目標

- 將特定型態的封包導向到同一個host

流程

- 判斷封包種類
- 尋找路徑
- 添加 flow entry
- 送出封包

判斷封包種類

- `Ethernet eth = IFloodlightProviderService.bcStore.get(cntx,IFloodlightProviderService.CONTEXT_PI_PAYLOAD);` //取得link layer封包
- `IPacket pkt = eth.getPayload();` //取得 link layer 封包 payload(network 封包)
- 用 instanceof 判斷封包種類ex: if (pkt instanceof ARP)，要注意要判斷的協定在第幾層
- net.floodlightcontroller.packet.UDP
- UDP. getDestinationPort()

尋找路徑

- 找出目標 switch, port
- 取得(switch, port) -> (switch, port)的路徑

找出目標 switch, port

- 利用 IDeviceService 提供的 getAllDevices() 回傳型態為 Collection<? extends IDevice>
- 利用 IDevice 提供的 getIPv4Addresses() 回傳型態為 Integer[]
  - 因為我們的環境一個 host 只有一個 IP 所以只判斷第 0 筆應該不會有影響
- 利用 IDevice 提供的 getAttachmentPoints() 回傳型態為 SwitchPort[] 取得 host 所連到的(switch, port)
  - 只取第 0 筆即可
- 利用 SwitchPort 提供的 getSwitchDPID() 跟 getPort() 就能取得 host 所連到的 switch 跟 port

取得(switch, port) -> (switch, port)的路徑

- 利用 IRoutingService 提供的 getRoute

```
Route routeIn =
routingEngine.getRoute(srcSwitchDPID, srcPort,
dstSwitchDPID, dstPort, 0);
```

添加 flow entry

- 設定 match
- Action: 修改封包 header
- Action: Forward 封包
- 寫入 flow entry

設定 match

- `OFMatch match = new OFMatch();`
- `OFPacketIn pin = (OFPacketIn) msg;`
- `match.loadFromPacket(pin.getPacketData(), pin.getInPort()); //`從 PACKET_IN 所帶的封包設定 match

Action: 修改封包 header

- `OFActionDataLayerDestination destMac = new OFActionDataLayerDestination( MACAddress.valueOf( getdstDev.getMACAddress()).toBytes());`
  - 設定目標MAC address
- `OFActionNetworkLayerDestination destIP = new OFActionNetworkLayerDestination( IPv4.toIPv4Address (“10.0.0.1”));`
  - 設定目標IP address

Action: Forward 封包

- 利用之前找到的路徑找出 output port
- Route 的內容是 (switch, port) 所組成的 list
- 所以第 1 個 switch 的 output port 是route 的第 1 項
- ```
  OFActionOutput out = newOFActionOutput(
      routeIn.getPath().get(1).getPortId());
  ```
- `out.setMaxLength((short)0xffff); //設定傳送封包大小`
  ```

寫入 flow entry

- 設定FlowMod
- List<OFAction> actions = new ArrayList<OFAction>();
- actions.add(destMac);
- actions.add(destIP);
- actions.add(out);
- OFFlowMod fm = (OFFlowMod) floodlightProvider.getOFMessageFactory().getMessage(OFType.FLOW_MOD);
- fm.setIdleTimeout(FLOWMOD_DEFAULT_IDLE_TIMEOUT)
  .setHardTimeout(FLOWMOD_DEFAULT_HARD_TIMEOUT)
  .setBufferId(OFPacketOut.BUFFER_ID_NONE)
  .setCookie((long) 0)
  .setPriority((short)1024)
  .setCommand(OFFlowMod.OFPFC_ADD)
  .setMatch(match)
  .setActions(actions)
  .setLengthU(OFFlowMod.MINIMUM_LENGTH
  +OFActionDataLayerDestination.MINIMUM_LENGTH
  +OFActionNetworkLayerDestination.MINIMUM_LENGTH
  +OFActionOutput.MINIMUM_LENGTH);

寫入 flow entry (cont.)

- 設定要match哪些欄位
- ```
  Integer wildcard_hints = ((Integer)
  sw.getAttribute(IOFSwitch.PROP_FASTWILDCARDS)).intValue()
          & ~OFMatch.OFPFW_IN_PORT
          & ~OFMatch.OFPFW_NW_SRC_MASK
          & ~OFMatch.OFPFW_NW_DST_MASK
          & ~OFMatch.OFPFW_TP_DST
          & ~OFMatch.OFPFW_NW_PROTO
          & ~OFMatch.OFPFW_DL_TYPE;
  ```
- fm.setMatch(wildcard(match, sw, wildcard_hints));
  ```

- protected OFMatch wildcard(OFMatch match, IOFSwitch sw, Integer wildcard_hints)
  {
  	if (wildcard_hints != null) {
  		return match.clone().setWildcards(wildcard_hints.intValue());
  	}
  return match.clone();
  }

寫入 flow entry (cont.)

- 利用OFMessageDamper提供的write()寫入FlowMod訊息
  - `messageDamper.write(sw, fm, cntx);`
- 利用迴圈將路徑上的所有switch都設定好output action
  - 除了第一個進來的switch，其他switch不用變更IP跟MAC address
  - 除了第一個進來的switch，其他switch match變更後的IP或MAC address

送出封包

- 將原封包使用PACKET_OUT送回網路中
- 可從別的module複製pushPacket function來使用
  - LearningSwitch 273 行
  - `void pushPacket(IOFSwitch sw, OFMatch match, OFPacketIn pi, short outport)`
- `pushPacket(ip_pkt, sw, msg.getBufferId(), msg.getInPort(), routeIn.getPath().get(1).getPortId(), cntx, true);`