MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh, Tangail -1902

Assignment No	**:**	**02**
Assignment Name	**:**	OpenFlow Protocol
Course Name	**:**	Telecommunication Engineering
Course Code	**:**	ICT - 4101

Submitted by,

Name : Md. Imtyaz Ahmed

ID: IT-17017

Session: 2016-17

Dept. of ICT, MBSTU.

Submitted to,

NAZRUL ISLAM

Assistant Professor

Dept. of ICT, MBSTU.

OpenFlow Protocol

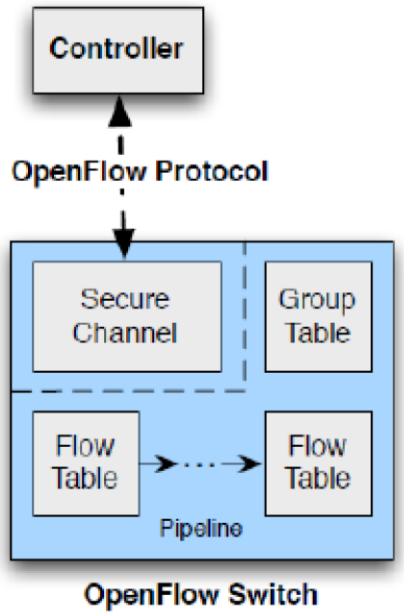
Objectives :

- 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.

OPENFLOW PROTOCOL:

Theory

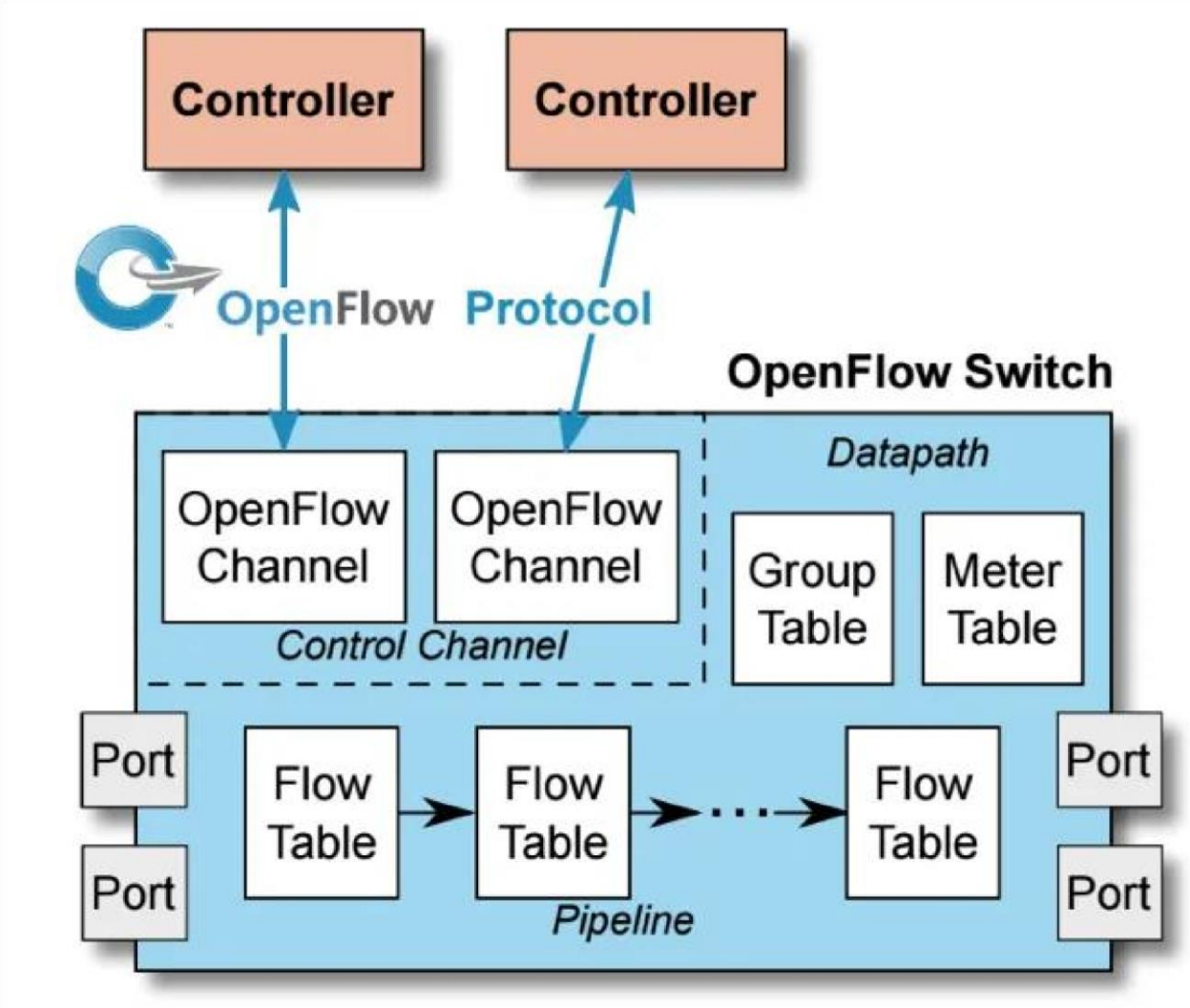
This speciation covers the components and the basic functions of the switch, and the OpenFlow protocol to manage an OpenFlow switch from a remote controller.



