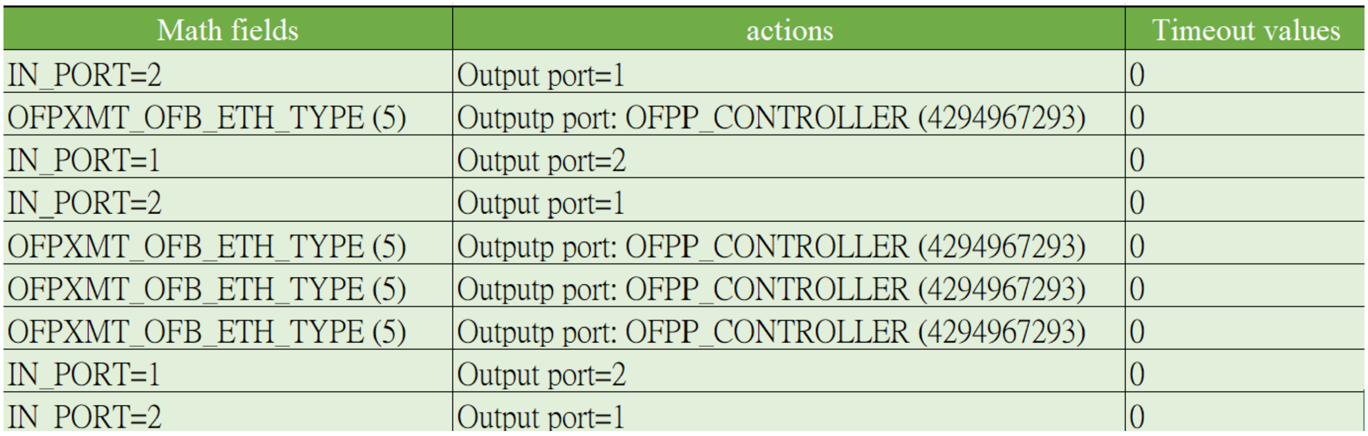
SDN/NFV Project2_310581027

Part 1: Answer Questions

1. How many OpenFlow headers with type “OFPT_FLOW_MOD” and command “OFPFC_ADD” are there among all the packets?
    - **Ans:** Totals are 9.
2. What are the match fields and the corresponding actions in each “OFPT_FLOW_MOD” message?
3. What are the timeout values for all flow rules on s1?

Report format



Math fields	actions	Timeout values
IN_PORT=2	Output port=1	0
OFPXMT_OFB_ETH_TYPE (5)	Outputp port: OFPP_CONTROLLER (4294967293)	0
IN_PORT=1	Output port=2	0
IN_PORT=2	Output port=1	0
OFPXMT_OFB_ETH_TYPE (5)	Outputp port: OFPP_CONTROLLER (4294967293)	0
OFPXMT_OFB_ETH_TYPE (5)	Outputp port: OFPP_CONTROLLER (4294967293)	0
OFPXMT_OFB_ETH_TYPE (5)	Outputp port: OFPP_CONTROLLER (4294967293)	0
IN_PORT=1	Output port=2	0
IN_PORT=2	Output port=1	0

Part 2 Install Flow Rules

Install Flow Rules (1/3)

- Please deactivate all the apps, except those initially activated. “org.onosproject.hostprovider”, “org.onosproject.lldpprovider”, “org.onosproject.optical-model”, “org.onosproject.openflow-base”, “org.onosproject.openflow”, “org.onosproject.drivers” and “org.onosproject.gui2”.
- Use the following topology (i.e. h1-s1-h2):

