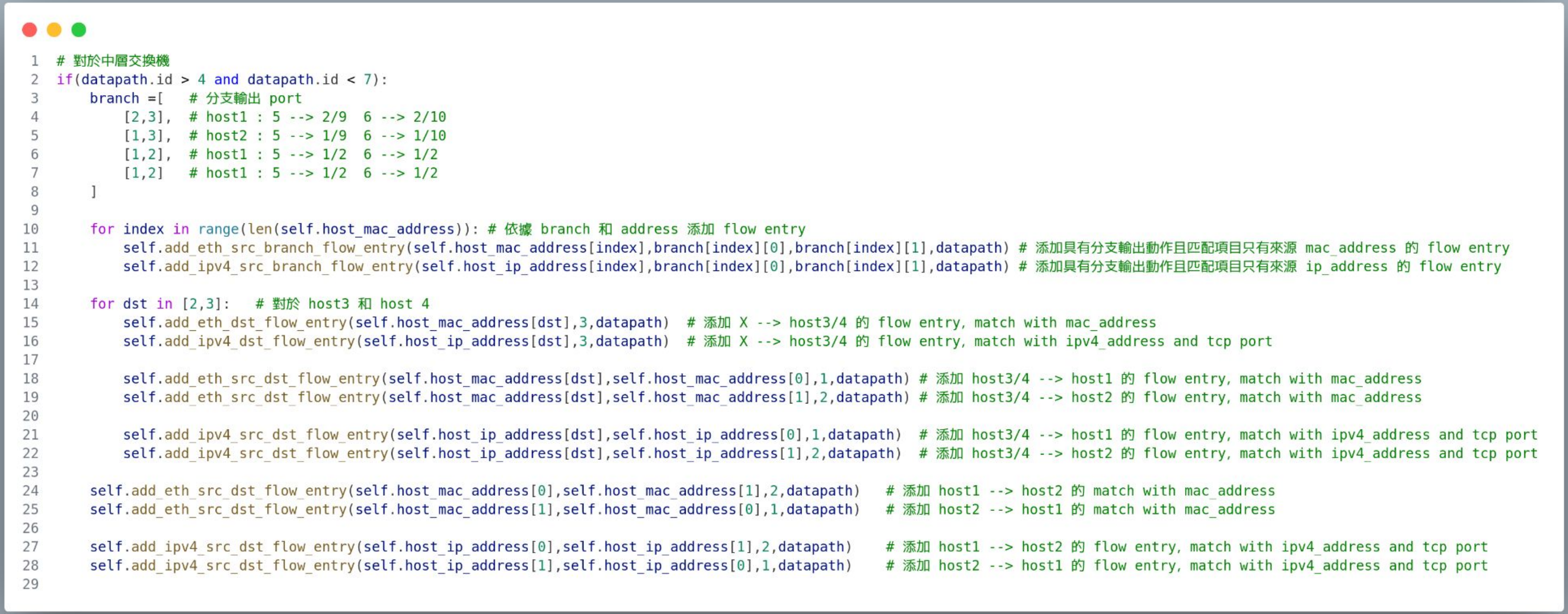
```python
if(datapath.id < 5):     # 對於所有連接到主機的交換機 id < 5
    for index in range(len(self.host_mac_address)): # 遍歷所有主機的 mac address
        if(index == (datapath.id-1)):   # 如果是對應連接的主機
            self.add_eth_src_flow_entry(self.host_mac_address[index], 2, datapath)    # 將封包發送到 port 2 (匹配 eth_src = [self.host_mac_address[index]])
            self.add_ipv4_src_flow_entry(self.host_ip_address[index], 2, datapath)    # 將封包發送到 port 2 (匹配 ipv4_src = [self.host_ip_address[index]])
        else:   # 對於來源非連接主機
            self.add_eth_src_flow_entry(self.host_mac_address[index], 1, datapath)    # 發送到連接主機
            self.add_ipv4_src_flow_entry(self.host_ip_address[index], 1, datapath)    # 發送到連接主機

```

Datapath_id > 8



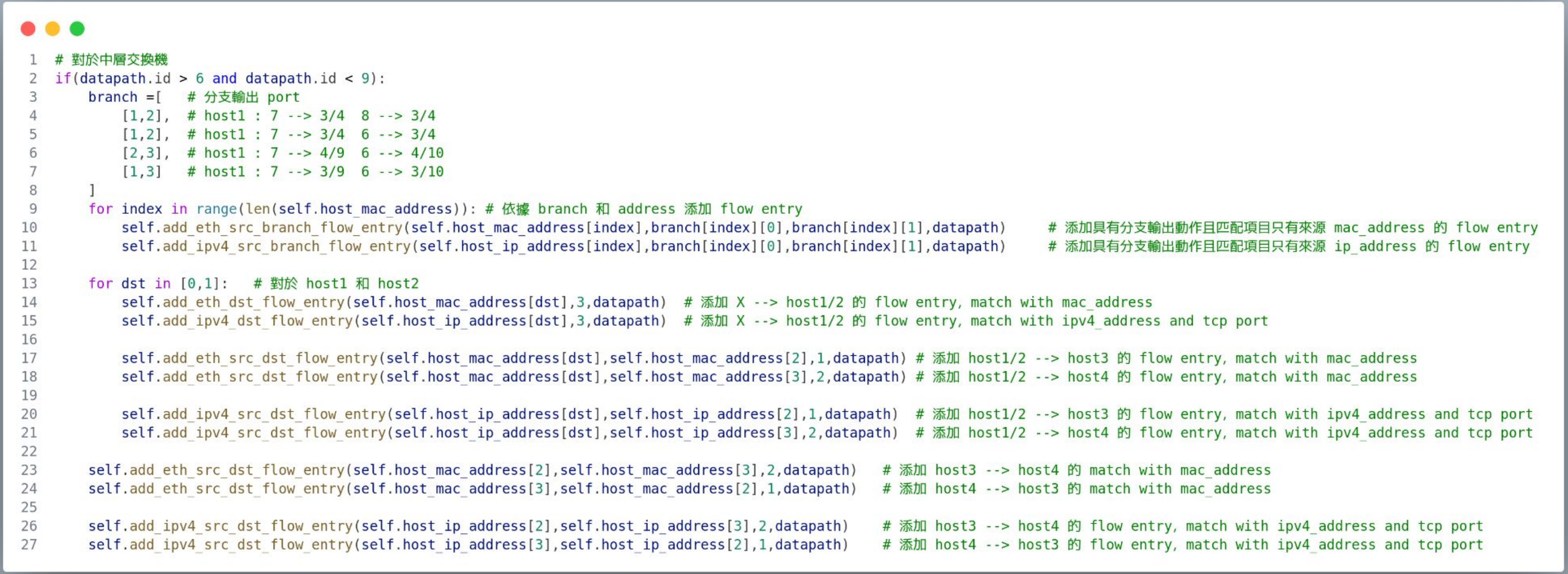
```
# 對於頂層的交換機
if(datapath.id > 8):
    message = "Datapath {:2d} add flow entry with match : in_port = {} ,actions : forwarding to port {}".format(datapath.id,1,2)
    match = ofp_parser.OFPMatch(in_port = 1)
    actions = [ofp_parser.OFPActionOutput(port = 2)]
    self.add_flow_entry(match, actions, 1, datapath, message)   # 9 --> 7 / 10 --> 8

    message = "Datapath {:2d} add flow entry with match : in_port = {} ,actions : forwarding to port {}".format(datapath.id,2,1)
    match = ofp_parser.OFPMatch(in_port = 2)
    actions = [ofp_parser.OFPActionOutput(port = 1)]    # 9 --> 5 / 10 --> 6
    self.add_flow_entry(match, actions, 1, datapath, message)
