

# Statistics of networks

# Consequent

---



# UDP Traffic (Using IPERF)

```
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> iperfudp 500M host1 host3
*** Iperf: testing UDP bandwidth between host1 and host3
*** Results: ['500M', '267 Mbits/sec', '267 Mbits/sec']
mininet> iperfudp 500M host2 host3
*** Iperf: testing UDP bandwidth between host2 and host3
*** Results: ['500M', '136 Mbits/sec', '136 Mbits/sec']
mininet> iperfudp 500M host1 host4
*** Iperf: testing UDP bandwidth between host1 and host4
*** Results: ['500M', '340 Mbits/sec', '340 Mbits/sec']
mininet> iperfudp 500M host2 host4
*** Iperf: testing UDP bandwidth between host2 and host4
*** Results: ['500M', '476 Mbits/sec', '476 Mbits/sec']
mininet>
```

H1 ↔ H3 with rate 200Mbps

H2 ↔ H3 with rate 100Mbps

H1 ↔ H4 with rate 300Mbps

H2 ↔ H4 with rate 400Mbps



# Querying switch features when switch connected to controller

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

wtsaichu@wtsaichu:~/Documents/workspace/MininetNetworkExperiments/orientation/controller
$ ryu-manager controller.py --log-file ryu.log
loading app controller.py
loading app ryu.controller.ofp_handler
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app controller.py of Flow_entries_management
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.topology.switches of Switches

Switch1 with datapath_id 1
n_buffer : NO_BUBBER ,n_tables : 254 ,auxiliary_id : MAIN_CONNECTION
capabilities : 79 = 001001111
OFPC_FLOW_STATS : SUPPORTED      OFPC_TABLE_STATS : SUPPORTED
OFPC_PORT_STATS : SUPPORTED      OFPC_GROUP_STATS : SUPPORTED
OFPC_IP_REASM   : NOT_SUPPORTED   OFPC_QUEUE_STATS : SUPPORTED
OFPC_PORT_BLOCKED : NOT_SUPPORTED

Switch2 with datapath_id 2
n_buffer : NO_BUBBER ,n_tables : 254 ,auxiliary_id : MAIN_CONNECTION
capabilities : 79 = 001001111
OFPC_FLOW_STATS : SUPPORTED      OFPC_TABLE_STATS : SUPPORTED
OFPC_PORT_STATS : SUPPORTED      OFPC_GROUP_STATS : SUPPORTED
OFPC_IP_REASM   : NOT_SUPPORTED   OFPC_QUEUE_STATS : SUPPORTED
OFPC_PORT_BLOCKED : NOT_SUPPORTED

Switch3 with datapath_id 3
```

EventOFPSwitchFeatures



# Receiving port status when switch ports changing happened

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

occupied_bandwidth_tx :      6528 bps =      0.01 Mbps
=====

Datapath 9 Port 2 Statics:
port_no = 2, hw_addr = 26:3d:1b:33:13:06, name = switch9-eth2

config : 0000
OFPPC_PORT_DOWN : UP          OFPPC_NO_PACKET_IN : CAN SEND PACKET-IN MESSAGES
OFPPC_NO_RECV   : CAN RECEIVE PACKETS OFPPC_NO_FWD   : CAN FORWARD PACKETS

port_state      : LIVE
curr            : 2112          supported           : 0
peer           : 0             advertised          : 0

OFPPF_10MB_HD   : NOT_SUPPORTED OFPPF_100MB_HD   : NOT_SUPPORTED OFPPF_1GB_HD   : NOT_SUPPORTED
OFPPF_10MB_FD   : NOT_SUPPORTED OFPPF_100MB_FD   : NOT_SUPPORTED OFPPF_1GB_FD   : NOT_SUPPORTED
max_speed       :      0 bps =    0.00 Mbps
curr_speed      : 10000000 bps =    9.54 Mbps

Last updated time :      5.04 second
rx_packets       :      5 rx_bytes       :      438 rx_errors : 0
tx_packets       :      2 tx_bytes       :      176 tx_errors : 0

free_bandwidth_rx : 9996268 bps =    9.53 Mbps
free_bandwidth_tx : 168387404 bps = 160.59 Mbps
occupied_bandwidth_rx : 3731 bps =    170.12 Mbps
occupied_bandwidth_tx : 178387404 bps = 170.12 Mbps
=====
advertised : No Advertised Features
```

