MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

DEPARTMENT OF ICT

Assignment No : 02

Course Code	: ICT-4101
Course Title	: Telecommunication Engineering
Assignment name	: OpenFlow Protocol

Submitted by	***Submitted to***
Md. Faruk Hosen	Nazrul Islam
ID : IT-17035	Assistant Professor,
Session : 2016-2017	Department of ICT,MBSTU
Year : 4th Semester : 1st	Santosh,Tangail-1902

Date of Submission : 20 October 2020

OpenFlow Protocol

Objectives :

- Understand the working principles of OpenFlow protocol.
- Configure a basic Software Defined Network for end-to-end communications.
- Understand the difference between interacting with real and virtual networks.

1.What is OpenFlow protocol ?

Ans : **OpenFlow protocol** : OpenFlow is a protocol that allows a server to tell network switches where to send packets. In a conventional network, each switch has proprietary software that tells it what to do. With OpenFlow, the packet-moving decisions are centralized, so that the network can be programmed independently of the individual switches and data center gear.

2.What are the Main components of an OpenFlow switch ?

Ans: The main components of an OpenFlow switch is described below :

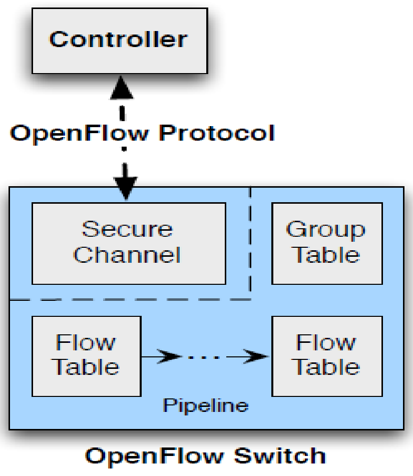


Figure : Main components of an OpenFlow switch

i. OpenFlow switch : An OpenFlow Switch consists of one or more flow tables and a group table, which perform packet lookups and forwarding, and an OpenFlow channel to an external controller. The switch communicates with the controller and the controller manages the switch via the OpenFlow protocol.

ii. OpenFlow protocol : Using the OpenFlow protocol, the controller can add, update, and delete flow entries in flow tables, both reactively (in response to packets) and proactively. Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets.

3. How does OpenFlow Protocol work?

Ans : Using the OpenFlow protocol, the controller can add, update, and delete flow entries in flow tables, both reactively (in response to packets) and proactively. Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets.

Matching starts at the first flow table and may continue to additional flow tables. Flow entries match packets in priority order, with the first matching entry in each table being used. If a matching entry is found, the instructions associated with the specific flow entry are executed. If no match is found in a flow table, the outcome depends on configuration of the table-miss flow entry: for example, the packet may be forwarded to the controller over the OpenFlow channel, dropped, or may continue to the next flow table.

Instructions associated with each flow entry either contain actions or modify pipeline processing. Actions included in instructions describe packet forwarding, packet modification and group table processing. Pipeline processing instructions allow packets to be sent to subsequent tables for further processing and allow information, in the form of metadata, to be communicated between tables. Table pipeline processing stops when the instruction set associated with a matching flow entry does not specify a next table; at this point the packet is usually modified and forwarded.

Flow entries may forward to a port. This is usually a physical port, but it may also be a logical port defined by the switch or a reserved port defined by this specification. Reserved ports may specify generic forwarding actions such as sending to the controller, flooding, or forwarding using non-OpenFlow methods, such as normal switch processing while switch-defined logical ports may specify link aggregation groups, tunnels or loopback interfaces.

Switch designers are free to implement the internals in any way convenient, provided that correct match and instruction semantics are preserved. For example, while a flow entry may use an all group to forward to multiple ports, a switch designer may choose to implement this as a single bitmask within the hardware forwarding table. Another example is matching; the pipeline exposed by an OpenFlow switch may be physically implemented with a different number of hardware tables.

4. How a switch hub works and how it is implemented using OpenFlow ?

Ans : Switching Hub by OpenFlow : OpenFlow switches can perform the following by receiving instructions from OpenFlow controllers such as Ryu:

i. Rewrites the address of received packets or transfers the packets from the specified port.
ii. Transfers the received packets to the controller (Packet-In).
iii. Transfers the packets forwarded by the controller from the specified port (Packet-Out).