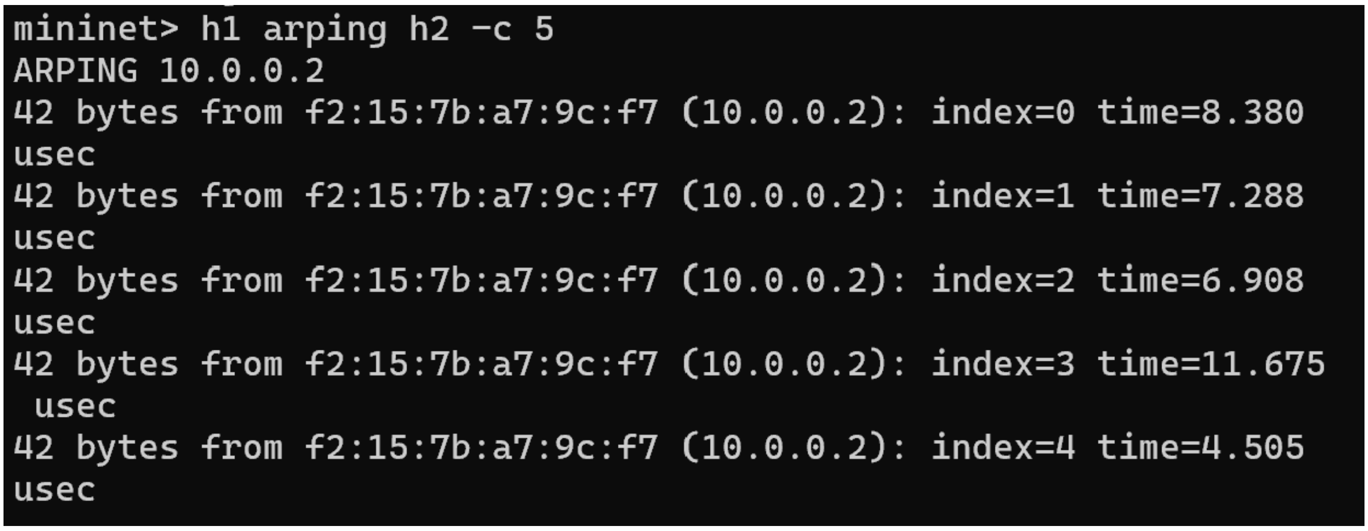
```
$ sudo mn --controller=remote,127.0.0.1:6653 --switch=ovs,protocols=OpenFlow14
```

Install Flow Rules (2/3)

- Install one flow rule to forward ARP packets with below condition
    - Match Fields
        - Ethernet type (ARP)
    - Actions
        - Forwarding ARP packets to all port in one instruction
- Verify the flow rules your installed and take screenshot

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x806"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "NORMAL"
      }
    ]
  }
}
```

```
mininet> h1 arping h2 # send ARP request
```



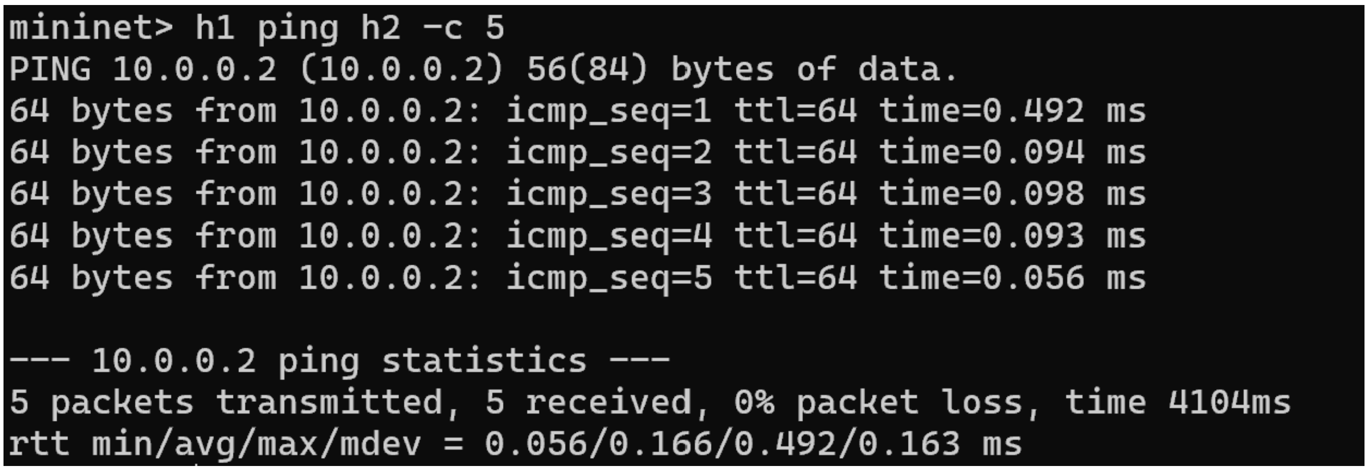
```
mininet> h1 arping h2 -c 5
ARPING 10.0.0.2
42 bytes from f2:15:7b:a7:9c:f7 (10.0.0.2): index=0 time=8.380
usec
42 bytes from f2:15:7b:a7:9c:f7 (10.0.0.2): index=1 time=7.288
usec
42 bytes from f2:15:7b:a7:9c:f7 (10.0.0.2): index=2 time=6.908
usec
42 bytes from f2:15:7b:a7:9c:f7 (10.0.0.2): index=3 time=11.675
 usec
42 bytes from f2:15:7b:a7:9c:f7 (10.0.0.2): index=4 time=4.505
usec
```

Install Flow Rules (3/3)

- Install two flow rules to forward IPv4 packets
  - Match Fields
    - IPv4 destination address and other required dependencies
  - Actions
    - Forwarding IPv4 packets to the right hostl
- Verify the flow rules your installed and take screenshot