from  
EventOFPPortDescStatsReply

mininet > iperf 500M host1 host3

from  
EventOFPPortStatsReply

Calculation



# Monitoring flow and port statistics every 5s

```
Terminal - wtsaichu@wtsaichu: ~/Documents/workspace/MininetNetworkExperiments/orientation/controller
File Edit View Terminal Tabs Help
advertised : No Advertised Features

=====
Datapath 9 Port 1 Statics:
port_no = 1, hw_addr = 9e:5f:5c:d1:e1:e1, name = switch9-eth1

config : 0000
OFPPC_PORT_DOWN : UP                OFPPC_NO_PACKET_IN : CAN SEND PACKET-IN MESSAGES
OFPPC_NO_RECV   : CAN RECEIVE PACKETS OFPPC_NO_FWD      : CAN FORWARD PACKETS

port_state      : LIVE
curr            : 2112                supported          : 0
peer            : 0                  advertised          : 0

OFPPF_10MB_HD   : NOT_SUPPORTED      OFPPF_100MB_HD    : NOT_SUPPORTED      OFPPF_1GB_HD   : NOT_SUPPORTED
OFPPF_10MB_FD   : NOT_SUPPORTED      OFPPF_100MB_FD    : NOT_SUPPORTED      OFPPF_1GB_FD   : NOT_SUPPORTED
max_speed       : 0 bps =             0.00 Mbps
curr_speed      : 10000000 bps =      9.54 Mbps

=====
Last updated time : 300.28 second  上一次更新的時間
rx_packets       : 29 rx_bytes       : 5202 rx_errors : 0
tx_packets       : 45 tx_bytes       : 8644 tx_errors : 0

=====
free_bandwidth_rx : 3115797 bps =     2.97 Mbps
free_bandwidth_tx : 9999802 bps =     9.54 Mbps
occupied_bandwidth_rx : 13115797 bps =    0.00 Mbps
occupied_bandwidth_tx : 197 bps =      0.00 Mbps
=====
```



# Monitoring flow and port statistics every 5s

Flow  
entry

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

=====
Datapath 2 with 13 flow entries: 交換機有的 flow entry 數量
=====

Datapath 2, 1th flow entry in time 5.22 second:
table_id : 0 priority : 8 duration_sec : 2 cookie : 0
flags : 0 length : 112
idle_timeout : 0 hard_timeout : 0
packet_count : 0 byte_count : 0 基本訊息

Match fields :
| eth_type | ipv4_src | ipv4_dst | ip_proto |
| 2048 | 10.0.0.11 | 10.0.0.12 | 17 | Match Field

Instructions fields :
8
[{'OFActionOutput': {'len': 16, 'max_len': 65509, 'port': 3, 'type': 0}}] Instruction Field

=====
Datapath 2, 2th flow entry in time 5.22 second: 上一次更新的時間
table_id : 0 priority : 8 duration_sec : 2 cookie : 0
flags : 0 length : 112
idle_timeout : 0 hard_timeout : 0
packet_count : 0 byte_count : 0

Match fields :
| eth_type | ipv4_src | ipv4_dst | ip_proto |
| 2048 | 10.0.0.12 | 10.0.0.11 | 17 |

Instructions fields :
8
[{'OFActionOutput': {'len': 16, 'max_len': 65509, 'port': 1, 'type': 0}}]
```

# Thinking

---





# UDP Traffic (Using IPERF)

對於所有 Datapath 添加四個 meter entry



```
1 self.add_meter_entry(1,100,datapath) # 限制流量速率為 100, meter_ID 為 1
2 self.add_meter_entry(2,200,datapath) # 限制流量速率為 200, meter_ID 為 2
3 self.add_meter_entry(3,300,datapath) # 限制流量速率為 300, meter_ID 為 3
4 self.add_meter_entry(4,400,datapath) # 限制流量速率為 400, meter_ID 為 4
```



# UDP Traffic (Using IPERF)

- H1 ↔ H3 with rate 200Mbps