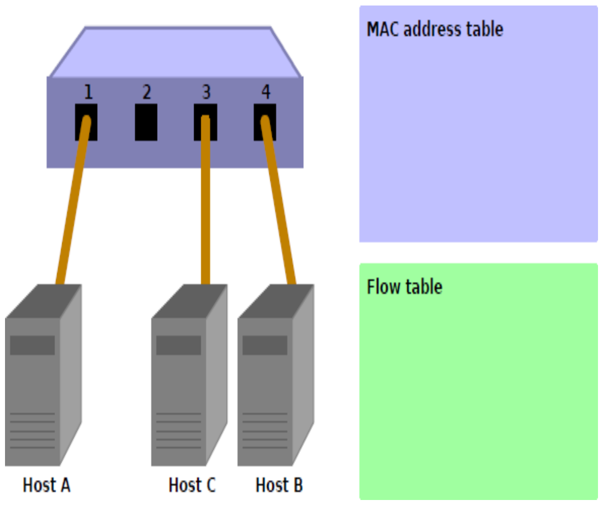
First of all, we need to use the Packet-In function to learn MAC addresses. The controller can use the Packet-In function to receive packets from the switch. The switch analyzes the received packets to learn the MAC address of the host and information about the connected port.

After learning, the switch transfers the received packets. The switch investigates whether the destination MAC address of the packets belong to the learned host. Depending on the investigation results, the switch performs the following processing.

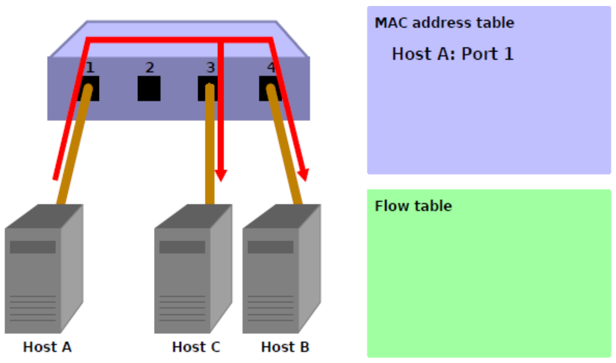
- If the host is already a learned host ... Uses the Packet-Out function to transfer the packets from the connected port.
- If the host is unknown host ... Use the Packet-Out function to perform flooding.

The following explains the above operation in a step-by-step way using figures.

i. Initial status : This is the initial status where the flow table is empty.



ii. Host A -> Host B : When packets are sent from host A to host B, a Packet-In message is sent and the MAC address of host A is learned by port 1.



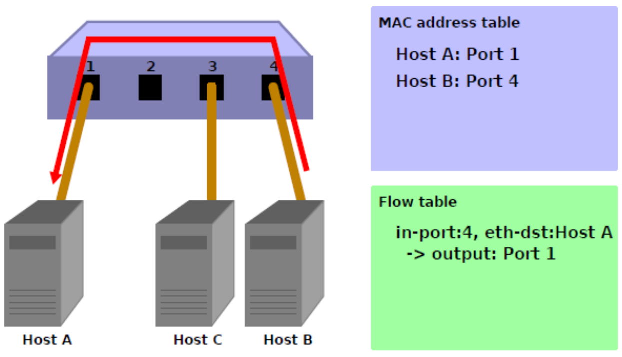
Packet-In:

```
in-port: 1
eth-dst: Host B
eth-src: Host A
```

Packet-Out:

```
action: OUTPUT:Flooding
```

iii. <u>Host B -> Host A :</u> When the packets are returned from host B to host A, an entry is added to the flow table and also the packets are transferred to port 1.



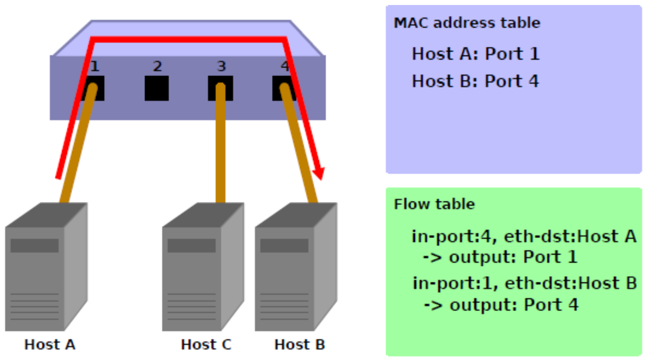
Packet-In:

```
in-port: 4
eth-dst: Host A
eth-src: Host B
```

Packet-Out:

```
action: OUTPUT:Port 1
```

iv. <u>Host A -> Host B :</u> Again, when packets are sent from host A to host B, an entry is added to the flow table and also the packets are transferred to port 4.