```
{
  "priority": 50000,
  "timeout": 0,
  "isPermanent": true,
  "selector": {
    "criteria": [
      {
        "type": "ETH_TYPE",
        "ethType": "0x800"
      }
    ]
  },
  "treatment": {
    "instructions": [
      {
        "type": "OUTPUT",
        "port": "NORMAL"
      }
    ]
  }
}
```

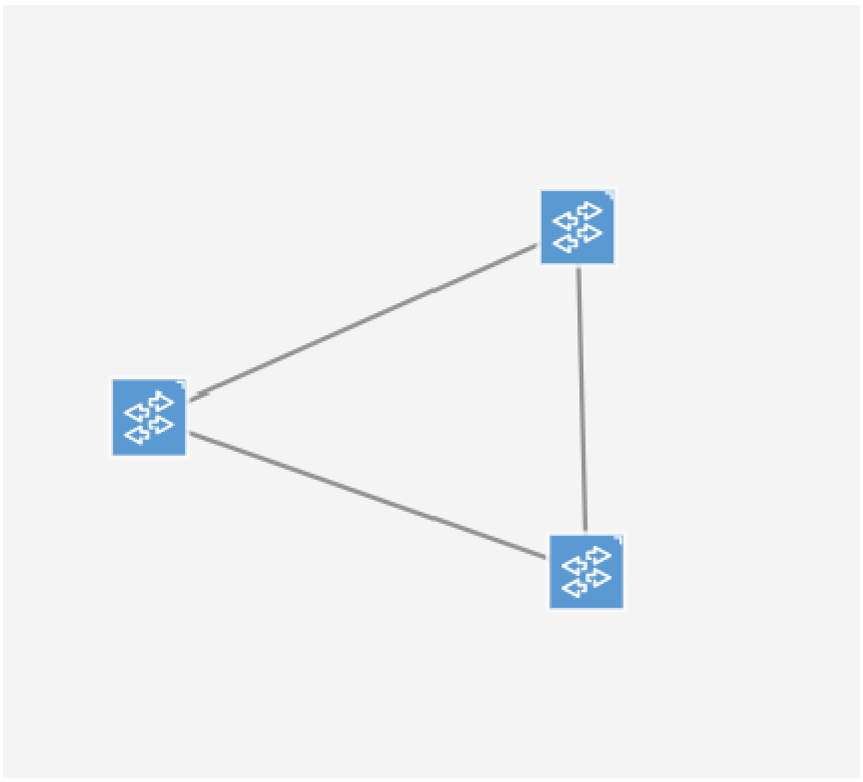
```
mininet> h1 arping h2 # send ICMP request
```



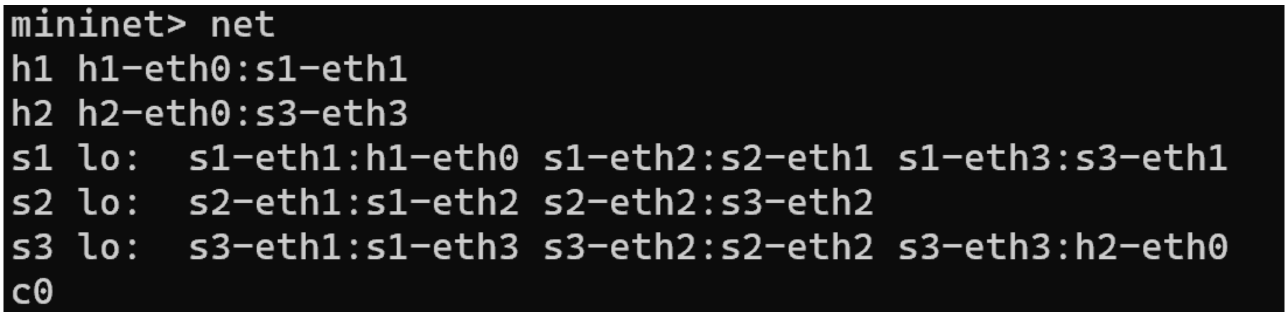
```
mininet> h1 ping h2 -c 5
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.492 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.056 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4104ms
rtt min/avg/max/mdev = 0.056/0.166/0.492/0.163 ms