```
1 if(datapath.id == 1):
2     self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[2],2,2,datapath)
3     self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[0],1,2,datapath)
4
5 if(datapath.id == 5):
6     self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[2],3,2,datapath)
7     self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[0],1,2,datapath)
8
9 if(datapath.id == 9):
10    self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[2],2,2,datapath)
11    self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[0],1,2,datapath)
12
13 if(datapath.id == 7):
14    self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[2],1,2,datapath)
15    self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[0],3,2,datapath)
16
17 if(datapath.id == 3):
18    self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[2],1,2,datapath)
19    self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[0],2,2,datapath)
```



# UDP Traffic (Using IPERF)

- H2 ☐ ☐ H3 with rate 100Mbps

```
1 if(datapath.id == 2):
2     self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[2],3,1,datapath)
3     self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[1],1,1,datapath)
4
5 if(datapath.id == 6):
6     self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[2],3,1,datapath)
7     self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[1],2,1,datapath)
8
9 if(datapath.id == 10):
10    self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[2],2,1,datapath)
11    self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[1],1,1,datapath)
12
13 if(datapath.id == 8):
14    self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[2],1,1,datapath)
15    self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[1],3,1,datapath)
16
17 if(datapath.id == 3):
18    self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[2],1,1,datapath)
19    self.add_limited_rate_flow_entry(self.host_ip_address[2],self.host_ip_address[1],3,1,datapath)
```



# UDP Traffic (Using IPERF)

- H1 ☐ ☐ H4 with rate 300Mbps

```
1 if(datapath.id == 1):
2     self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[3],2,3,datapath)
3     self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[0],1,3,datapath)
4
5 if(datapath.id == 5):
6     self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[3],3,3,datapath)
7     self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[0],1,3,datapath)
8
9 if(datapath.id == 9):
10    self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[3],2,3,datapath)
11    self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[0],1,3,datapath)
12
13 if(datapath.id == 7):
14    self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[3],2,3,datapath)
15    self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[0],3,3,datapath)
16
17 if(datapath.id == 4):
18    self.add_limited_rate_flow_entry(self.host_ip_address[0],self.host_ip_address[3],1,3,datapath)
19    self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[0],2,3,datapath)
20
```



# UDP Traffic (Using IPERF)

- H2 ↔ H4 with rate 400Mbps

```
1 if(datapath.id == 2):
2     self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[3],3,4,datapath)
3     self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[1],1,4,datapath)
4
5 if(datapath.id == 6):
6     self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[3],3,4,datapath)
7     self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[1],2,4,datapath)
8
9 if(datapath.id == 10):
10    self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[3],2,4,datapath)
11    self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[1],1,4,datapath)
12
13 if(datapath.id == 8):
14    self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[3],2,4,datapath)
15    self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[1],3,4,datapath)
16
17 if(datapath.id == 4):
18    self.add_limited_rate_flow_entry(self.host_ip_address[1],self.host_ip_address[3],1,4,datapath)
19    self.add_limited_rate_flow_entry(self.host_ip_address[3],self.host_ip_address[1],3,4,datapath)
20
```



## Querying switch features when switch connected to controller

```
1 switch_features = [message.datapath_id,message.n_buffers,message.n_tables,message.auxiliary_id,message.capabilities] # 取得 features
2 self.switch_features.append(switch_features) # 添加 features 在 switch_features
3 self.write_switch_features() # 寫入 switch_features.csv
4 self.show_switch_features_and_configuration(datapath) # 顯示交換機 feature 的能力
5
```



# Querying switch features when switch connected to controller