```

Datapath_id > 4 and Datapath_id < 7



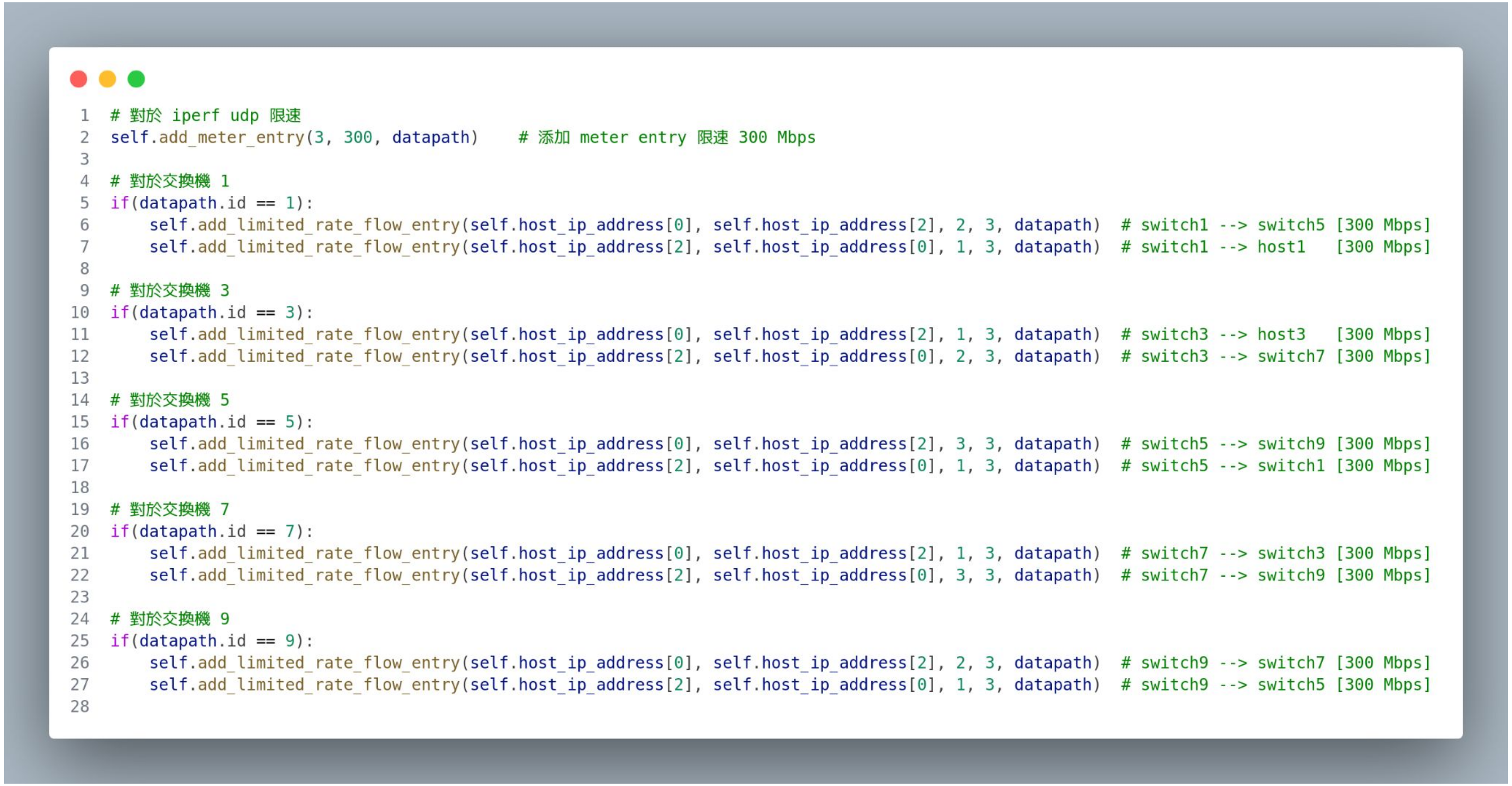
```
# 對於中層交換機
if(datapath.id > 4 and datapath.id < 7):
    branch =[   # 分支輸出 port
        [2,3],  # host1 : 5 --> 2/9  6 --> 2/10
        [1,3],  # host2 : 5 --> 1/9  6 --> 1/10
        [1,2],  # host1 : 5 --> 1/2  6 --> 1/2
        [1,2]   # host1 : 5 --> 1/2  6 --> 1/2
    ]

    for index in range(len(self.host_mac_address)): # 依據 branch 和 address 添加 flow entry
        self.add_eth_src_branch_flow_entry(self.host_mac_address[index],branch[index][0],branch[index][1],datapath) # 添加具有分支輸出動作且匹配項目只有來源 mac_address 的 flow entry
        self.add_ipv4_src_branch_flow_entry(self.host_ip_address[index],branch[index][0],branch[index][1],datapath) # 添加具有分支輸出動作且匹配項目只有來源 ip_address 的 flow entry

    for dst in [2,3]:   # 對於 host3 和 host 4
        self.add_eth_dst_flow_entry(self.host_mac_address[dst],3,datapath)  # 添加 X --> host3/4 的 flow entry, match with mac_address
        self.add_ipv4_dst_flow_entry(self.host_ip_address[dst],3,datapath)  # 添加 X --> host3/4 的 flow entry, match with ipv4_address and tcp port

        self.add_eth_src_dst_flow_entry(self.host_mac_address[dst],self.host_mac_address[0],1,datapath) # 添加 host3/4 --> host1 的 flow entry, match with mac_address
        self.add_eth_src_dst_flow_entry(self.host_mac_address[dst],self.host_mac_address[1],2,datapath) # 添加 host3/4 --> host2 的 flow entry, match with mac_address