Packet-In:

```
in-port: 1
eth-dst: Host B
eth-src: Host A
```

Packet-Out:

```
action: OUTPUT:Port 4
```

5.Describe the OpenFlow specification terms ?

Ans : The key OpenFlow specification terms :

- **Byte:** an 8-bit octet.
- **Packet:** an Ethernet frame, including header and payload.
- **Port:** where packets enter and exit the OpenFlow pipeline.
- **Pipeline:** the set of linked flow tables that provide matching, forwarding, and packet modifications in an OpenFlow switch.
- **Flow Table:** A stage of the pipeline, contains flow entries.
- **Flow Entry:** an element in a flow table used to match and process packets.
- **Match Field:** a field against which a packet is matched, including packet headers, the ingress port, and the metadata value.
- **Metadata:** a maskable register value that is used to carry information from one table to the next.
- **Action:** an operation that forwards the packet to a port or modifies the packet, such as decrementing the TTL field.
- **Group:** a list of action buckets and some means of choosing one or more of those buckets to apply on a per-packet basis.
- **Action Bucket**: a set of actions and associated parameters, defined for groups.
- **Tag:** a header that can be inserted or removed from a packet via push and pop actions.
- **Outermost Tag:** the tag that appears closest to the beginning of a packet.
- **Controller:** An entity interacting with the OpenFlow switches using the OpenFlow protocol.

6. Using Graphical interface of the RYU controller. How you create the ryu.app.gui_topology which provides topology visualization ?

Ans: First Run mininet in terminal-1 using the following script (save as remote_controller_mininet_tree.py).

```
from  mininet.node  import  CPULimitedHost
from  mininet.topo  import  Topo
from  mininet.net  import  Mininet
from  mininet.log  import  setLogLevel,  info
from  mininet.node  import  RemoteController
from  mininet.cli  import  CLI
from  mininet.link  import  TCLink

class  GenericTree(Topo):
    def init(self, depth=1, fanout=2):
        Topo.init(self)
        self.hostNum = 1
        self.switchNum = 1
        self.addTree(depth, fanout)
    def  addTree(  self,  depth,  fanout  ):
        isSwitch = depth > 0
    if isSwitch:
        node = self.addSwitch('s%s' % self.switchNum)
    self.switchNum += 1
    for _ in range(fanout):
        child = self.addTree(depth - 1, fanout)
    self.addLink(node, child)
    else:
        node  =  self.addHost(  'h%s'  %  self.hostNum  )
        self.hostNum  +=  1 return node
def run():
    c = RemoteController('c', '0.0.0.0', 6633)
    net = Mininet(topo=GenericTree(depth=3, fanout=2),
host=CPULimitedHost, controller=None)
    net.addController(c)
    net.start()
    net.stop()
    setLogLevel('info')
    run()
```

<u>Run RYU GUI application in Terminal-1:</u>

```
sudo  python      /usr/local/bin/ryu  run    --observe-link
      /home/user/Desktop/ryu-
master/ryu/app/gui_topology/gui_topology.py
```

Access http://<ip address of ryu host>:8080 with your web browser.

Example: http://0.0.0.0:8080/

<u>Conclusion :</u> From this assignment, we Understand the working principles of OpenFlow protocol. We also know how to Configure a basic Software Defined Network for end-to-end communications. Software-defined network (SDN) is a new programmable networking designed to perform tasks easier by enabling network administrators to add network control via a centralized control platform and open interfaces. Common network use procedures such as traffic shifts, troubleshooting and various types of device configuration. Thus devices are needed to reconfigure with the network, in order to create a reaction with events. Here we have learned how to developed an SDN testbed using Zodiac FX a hardware switch for OpenFlow experiment. This is utilized Zodiac FX a hardware switch to test the usage and discuss about the SDN controllers. Ryu controller is configured for testbed development. Last of all, we understand the difference between interacting with real and virtual networks.