```
1 # 顯示交換機 features 以及配置
2 def show_switch_features_and_configuration(self, datapath):
3     datapath_id = str(datapath.id) # 交換機 ID
4
5     feature = [] # 交換機 feature
6     n_buffer = 0 # 可用的封包緩衝區數量
7     auxiliary_id = 0 # 輔助連接 ID
8
9     for switch_feature in self.switch_features:
10         if (str(switch_feature[0]) == datapath_id):
11             feature = switch_feature
12         if (feature[1] == 0): # 交換機是否支持 buffer
13             n_buffer = "NO_BUBBER"
14         else:
15             n_buffer = str(feature[1])
16         if (feature[3] == 0):
17             auxiliary_id = "MAIN_CONNECTION"
18         else:
19             auxiliary_id = str(feature[3])
20
21     capabilities = feature[4] # 交換機支持的功能
22     capabilities_binary_string = format(capabilities, '09b') # 轉換為 9 位元二元序列
23
24     OFPC_FLOW_STATS = "SUPPORTED" if bool(int(capabilities_binary_string[8])) else "NOT_SUPPORTED" # 是否支持 flow 統計功能
25     OFPC_TABLE_STATS = "SUPPORTED" if bool(int(capabilities_binary_string[7])) else "NOT_SUPPORTED" # 是否支持 table 統計功能
26     OFPC_PORT_STATS = "SUPPORTED" if bool(int(capabilities_binary_string[6])) else "NOT_SUPPORTED" # 是否支持 port 統計功能
27     OFPC_GROUP_STATS = "SUPPORTED" if bool(int(capabilities_binary_string[5])) else "NOT_SUPPORTED" # 是否支持 group 統計功能
28     OFPC_IP_REASM = "SUPPORTED" if bool(int(capabilities_binary_string[3])) else "NOT_SUPPORTED" # 是否支持 IP 重組功能
29     OFPC_QUEUE_STATS = "SUPPORTED" if bool(int(capabilities_binary_string[2])) else "NOT_SUPPORTED" # 是否支持 Queue 統計功能
30     OFPC_PORT_BLOCKED = "SUPPORTED" if bool(int(capabilities_binary_string[0])) else "NOT_SUPPORTED" # 是否支持 PORT_BLOCKED
31
32     self.print_split_line('-', True)
33     print("Switch{:2s} with datapath id {:2s}".format(datapath_id, datapath_id))
34     print("n_buffer : {:10s}, n_tables : {:3d}, auxiliary_id : {:16s}".format(n_buffer, feature[2], auxiliary_id))
35     print("capabilities : {:3d} = {:8s}".format(capabilities, capabilities_binary_string))
36     print("OFPC_FLOW_STATS : {:15s} OFPC_TABLE_STATS : {:15s}".format(OFPC_FLOW_STATS, OFPC_TABLE_STATS))
37     print("OFPC_PORT_STATS : {:15s} OFPC_GROUP_STATS : {:15s}".format(OFPC_PORT_STATS, OFPC_GROUP_STATS))
38     print("OFPC_IP_REASM : {:15s} OFPC_QUEUE_STATS : {:15s}".format(OFPC_IP_REASM, OFPC_QUEUE_STATS))
39     print("OFPC_PORT_BLOCKED : {:15s}".format(OFPC_PORT_BLOCKED))
40     self.print_split_line('-', False)
```

	Datapath_id▼	N_buffers ▼	N_tables ▼	Auxiliary_id▼	Capabilities▼
	1	0	254	0	79
	2	0	254	0	79
	3	0	254	0	79
	4	0	254	0	79
	5	0	254	0	79
	6	0	254	0	79
	7	0	254	0	79
	8	0	254	0	79
	9	0	254	0	79
	10	0	254	0	79





# Receiving port status when switch ports changing happened

