Lab-5 Observing Flow Table Overflows

1) What is the command for setting the flow-table size in Open vSwitch. Please explain the meaning of each option.

Ans: The command for setting the flow-table size in Open vSwitch is:

"ovs-vsctl add Bridge s1 flow_tables 0=@ash -- --id=@ash create flow_table flow_limit=100"

The meaning of each option is as follows

a) **ovs-vsctl:** The ovs-vsctl program configures ovs-vswitchd by providing a high-level interface to its configuration database.

b) **add Bridge s1:** add switch S1.

c) **flow_tables 0=@ash -- --id=@ash create flow_table:** Create flow table as 0.

d) **flow_limit=100**: set flow table size to 100.

2) Submit a snapshot showing that the table is configured to a bounded size.

Ans:

```
rakshit@rakshit-VirtualBox:~$ sudo ovs-vsctl add Bridge S1 flow_tables 0=@ash --
 --id=@ash create flow_table flow_limit=100
d2b13baa-428f-47eb-ba6fac167261
```

3) Submit the snapshot when you receive table overflow message sent by the controller.

Ans:

```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
EventOFPErrorMsg recieved.
version=0x4, msg_type=0x1, msg_len=0x4a, xid=0xed4d69b5
 '-- msg_type: OFPT_ERROR(1)
OFPErrorMsg(type=0x5, code=0x1, data='\x04\x0e\x00h\xf9\x95\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\xf4\xff\xff\xff\xf
f\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x1b\x80\x00\n\x02\x08\x00
\x80\x00\x14\x01\x06\x80')
 |--type: OFPET_FLOW_MOD_FAILED(5)
 |--code: OFPFMFC_TABLE_FULL(1)
 '--data: version=0x4, msg_type=0xe, msg_len=0x34, xid=0xed4d69b5
    '--msg_type: OFPT_FLOW_MOD(14)
```

4) Submit the PacketIn event call back method of your controller script. Add comments to clarify the logic if necessary.

Ans:

```
def _packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        in_port = msg.match['in_port']

#Reference how to identify the packet's protocols

        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocols(ethernet.ethernet)[0]
        eth_type = eth.ethertype

# process ARP

        if eth_type == ether.ETH_TYPE_ARP:
                self.handle_arp(datapath, in_port, pkt)
                return
        elif eth_type == ether.ETH_TYPE_IP:
                self.handle_ip(datapath, in_port, pkt)
        else:
                return

def handle_arp(self,datapath,in_port,pkt):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
                                # parse out the ethernet and arp packet
        eth_pkt = pkt.get_protocol(ethernet.ethernet)
        arp_pkt = pkt.get_protocol(arp.arp)    # obtain the MAC of dst IP
        arp_resolv_mac = self.arp_table[arp_pkt.dst_ip]

        new_pkt = packet.Packet()
        new_pkt.add_protocol(ethernet.ethernet(ethertype=eth_pkt.ethertype,
                                dst=eth_pkt.src,src=arp_resolv_mac))
        new_pkt.add_protocol(arp.arp(opcode=arp.ARP_REPLY,
                        src_mac=arp_resolv_mac,src_ip=arp_pkt.dst_ip,
                        dst_mac=arp_pkt.src_mac,dst_ip=arp_pkt.src_ip))
                # send the Packet Out mst to back to the
        new_pkt.serialize()
        actions = [parser.OFPActionOutput(in_port)]
        out = parser.OFPPacketOut(datapath,ofproto.OFP_NO_BUFFER,
                ofproto.OFPP_CONTROLLER,actions,new_pkt.data)

        datapath.send_msg(out)
```

```
def handle_ip(self,datapath,in_port,pkt):
        ofproto=datapath.ofproto
        parser=datapath.ofproto_parser
                        # parse out the IPv4 pkt
        eth_pkt=pkt.get_protocol(ethernet.ethernet)
        ipv4_pkt=pkt.get_protocol(ipv4.ipv4)
        tcp_pkt=pkt.get_protocol(tcp.tcp)  # parse out TCP packet
        if datapath.id == 1 and ipv4_pkt.proto==inet.IPPROTO_TCP:
                match=parser.OFPMatch(eth_type=0x0800,ip_proto=6,
                                      ipv4_src='10.0.0.1',
                                      ipv4_dst='10.0.0.2',
                                      tcp_src=tcp_pkt.src_port,
                                      tcp_dst=tcp_pkt.dst_port)
                actions=[parser.OFPActionOutput(2)]
        self.add_flow(datapath,10,match,actions)

        match=parser.OFPMatch(eth_type=0x0800,ip_proto=6,
                              ipv4_src='10.0.0.2',
                              ipv4_dst='10.0.0.1',
                              tcp_src=tcp_pkt.src_port,
                              tcp_dst=tcp_pkt.dst_port)
        actions=[parser.OFPActionOutput(1)]
        self.add_flow(datapath,10,match,actions)
```

5) Consider the implementation of flow table memory in Open vSwitch. What will happen if the table size is not set while millions of flow entries are inserted into the vSwitch.

Ans: When the table size is not set in Open, it is set to 1000 by default. So even if millions of flow entries were inserted into the vSwitch, table overflow message will be sent by the controller after 1000 entries in the flow table.

6) Name at least 3 consequences when flow table gets overflowed.

Ans:

1) Observed that no additional flow entries can be added subsequently the controller will continuously receive packets dependent on these flows. A controller-dependency performance hit is incurred.

2) Observed the controller will continuously attempt to add new flow entries and overflow the switch's flow table. Error messages from the switch is expected. A switch erroring performance hit is incurred.

3) Observed the packets dependent on these additional flow entries will be dropped as the switch will be unable to forward them.