```

Part 3: Create Topology with Broadcast Storm

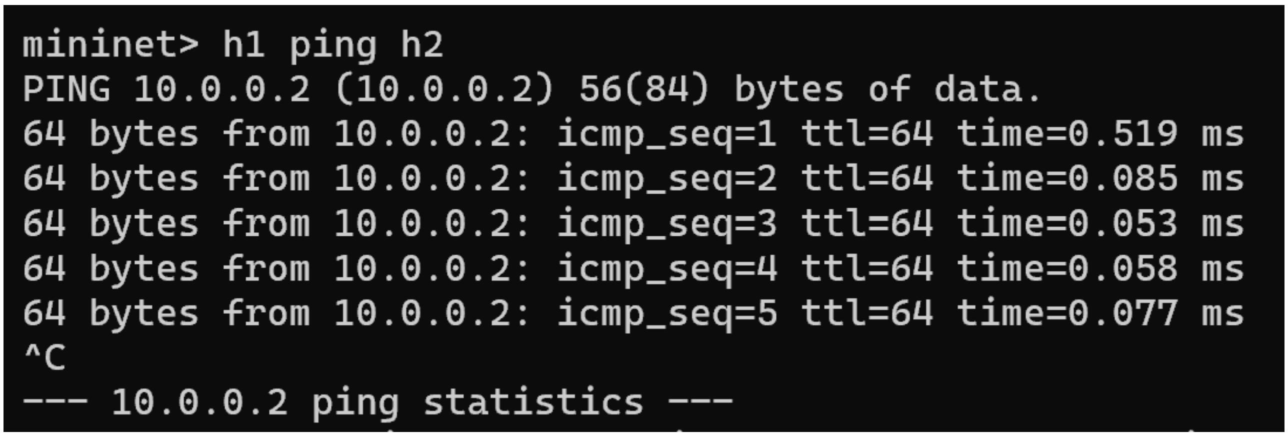


- **Topology of Broadcast Storm**
- Net



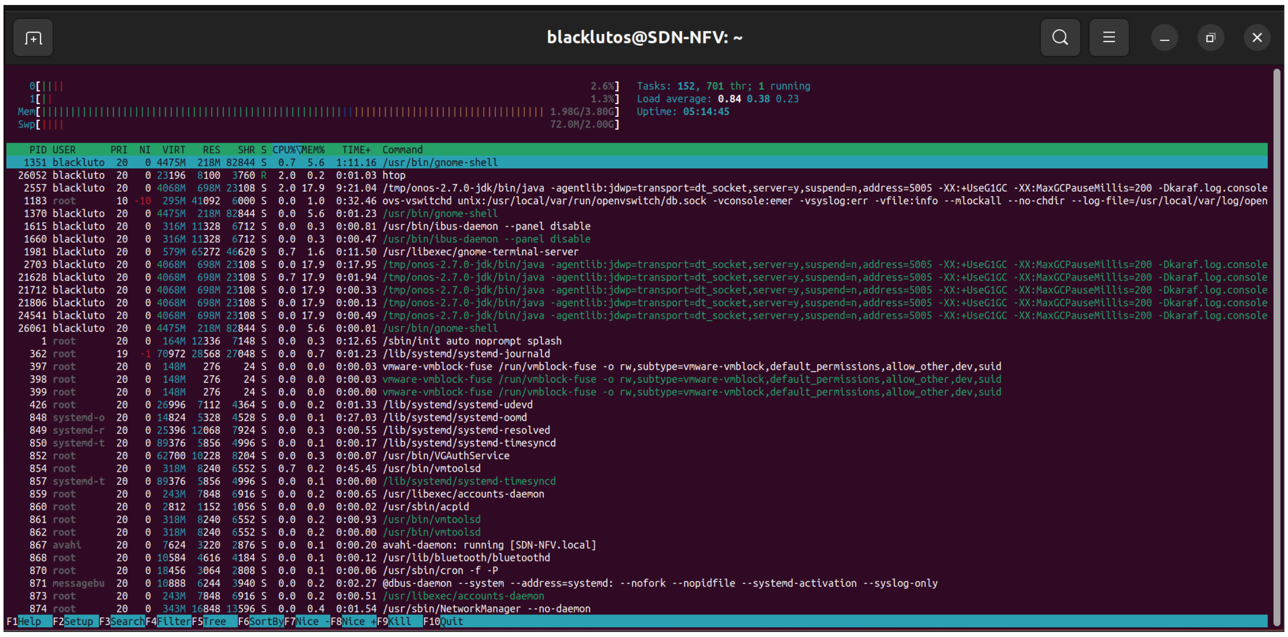
```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s3-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth1 s1-eth3:s3-eth1
s2 lo:  s2-eth1:s1-eth2 s2-eth2:s3-eth2
s3 lo:  s3-eth1:s1-eth3 s3-eth2:s2-eth2 s3-eth3:h2-eth0
c0
```

- Ping result



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.519 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.058 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.077 ms
^C
--- 10.0.0.2 ping statistics ---
```