```
Terminal - wtsaichu@wtsaichu: ~/Documents/workspace/MininetNetworkExperiments/orientation/controller
File Edit View Terminal Tabs Help
occupied_bandwidth_tx :      6528 bps =      0.01 Mbps
=====
Datapath 9 Port 2 Statics:
port_no = 2, hw_addr = 26:3d:1b:33:13:06, name = switch9-eth2
config : 0000
OFPPC_PORT_DOWN : UP          OFPPC_NO_PACKET_IN : CAN SEND PACKET-IN MESSAGES
OFPPC_NO_RECV   : CAN RECEIVE PACKETS OFPPC_NO_FWD   : CAN FORWARD PACKETS
port_state      : LIVE
curr            : 2112          supported           : 0
peer            : 0            advertised            : 0
OFPPF_10MB_HD   : NOT_SUPPORTED OFPPF_100MB_HD : NOT_SUPPORTED OFPPF_1GB_HD : NOT_SUPPORTED
OFPPF_10MB_FD   : NOT_SUPPORTED OFPPF_100MB_FD : NOT_SUPPORTED OFPPF_1GB_FD : NOT_SUPPORTED
max_speed       :      0 bps =    0.00 Mbps
curr_speed      : 10000000 bps =    9.54 Mbps
Last updated time :      5.04 second
rx_packets      :      5 rx_bytes      :      438 rx_errors : 0
tx_packets      :      2 tx_bytes      :      176 tx_errors : 0
free_bandwidth_rx : 9996268 bps =    9.53 Mbps
free_bandwidth_tx : 168387404 bps = 160.59 Mbps
occupied_bandwidth_rx : 3731 bps =    170.12 Mbps
occupied_bandwidth_tx : 178387404 bps = 170.12 Mbps
advertisd : No Advertised Features
```

from  
EventOFPPortDescStatsReply

mininet > iperf 500M host1 host3

from  
EventOFPPortStatsReply

Calculation





## Receiving port status when switch ports changing happened

```
1 # iperfudp 會觸發 EventOFPPortStatsReply 事件進行統計
2 @set_ev_cls(ofp_event.EventOFPPortStatsReply,MAIN_DISPATCHER)
3 def _port_stats_reply_handler(self,event):
4     ofproto = event.msg.datapath.ofproto # OpenFlow 協議相關訊息
5     ports = event.msg.body # 交換機上的 port
6     for port in ports:
7         port_number = port.port_no
8         datapath_id = event.msg.datapath.id
9         if(port_number < ofproto.OFPP_MAX):
10             rx_packets = port.rx_packets # 接收到的總 packet 總數
11             tx_packets = port.tx_packets # 傳輸的總 packet 總數
12             rx_bytes = port.rx_bytes # 接收到的 bytes 總數
13             tx_bytes = port.tx_bytes # 傳輸的 bytes 總數
14             rx_errors = port.rx_errors # 接收到的 error 總數
15             tx_errors = port.tx_errors # 傳輸的 error 總數
```



# Receiving port status when switch ports changing happened