Components of 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 (Figure 2-1). The switch communicates with the controller and the controller manages the switch via the 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.



QUESTIONS:

Question 5.1:How the RYU GUI interface can be improved? Provide some ideas.

Answer: Web design consists, for the most part, of interface design. There are many techniques involved in crafting beautiful and functional interfaces. Here's my collection of 10 that I think you'll find useful in your work. They're not related to any particular theme, but are rather a collection of techniques I use in my own projects. Without further ado, let's get started.

1. **Padded block links**

Links (or anchors) are inline elements by default, which means that their clickable area spans only the height and width of the text. This clickable area, or the space where you can click to go to that link's destination, can be increased for greater usability. We can do this by **adding padding** and, in some cases, also **converting the link into a block element**.

2. **Typesetting buttons**

Attention to every detail is what separates a great product from a mediocre one. Interface elements such as buttons and tabs are clicked on many times a day by your users, so it pays to typeset them properly; and by typesetting I mean positioning the label.

3. **Using contrast to manage focus**

Similarly, you can also manage the focus of your visitors' attention with contrast between elements.

4. **Using color to manage attention**

Color can also be used to effectively **focus your visitors' attention on important or actionable elements**. For example, during the US presidential election, pretty much all of the candidates' websites had the donation button colored red. Red is a very bright and powerful color so it attracts attention, and it stands out even more when the rest of the website is blue or another colder color.

Warmer tones like red, yellow and orange are naturally bright and so tend to attract the eye. They also "expand" when set against colder colors like blue and green. This means that an orange button on a blue background looks like it's flowing outwards and taking the front seat. Conversely, a blue button on an orange background contracts inward, wishing to stay in the background.

5. White space indicates relationships

One of the most crucial elements in an interface is the white space between elements. If you're not familiar with the term white space, it means just that: space between one interface element and another, be it a button, a navigation bar, article text, a headline and so on. By manipulating white space, we can indicate relationships between certain elements or groups of elements.

So, for example, by putting the headline near the article text we indicate that it is related to that text. The text is then placed farther away from other elements to separate it and make it more readable.

6. Letter spacing

Web design is pretty limiting for typographers. But while there are only a few safe Web fonts and not a great many things you can do to style them, it's worth remembering that we do still have some level of control. "Tracking" is a term used in the field of typography to describe the adjustment of **spacing between letters in words**. We've got the ability to do this with CSS using the letter-spacing property.