- CPU's utilization



```
blacklutos@SDN-NFV: ~

    0[||||                                                                              2.6%]   Tasks: 152, 701 thr; 1 running
    1[||                                                                                1.3%]   Load average: 0.84 0.38 0.23
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 1.98G/3.80G]   Uptime: 05:14:45
  Swp[||||                                                                         72.0M/2.00G]

    PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
   1351 blackluto  20   0 4475M  218M 82844 S  0.7  5.6  1:11.16 /usr/bin/gnome-shell
  26052 blackluto  20   0 23196  8100  3760 R  2.0  0.2  0:01.03 htop
   2557 blackluto  20   0 4068M  698M 23108 S  2.0 17.9  9:21.04 /tmp/onos-2.7.0-jdk/bin/java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -Dkaraf.log.console
   1183 root       10 -10  295M 41092  6000 S  0.0  1.0  0:32.46 ovs-vswitchd unix:/usr/local/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir --log-file=/usr/local/var/log/open
   1370 blackluto  20   0 4475M  218M 82844 S  0.0  5.6  0:01.23 /usr/bin/gnome-shell
   1615 blackluto  20   0  316M 11328  6712 S  0.0  0.3  0:00.81 /usr/bin/ibus-daemon --panel disable
   1660 blackluto  20   0  316M 11328  6712 S  0.0  0.3  0:00.47 /usr/bin/ibus-daemon --panel disable
   1981 blackluto  20   0  579M 65272 46620 S  0.7  1.6  0:11.50 /usr/libexec/gnome-terminal-server
   2703 blackluto  20   0 4068M  698M 23108 S  0.0 17.9  0:17.95 /tmp/onos-2.7.0-jdk/bin/java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -Dkaraf.log.console
  21628 blackluto  20   0 4068M  698M 23108 S  0.7 17.9  0:01.94 /tmp/onos-2.7.0-jdk/bin/java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -Dkaraf.log.console
  21712 blackluto  20   0 4068M  698M 23108 S  0.0 17.9  0:00.33 /tmp/onos-2.7.0-jdk/bin/java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -Dkaraf.log.console
  21806 blackluto  20   0 4068M  698M 23108 S  0.0 17.9  0:00.13 /tmp/onos-2.7.0-jdk/bin/java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -Dkaraf.log.console
  24541 blackluto  20   0 4068M  698M 23108 S  0.0 17.9  0:00.49 /tmp/onos-2.7.0-jdk/bin/java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005 -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -Dkaraf.log.console
  26061 blackluto  20   0 4475M  218M 82844 S  0.0  5.6  0:00.01 /usr/bin/gnome-shell
      1 root       20   0  164M 12336  7148 S  0.0  0.3  0:12.65 /sbin/init auto noprompt splash
    362 root       19  -1 70972 28568 27048 S  0.0  0.7  0:01.23 /lib/systemd/systemd-journald
    397 root       20   0  148M   276    24 S  0.0  0.0  0:00.03 vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-vmblock,default_permissions,allow_other,dev,suid
    398 root       20   0  148M   276    24 S  0.0  0.0  0:00.03 vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-vmblock,default_permissions,allow_other,dev,suid
    399 root       20   0  148M   276    24 S  0.0  0.0  0:00.00 vmware-vmblock-fuse /run/vmblock-fuse -o rw,subtype=vmware-vmblock,default_permissions,allow_other,dev,suid
    426 root       20   0 26996  7112  4364 S  0.0  0.2  0:01.33 /lib/systemd/systemd-udevd
    848 systemd-o  20   0 14824  5328  4528 S  0.0  0.1  0:27.03 /lib/systemd/systemd-oomd
    849 systemd-r  20   0 25396 12068  7924 S  0.0  0.3  0:00.55 /lib/systemd/systemd-resolved
    850 systemd-t  20   0 89376  5856  4996 S  0.0  0.1  0:00.17 /lib/systemd/systemd-timesyncd
    852 root       20   0 62700 10228  8204 S  0.0  0.3  0:00.07 /usr/bin/VGAuthService
    854 root       20   0  318M  8240  6552 S  0.7  0.2  0:45.45 /usr/bin/vmtoolsd
    857 systemd-t  20   0 89376  5856  4996 S  0.0  0.1  0:00.00 /lib/systemd/systemd-timesyncd
    859 root       20   0  243M  7848  6916 S  0.0  0.2  0:00.65 /usr/libexec/accounts-daemon
    860 root       20   0  2812  1152  1056 S  0.0  0.0  0:00.02 /usr/sbin/acpid
    861 root       20   0  318M  8240  6552 S  0.0  0.2  0:00.93 /usr/bin/vmtoolsd
    862 root       20   0  318M  8240  6552 S  0.0  0.2  0:00.00 /usr/bin/vmtoolsd
    867 avahi      20   0  7624  3220  2876 S  0.0  0.1  0:00.20 avahi-daemon: running [SDN-NFV.local]
    868 root       20   0 10584  4616  4184 S  0.0  0.1  0:00.12 /usr/lib/bluetooth/bluetoothd
    870 root       20   0 18456  3064  2808 S  0.0  0.1  0:00.06 /usr/sbin/cron -f -P
    871 messagebu  20   0 10888  6244  3940 S  0.0  0.2  0:02.27 @dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
    873 root       20   0  243M  7848  6916 S  0.0  0.2  0:00.51 /usr/libexec/accounts-daemon
    874 root       20   0  343M 16848 13596 S  0.0  0.4  0:01.54 /usr/sbin/NetworkManager --no-daemon
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

- flows
  - inport=1,output=3, and ouput=1,input=3 on S1
  - inport=1,output=3, and ouput=1,input=3 on S3
  - h1 --> s1-eth1 --> s1-eth3 --> s3-eth3 --> s1-eth1 --> h2

Why the broadcast storm occurred

- Because we make the switches connect to each other, therefore, when the packet starts forwarding from host1, when it through s1, it has 2 entrance that needs to choose how to forward. So once the problem occurs, it will cause the loop, making the packet can't find the exit. Therefore, the packet will start forward in the loop, that's the reason why the problem occurs broadcast storm.

Part4: Trace ReactiveForwarding

- **Observation**
  - Arp request and reply to controller
    - h1 -> pakcet in -> controoler -> packet out -> h2 -> packet in -> controller -> packet out -> h1
    - when h1 ping h2, mac address of h1 don't know where is h2, so it packet in to controller to sned a arp request for finding h2's address. Then controller recived, and packout a arp request to start to brocast for find h2's mac address. When h2 received the arp request, it packet in to controller that wants to send a arp reply to h1, then controller recived the arp reply from h2, it packet out the arp reply to h1.
  - ICMP request and reply to controller
    - h1 -> pakcet in -> controoler -> packet out -> h2 -> packet in -> controller -> packet out -> h1
    - h1 know the address of h2, so it packet in to controller that want to send a icmp request to h2, and when controller received, it packet out the icmp request to h2, and while h2 received, h2 packet in the icmp reply to controller, and controller revied, packet out the icmp reply to h1.
- **TraceCode ReactiveForwarding application**
- Source Code