```
1 # OFPPortDescStatsRequest 的響應, 統計 port 的資訊
2 @set_ev_cls(ofp_event.EventOFPPortDescStatsReply, MAIN_DISPATCHER)
3 def port_desc_stats_reply_handler(self, event):
4     # 取得訊息
5     datapath = event.msg.datapath # 數據平面的交換機 (datapath) 結構
6     ofproto = datapath.ofproto    # OpenFlow 協議相關訊息
7
8     ports = {}
9     # 遍歷 event 中收到的每個 port 的統計訊息
10    for statistic in event.msg.body:
11        if statistic.port_no <= ofproto.OFPP_MAX: # 如果 port_no(port number) 小於或等於 OFPP_MAX (最大的 port number) -> 表示該 port 有效且不是 reserved port
12            # print("\nconfig:")
13            config = format(int(statistic.config), '04b')
14            OFPPC_PORT_DOWN = "DOWN" if int(config[3]) == 1 else "UP"
15            OFPPC_NO_RECV = "NOT RECEIVE PACKETS" if int(config[2]) == 1 else "CAN RECEIVE PACKETS"
16            OFPPC_NO_FWD = "NOT FORWARD PACKETS" if int(config[1]) else "CAN FORWARD PACKETS"
17            OFPPC_NO_PACKET_IN = "NOT FORWARD PACKETS".upper() if int(config[0]) else "can send packet-in messages".upper()
18
19            port_state = "LIVE" if statistic.state else "NOT LIVE"
20
21            curr = statistic.curr
22            OFPPF_10MB_HD = "SUPPORTED" if bool(curr & 1) else "NOT_SUPPORTED"
23            OFPPF_10MB_FD = "SUPPORTED" if bool(curr & (1 << 9)) else "NOT_SUPPORTED"
24
25            advertised = statistic.advertised
26            if(advertised == 0):
27                print("advertised : No Advertised Features")
28            else:
29                OFPPPF_10MB_HD = "SUPPORTED" if bool(advertised & (1 << 0)) else "NOT_SUPPORTED" # 10 Mb half-duplex rate support.
30                OFPPPF_10MB_FD = "SUPPORTED" if bool(advertised & (1 << 1)) else "NOT_SUPPORTED" # 10 Mb full-duplex rate support.
31                OFPPPF_100MB_HD = "SUPPORTED" if bool(advertised & (1 << 2)) else "NOT_SUPPORTED" # 100 Mb half-duplex rate support.
32                OFPPPF_100MB_FD = "SUPPORTED" if bool(advertised & (1 << 3)) else "NOT_SUPPORTED" # 100 Mb full-duplex rate support.
33                OFPPPF_1GB_HD = "SUPPORTED" if bool(advertised & (1 << 4)) else "NOT_SUPPORTED" # 1 Gb half-duplex rate support.
34                OFPPPF_1GB_FD = "SUPPORTED" if bool(advertised & (1 << 5)) else "NOT_SUPPORTED" # 1 Gb full-duplex rate support.
35
36            OFPPPF_10MB_HD = "SUPPORTED" if bool(advertised & (1 << 0)) else "NOT_SUPPORTED" # 10 Mb half-duplex rate support.
37            OFPPPF_10MB_FD = "SUPPORTED" if bool(advertised & (1 << 1)) else "NOT_SUPPORTED" # 10 Mb full-duplex rate support.
38            OFPPPF_100MB_HD = "SUPPORTED" if bool(advertised & (1 << 2)) else "NOT_SUPPORTED" # 100 Mb half-duplex rate support.
39            OFPPPF_100MB_FD = "SUPPORTED" if bool(advertised & (1 << 3)) else "NOT_SUPPORTED" # 100 Mb full-duplex rate support.
40            OFPPPF_1GB_HD = "SUPPORTED" if bool(advertised & (1 << 4)) else "NOT_SUPPORTED" # 1 Gb half-duplex rate support.
41            OFPPPF_1GB_FD = "SUPPORTED" if bool(advertised & (1 << 5)) else "NOT_SUPPORTED" # 1 Gb full-duplex rate support.
42
```



# Receiving port status when switch ports changing happened