        self.add_ipv4_src_dst_flow_entry(self.host_ip_address[dst],self.host_ip_address[0],1,datapath)  # 添加 host3/4 --> host1 的 flow entry, match with ipv4_address and tcp port
        self.add_ipv4_src_dst_flow_entry(self.host_ip_address[dst],self.host_ip_address[1],2,datapath)  # 添加 host3/4 --> host2 的 flow entry, match with ipv4_address and tcp port

    self.add_eth_src_dst_flow_entry(self.host_mac_address[0],self.host_mac_address[1],2,datapath)   # 添加 host1 --> host2 的 match with mac_address
    self.add_eth_src_dst_flow_entry(self.host_mac_address[1],self.host_mac_address[0],1,datapath)   # 添加 host2 --> host1 的 match with mac_address

    self.add_ipv4_src_dst_flow_entry(self.host_ip_address[0],self.host_ip_address[1],2,datapath)    # 添加 host1 --> host2 的 flow entry, match with ipv4_address and tcp port
    self.add_ipv4_src_dst_flow_entry(self.host_ip_address[1],self.host_ip_address[0],1,datapath)    # 添加 host2 --> host1 的 flow entry, match with ipv4_address and tcp port
```

Datapath_id > 6 and Datapath_id < 9



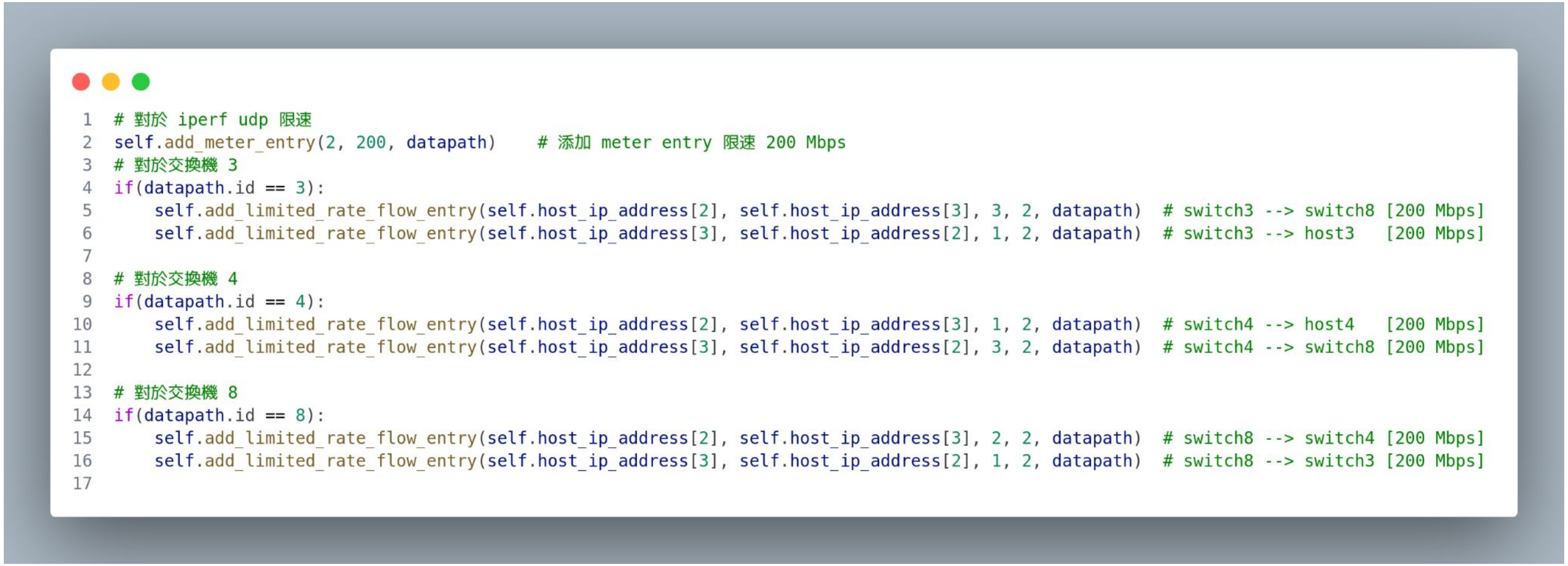
```python
# 對於中層交換機
if(datapath.id > 6 and datapath.id < 9):
    branch =[   # 分支輸出 port
        [1,2],  # host1 : 7 --> 3/4  8 --> 3/4
        [1,2],  # host1 : 7 --> 3/4  6 --> 3/4
        [2,3],  # host1 : 7 --> 4/9  6 --> 4/10
        [1,3]   # host1 : 7 --> 3/9  6 --> 3/10
    ]
    for index in range(len(self.host_mac_address)): # 依據 branch 和 address 添加 flow entry