```
/**
* Sample reactive forwarding application.
*/
@Component(immediate = true)
@Service(value = ReactiveForwarding.class)
public class ReactiveForwarding {
  private static final int DEFAULT_TIMEOUT = 10;
  private static final int DEFAULT_PRIORITY = 10;
  private final Logger log = getLogger(getClass());
  //用於提供網絡拓補信息
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected TopologyService topologyService;

  //攔截封包，選擇要攔截哪些封包，或是做哪些動作
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected PacketService packetService;
  //用於與終端主機交互
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected HostService hostService;
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected FlowRuleService flowRuleService;
  //用於新增flow rule
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected FlowObjectiveService flowObjectiveService;
  //在使用onos的各種服務前，要先向CoreService註冊
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected CoreService coreService;
  //追蹤整個系統各個組件的配置
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected ComponentConfigService cfgService;
  @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
  protected StorageService storageService;
  //封包進來的時候，進行處理的handler
  private ReactivePacketProcessor processor = new
ReactivePacketProcessor();
  //metrics:記錄所有host的資訊
  private  EventuallyConsistentMap<MacAddress, ReactiveForwardMetrics> metrics;
  //ONOS下，每個app都要有自己的appId
  private ApplicationId appId;
  //一些設定
  @Property(name = "packetOutOnly", boolValue = false,
    label = "Enable packet-out only forwarding; default is false")
  private boolean packetOutOnly = false;
  @Property(name = "packetOutOfppTable", boolValue = false,
    label = "Enable first packet forwarding using OFPP_TABLE port " +"instead of
PacketOut with actual port; default is false")
  private boolean packetOutOfppTable = false;
  .
  .
  .
  //Entity capable of receiving network topology related events
  private final TopologyListener topologyListener = new
```