```
1 if('rx_bytes' in self.ports_statistic[datapath_id][port_number].keys()):
2     now_time = time.time() # 取得現在時間
3     elapsed_time = now_time - self.start_time # 取得執行時間
4     last_time = float(self.ports_statistic[datapath_id][port_number]['update_time']) # 上一次的更新時間
5     interval_time = elapsed_time - last_time # 兩次的時間間隔
6
7     rx_bytes_diff = rx_bytes - self.ports_statistic[datapath_id][port_number]['rx_bytes'] # 接收到的總 bytes 總數差異
8     tx_bytes_diff = tx_bytes - self.ports_statistic[datapath_id][port_number]['tx_bytes'] # 傳輸的總 bytes 總數差異
9
10    port_statistic = self.ports_statistic[datapath_id][port_number] # port 的統計資訊
11
12    occupied_bandwidth_rx = (rx_bytes_diff / interval_time) * 8 # bytes 轉換為 bits, 取得接收的 bandwidth, 也就是已經佔用的 bandwidth
13    occupied_bandwidth_tx = (tx_bytes_diff / interval_time) * 8 # bytes 轉換為 bits, 取得傳輸的 bandwidth, 也就是已經佔用的 bandwidth
14
15    self.ports_statistic[datapath_id][port_number].update({'occupied_bandwidth_rx': occupied_bandwidth_rx}) # 更新統計資訊
16    self.ports_statistic[datapath_id][port_number].update({'occupied_bandwidth_tx': occupied_bandwidth_tx})
17
18    free_bandwidth_rx = abs(port_statistic['curr_speed'] - occupied_bandwidth_rx) # 計算空間的 bandwidth
19    free_bandwidth_tx = abs(port_statistic['curr_speed'] - occupied_bandwidth_tx) # 計算空間的 bandwidth
20
21    self.ports_statistic[datapath_id][port_number].update({'free_bandwidth_rx': free_bandwidth_rx}) # 更新統計資訊
22    self.ports_statistic[datapath_id][port_number].update({'free_bandwidth_tx': free_bandwidth_tx})
23
24    # 解註解查看 port 統計資訊
25    # self.show_port_statistic_information(datapath_id, port_number, True)
26
27    now_time = time.time() # 取得現在時間
28    elapsed_time = now_time - self.start_time # 取得執行時間
29    self.ports_statistic[datapath_id][port_number].update({'update_time': elapsed_time}) # 更新執行時間
30    self.ports_statistic[datapath_id][port_number].update({'rx_packets': rx_packets}) # 接收到的總 packet 總數
31    self.ports_statistic[datapath_id][port_number].update({'tx_packets': tx_packets}) # 傳輸的總 packet 總數
32    self.ports_statistic[datapath_id][port_number].update({'rx_bytes': rx_bytes}) # 接收到的 bytes 總數
33    self.ports_statistic[datapath_id][port_number].update({'tx_bytes': tx_bytes}) # 傳輸的 bytes 總數
34    self.ports_statistic[datapath_id][port_number].update({'rx_errors': rx_errors}) # 接收到的 error 總數
35    self.ports_statistic[datapath_id][port_number].update({'tx_errors': tx_errors}) # 傳輸的 error 總數
36    self.write_port_statistic()
37
38    # 解註解查看 port 統計資訊
39    # self.show_port_statistic_information(datapath_id, port_number, False)
40
```



# Monitoring flow and port statistics every 5s

```
1  # 監控線程執行的函數，每 5 秒執行一次
2  def every_five_second_monitoring(self):
3      while(True):
4          try:
5              for datapath_id in self.datapaths:
6                  self.send_port_desc_stats_request(self.datapaths[datapath_id]) # 取得 port 配置
7                  self.send_flow_stats_request(self.datapaths[datapath_id]) # 取得 port 統計資訊
8                  self.send_port_stats_request(self.datapaths[datapath_id]) # 取得 Flow entry 統計資訊
9          except KeyError:
10             self.logger.debug("Topology discovery happens KeyError")
11             hub.sleep(5) # 停止 5 秒
```



# Monitoring flow and port statistics every 5s

```
1  # 監控線程執行的函數，每 5 秒執行一次
2  def every_five_second_monitoring(self):
3      while(True):
4          try:
5              for datapath_id in self.datapaths:
6                  self.send_port_desc_stats_request(self.datapaths[datapath_id]) # 取得 port 配置
7                  self.send_flow_stats_request(self.datapaths[datapath_id]) # 取得 port 統計資訊
8                  self.send_port_stats_request(self.datapaths[datapath_id]) # 取得 Flow entry 統計資訊
9          except KeyError:
10             self.logger.debug("Topology discovery happens KeyError")
11             hub.sleep(5) # 停止 5 秒
```



# Monitoring flow and port statistics every 5s