        self.add_eth_src_branch_flow_entry(self.host_mac_address[index],branch[index][0],branch[index][1],datapath)     # 添加具有分支輸出動作且匹配項目只有來源 mac_address 的 flow entry
        self.add_ipv4_src_branch_flow_entry(self.host_ip_address[index],branch[index][0],branch[index][1],datapath)     # 添加具有分支輸出動作且匹配項目只有來源 ip_address 的 flow entry

    for dst in [0,1]:   # 對於 host1 和 host2
        self.add_eth_dst_flow_entry(self.host_mac_address[dst],3,datapath)  # 添加 X --> host1/2 的 flow entry, match with mac_address
        self.add_ipv4_dst_flow_entry(self.host_ip_address[dst],3,datapath)  # 添加 X --> host1/2 的 flow entry, match with ipv4_address and tcp port

        self.add_eth_src_dst_flow_entry(self.host_mac_address[dst],self.host_mac_address[2],1,datapath) # 添加 host1/2 --> host3 的 flow entry, match with mac_address
        self.add_eth_src_dst_flow_entry(self.host_mac_address[dst],self.host_mac_address[3],2,datapath) # 添加 host1/2 --> host4 的 flow entry, match with mac_address

        self.add_ipv4_src_dst_flow_entry(self.host_ip_address[dst],self.host_ip_address[2],1,datapath)  # 添加 host1/2 --> host3 的 flow entry, match with ipv4_address and tcp port
        self.add_ipv4_src_dst_flow_entry(self.host_ip_address[dst],self.host_ip_address[3],2,datapath)  # 添加 host1/2 --> host4 的 flow entry, match with ipv4_address and tcp port

    self.add_eth_src_dst_flow_entry(self.host_mac_address[2],self.host_mac_address[3],2,datapath)   # 添加 host3 --> host4 的 match with mac_address