```
InternalTopologyListener();
  private ExecutorService blackHoleExecutor;
  //@Activate - marks a method as the method to call during the component startup
routine.
  @Activate
  public void activate(ComponentContext context) {
    //一個快速高效的反序列化框架
    KryoNamespace.Builder metricSerializer = KryoNamespace.newBuilder()
      .register(KryoNamespaces.API)
      .register(ReactiveForwardMetrics.class)
      .register(MultiValuedTimestamp.class);
    metrics =  storageService.<MacAddress,
ReactiveForwardMetrics>eventuallyConsistentMapBuilder()
      .withName("metrics-fwd")
      .withSerializer(metricSerializer)
      .withTimestampProvider((key, metricsData) -> new
    MultiValuedTimestamp<>(new WallClockTimestamp(), System.nanoTime()))
      .build();
    //topology有改變的話，就可以call(?)
    blackHoleExecutor =  newSingleThreadExecutor(groupedThreads("onos/app/fwd",
      "black-hole-fixer",
      log));
    //追蹤整個系統各個組件的配置
    cfgService.registerProperties(getClass());
    //註冊appId
    appId = coreService.registerApplication("org.onosproject.fwd");
    //ReactivePacketProcessor(這裡的變數是processor).process裡定義了接收到封包要做什
麼動作
    packetService.addProcessor(processor, PacketProcessor.director(2));
    //當拓墣改變的話，會呼叫
    topologyService.addListener(topologyListener);
    //In this way the application components can continue to use the standard and
well-understood OSGi configuration via ComponentContext.
    readComponentConfiguration(context);
    //想要攔截哪些種類的封包
    requestIntercepts();
    log.info("Started", appId.id());
  }
  /**
  * Request packet in via packet service.
  *
  * 在擷取的時候，指定想要擷取哪些封包
  */
  private void requestIntercepts() {
    //Abstraction of a slice of network traffic.
    //Builder of traffic selector entities.
    TrafficSelector.Builder selector = DefaultTrafficSelector.builder();
    selector.matchEthType(Ethernet.TYPE_IPV4);