```
1 # 響應 ofp_event.EventOFPPFlowStatsReuest
2 @set_ev_cls(ofp_event.EventOFPPFlowStatsReply,MAIN_DISPATCHER)
3 def flow_stats_reply_handler(self,event):
4     flow_entries = event.msg.body # flow table(?) 該交換機的所有 flow enties
5     datapath = event.msg.datapath # 交換機 (datapath) 結構
6
7     self.print_split_line("-",True) # 起始分隔線
8     self.logger.info("Datapath{:2d} with {:3d} flow entries: ".format(datapath.id,len(flow_entries))) # 交換機名稱以及 flow entry 總數目
9
10    for index,flow_statistic in enumerate(flow_entries): # 遍歷 flow table
11        self.print_split_line("-",True) # 內部起始分隔線
12
13        now_time = time.time() # 取得現在時間
14        elapsed_time = now_time - self.start_time # 取得執行時間
15
16        self.logger.info("Datapath{:2d} , {:3d}th flow entry in time {:3.2f} second: ".format(datapath.id,index+1,elapsed_time)) # 再次顯示時間
17        cookie = flow_statistic.cookie # cookie
18        table_id = flow_statistic.table_id # flow table id
19        duration_sec = flow_statistic.duration_sec # 存活時間 (以秒為單位)
20        priority = flow_statistic.priority # flow entry 優先級
21
22        idle_timeout = flow_statistic.idle_timeout # 未匹配過期時間
23        hard_timeout = flow_statistic.hard_timeout # 存活過期時間
24
25        flags = flow_statistic.flags # 標誌
26        length = flow_statistic.length # 長度
27
28        packet_count = flow_statistic.packet_count # 匹配到的封包數量統計
29        byte_count = flow_statistic.byte_count # 匹配到的封包 bytes 數量統計
30
31        match = flow_statistic.match # match filed
32        instructions = flow_statistic.instructions # 執行動作
33
34        # 格式化輸出
35        self.logger.info("table id      :{:8d} priority      :{:8d} duration_sec :{:8d} cookie      :{:8d}".format(table_id,priority,duration_sec,cookie))
36        self.logger.info("flags      :{:16d} length      :{:16d}".format(flags,length))
37        self.logger.info("idle_timeout :{:16d} hard_timeout :{:16d}".format(idle_timeout,hard_timeout))
38        self.logger.info("packet_count :{:16d} byte_count  :{:16d}".format(packet_count,byte_count))
39
```





# Monitoring flow and port statistics every 5s

```
1  # 輸出 match field
2  self.logger.info("")
3  match_field_string = ""
4  match_value_string = ""
5  for match_field in match.to_jsondict()['OFFPMatch']['oxm_fields']: # 根據類型決定格式長度
6      field = match_field['OXMtlv']['field']
7      value = match_field['OXMtlv']['value']
8      if(field == 'eth_type'):
9          match_field_string += ' eth_type |'
10         match_value_string += " {:8s} |".format(str(value))
11     if(field == 'ip_proto'):
12         match_field_string += ' ip_proto |'
13         match_value_string += " {:8d} |".format(value)
14     if(field == 'eth_src'):
15         match_field_string += ' eth_src |'
16         match_value_string += " {:17s} |".format(value)
17     if(field == 'eth_dst'):
18         match_field_string += ' eth_dst |'
19         match_value_string += " {:17s} |".format(value)
20     if(field == 'ipv4_src'):
21         match_field_string += ' ipv4_src |'
22         match_value_string += " {:10s} |".format(value)
23     if(field == 'ipv4_dst'):
24         match_field_string += ' ipv4_dst |'
25         match_value_string += " {:10s} |".format(value)
26     if(field == 'tcp_dst'):
27         match_field_string += ' tcp_dst |'
28         match_value_string += " {:7d} |".format(value)
29     if(field == 'in_port'):
30         match_field_string += ' in_port |'
31         match_value_string += " {:7d} |".format(value)
32
33     if(len(match.to_jsondict()['OFFPMatch']['oxm_fields']) > 0):
34         self.logger.info("Match fields : ")
35         self.logger.info(match_field_string)
36         self.logger.info(match_value_string)
37     else:
38         self.logger.info("Match fields : ALL MATCH") # 匹配所有封包就直接印出來
39
40     self.logger.info("")
41     self.logger.info("Instructions fields : ") # 取得執行動作
42     for action in instructions:
43         OFPInstructionActions = action.to_jsondict()[list(action.to_jsondict().keys())[0]]
44         action_key = list(OFPInstructionActions.keys())[0]
45         actions = OFPInstructionActions[action_key]
46
47         self.logger.info(actions) # 打印執行動作
48
49     self.print_split_line("=", False) # 結束分割線
```