    self.add_eth_src_dst_flow_entry(self.host_mac_address[3],self.host_mac_address[2],1,datapath)   # 添加 host4 --> host3 的 match with mac_address

    self.add_ipv4_src_dst_flow_entry(self.host_ip_address[2],self.host_ip_address[3],2,datapath)    # 添加 host3 --> host4 的 flow entry, match with ipv4_address and tcp port
    self.add_ipv4_src_dst_flow_entry(self.host_ip_address[3],self.host_ip_address[2],1,datapath)    # 添加 host4 --> host3 的 flow entry, match with ipv4_address and tcp port
```

iperfudp test H1□□H3 with rate 300Mbps



```
# 對於 iperf udp 限速
self.add_meter_entry(3, 300, datapath)    # 添加 meter entry 限速 300 Mbps

# 對於交換機 1
if(datapath.id == 1):
    self.add_limited_rate_flow_entry(self.host_ip_address[0], self.host_ip_address[2], 2, 3, datapath)  # switch1 --> switch5 [300 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[0], 1, 3, datapath)  # switch1 --> host1   [300 Mbps]

# 對於交換機 3
if(datapath.id == 3):
    self.add_limited_rate_flow_entry(self.host_ip_address[0], self.host_ip_address[2], 1, 3, datapath)  # switch3 --> host3   [300 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[0], 2, 3, datapath)  # switch3 --> switch7 [300 Mbps]

# 對於交換機 5
if(datapath.id == 5):
    self.add_limited_rate_flow_entry(self.host_ip_address[0], self.host_ip_address[2], 3, 3, datapath)  # switch5 --> switch9 [300 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[0], 1, 3, datapath)  # switch5 --> switch1 [300 Mbps]

# 對於交換機 7
if(datapath.id == 7):