    //要求讓ipv4的封包都packet in進來
    packetService.requestPackets(selector.build(), PacketPriority.REACTIVE,
appId);

    selector.matchEthType(Ethernet.TYPE_IPV6);
    if (ipv6Forwarding) {
      packetService.requestPackets(selector.build(),
PacketPriority.REACTIVE, appId);
    } else {
      packetService.cancelPackets(selector.build(),     PacketPriority.REACTIVE,
appId);
    }
  }

  /**
   * Cancel request for packet in via packet service.
   * 反正就是做跟requestIntercepts相反的事情
   */
  private void withdrawIntercepts() {
    TrafficSelector.Builder selector = DefaultTrafficSelector.builder();

   selector.matchEthType(Ethernet.TYPE_IPV4);
    packetService.cancelPackets(selector.build(),
      PacketPriority.REACTIVE, appId);
    selector.matchEthType(Ethernet.TYPE_IPV6);
    packetService.cancelPackets(selector.build(),
PacketPriority.REACTIVE, appId);
  }
  /**
   * Extracts properties from the component configuration context.
   *
   * 把一些設定從context裡面抽取出來
   * @param context the component context
   */
   private void readComponentConfiguration(ComponentContext context
   {
   }
  /**
  * Packet processor responsible for forwarding packets along their paths.
  * 這個function來決定要怎麼轉送封包
  */
  private class ReactivePacketProcessor implements PacketProcessor {
  //修改父母類別的function
    @Override
    public void process(PacketContext context) {
    }
  }
  // 判斷是不是 control packet(LLDP, BDDP)
  // Indicates whether this is a control packet, e.g. LLDP, BDDP
  private boolean isControlPacket(Ethernet eth) {
    short type = eth.getEtherType();
    return type == Ethernet.TYPE_LLDP || type == Ethernet.TYPE_BSN;
  }
  // 假如目前的點可以flood，那就flood
  // Floods the specified packet if permissible.
  private void flood(PacketContext context, ReactiveForwardMetrics macMetrics) {

if(topologyService.isBroadcastPoint(topologyService.currentTopology(),context.inPa
cket().receivedFrom())) {
      packetOut(context, PortNumber.FLOOD, macMetrics);
    } else {
      context.block();
    }
  }
  // Sends a packet out the specified port.
  // 送出封包到指定的port
  private void packetOut(PacketContext context, PortNumber portNumber,
ReactiveForwardMetrics macMetrics) {
  }

  // Install a rule forwarding the packet to the specified port.
  // 1.packetout, 2.flow_mod
  // flow_mod之後要packetout，這樣才不會把封包給平白無故地丟掉
  private void installRule(PacketContext context, PortNumber portNumber,
ReactiveForwardMetrics macMetrics) {
  }
```