If used with restraint and taste, this property can be effective in improving the look of your headlines. I wouldn't recommend using letter spacing on the body text because the default spacing generally provides the best readability for smaller font sizes.

7. Auto-focus on input

Many Web applications and websites feature forms. These may be search forms or input forms inviting you to submit something. If this form is the **core feature** of your application or website, you may want to consider **automatically focusing the user's cursor on the input field** when the website loads. This will speed things up because users can start typing right away without having to click on it.

8. Custom input focus

While the default look of form elements suffices for most functions, sometimes we want something a little prettier or a little more standardized across various browsers and systems. We can style input fields by simply targeting it with an "id," "class" or plain old "input," like so.

9. Hover controls

Some Web applications have extra utility controls, such as edit and delete buttons, that don't necessarily have to be shown beside every item at all times. They can be hidden to **simplify the interface and focus visitors' attention on the main controls and content**.

10. Verbs in labels

You can make options dialogs much more usable by thinking through the labels you use on buttons and links. If an error or message pops up and the options are "Yes," "No" and "Cancel," you have to read the whole message to be able to answer. Seems normal, right?

But we can actually speed things up by using verbs in the labels. So, if instead of "Yes," "No" and "Cancel," we have "Save," "Don't Save" and "Cancel" buttons, you wouldn't even need to read the message to understand what the options are and which action to perform. **All the information is contained in the button labels.**

Question 5.2: Explain at least two the advantage and disadvantage of using mininet or Zodiac FX?

Answer: Mininet combines many of the best features of emulators, hardware testbeds, and simulators.

Compared to full system virtualization based approaches, Mininet:

- **Boots faster**: seconds instead of minutes
- **Scales larger**: hundreds of hosts and switches vs. single digits
- **Provides more bandwidth**: typically 2Gbps total bandwidth on modest hardware
- **Installs easily**: a prepackaged VM is available that runs on VMware or VirtualBox for Mac/Win/Linux with OpenFlow v1.0 tools already installed.

Compared to hardware testbeds, Mininet

- is **inexpensive** and **always available** (even before conference deadlines)
- is **quickly reconfigurable and restartable**

Compared to simulators, Mininet

- runs **real, unmodified code** including application code, OS kernel code, and control plane code (both OpenFlow controller code and Open vSwitch code)
- easily **connects to real networks**
- offers **interactive performance** - you can type at it

Some of the advantages of Mininet are:

• Ease of creating a network topology

• Fast and efficient

• Run real programs

• Customize packet forwarding

• Runs on any platform (Virtual or Native)

Disadvantages of Mininet:

Mininet-based networks cannot (currently) exceed the CPU or bandwidth available on a single server.

Mininet cannot (currently) run non-Linux-compatible OpenFlow switches or applications; this has not been a major issue in practice.

Question 5.3: Explain how the open flow tables are created?

Answer: Using the OpenFlow switch protocol, the controller can add, update, and delete flow entries in *flow tables*, both reactively (in response to packets) and proactively.

Reactive Flow Entries are created when the controller dynamically learns where devices are in the topology and must update the flow tables on those devices to build end-to-end connectivity. For example, since the switches in a pure OpenFlow environment are simply forwarders of traffic, all rational logic must first be dictated and programmed by the controller. So, if a host on switch A needs to talk to a host switch B, messages will be sent to the controller to find out how to get to this host. The controller will learn the host MAC address tables of the switches and how they connect, programming the logic into the *flow tables* of each switch. This is a reactive flow entry.

Proactive Flow Entries are programmed before traffic arrives. If it's already known that two devices should or should not communicate, the controller can program these *flow entries* on the OpenFlow endpoints ahead of time.

Question 5.6: 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()
```

Conclusion:

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.Open Flow switch addresses bottlenecks to high performance and scalability in SDN environment. Providing an efficient, vendor-independent approach to managing complex networks with dynamic demands, it is likely to become commonplace in large carrier networks, cloud infrastructures, and other networks.

The END