    self.add_limited_rate_flow_entry(self.host_ip_address[0], self.host_ip_address[2], 1, 3, datapath)  # switch7 --> switch3 [300 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[0], 3, 3, datapath)  # switch7 --> switch9 [300 Mbps]

# 對於交換機 9
if(datapath.id == 9):
    self.add_limited_rate_flow_entry(self.host_ip_address[0], self.host_ip_address[2], 2, 3, datapath)  # switch9 --> switch7 [300 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[0], 1, 3, datapath)  # switch9 --> switch5 [300 Mbps]

```

iperfudp test H3□□H4 with rate 200Mbps


```
# 對於 iperf udp 限速
self.add_meter_entry(2, 200, datapath)    # 添加 meter entry 限速 200 Mbps
# 對於交換機 3
if(datapath.id == 3):
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[3], 3, 2, datapath)  # switch3 --> switch8 [200 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[3], self.host_ip_address[2], 1, 2, datapath)  # switch3 --> host3   [200 Mbps]

# 對於交換機 4
if(datapath.id == 4):
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[3], 1, 2, datapath)  # switch4 --> host4   [200 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[3], self.host_ip_address[2], 3, 2, datapath)  # switch4 --> switch8 [200 Mbps]

# 對於交換機 8
if(datapath.id == 8):
    self.add_limited_rate_flow_entry(self.host_ip_address[2], self.host_ip_address[3], 2, 2, datapath)  # switch8 --> switch4 [200 Mbps]
    self.add_limited_rate_flow_entry(self.host_ip_address[3], self.host_ip_address[2], 1, 2, datapath)  # switch8 --> switch3 [200 Mbps]
```