

CS4055 High Performance Data Networking

---

**IMPLEMENTATION OF IEEE 802.1Q VLAN  
TAGGING USING RYU OPENFLOW CONTROLLER**

---

January 16, 2016

VARUN NAIR

Student Number: 4504550

Department of Electrical Engineering

Delft University of Technology

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Local Area Networks and their Limitations . . . . .	3
1.2	The Virtual Local Area Network (VLAN) Concept . . . . .	4
1.3	IEEE 802.1Q Standard . . . . .	5
1.4	SDN Implementation of 802.1Q VLAN Tagging . . . . .	5
<b>2</b>	<b>IEEE 802.1Q Standard</b>	<b>7</b>
2.1	VLAN Header (Tag) Format . . . . .	7
2.2	VLAN ID Classification . . . . .	8
<b>3</b>	<b>OpenFlow 1.3 VLAN Related Specifications</b>	<b>8</b>
<b>4</b>	<b>802.1Q Implementation in OpenvSwitch</b>	<b>9</b>
<b>5</b>	<b>802.1Q Implementation in RYU</b>	<b>11</b>
5.1	VLAN-Only Network with Single Trunk Line per Switch . . . . .	13
5.2	VLAN-Only Network with Multiple Trunk Lines per Switch . . . . .	14
5.3	Hybrid Network with VLAN and Non-VLAN aware ("Native" VLAN) Nodes . . .	15
<b>6</b>	<b>Results</b>	<b>18</b>
6.1	VLAN-Only Network with Single Trunk Line per Switch . . . . .	18
6.1.1	Ping Reachability Test . . . . .	18
6.1.2	Flow Entries in Switch . . . . .	18
6.1.3	Wireshark Packet Capture . . . . .	19
6.2	VLAN-Only Network with Multiple Trunk Lines per Switch . . . . .	20
6.2.1	Ping Reachability Test . . . . .	20
6.2.2	Flow Entries in Switch . . . . .	20
6.2.3	Wireshark Packet Capture . . . . .	21
6.3	Hybrid Network with VLAN and Non-VLAN aware ("Native" VLAN) Nodes . . .	22
6.3.1	Ping Reachability Test . . . . .	22
6.3.2	Flow Entries in Switch . . . . .	22
<b>7</b>	<b>Summary and Conclusions</b>	<b>23</b>

---

## ABSTRACT

In this project, a RYU application is developed which divides a given physical network into logical networks using Virtual Local Area Networks (VLANs) and implements IEEE 802.1Q VLAN tagging to enable communication between nodes(members) of the same VLAN. The RYU controller is based on OpenFlow 1.3 and interacts with switches running OpenvSwitch 1.2. For simplicity, it is assumed that the topology and the required VLAN configuration (for each switch) is known priory and therefore no topology discovery is carried out in this application. The RYU application is tested for 3 types of networks: A VLAN-Only linear (2 Switch-4 Host) topology, A VLAN-only topology containing multiple switches and trunk lines and lastly a hybrid network with VLAN aware and Non-VLAN aware switches. The application has been developed such that it is scalable with higher number of nodes and compatible with any kind of VLAN configuration. A Software Defined Networking (SDN) based approach towards VLAN configuration is found to be more efficient in implementing configuration changes in a large network, as compared to when using a conventional distributed approach.

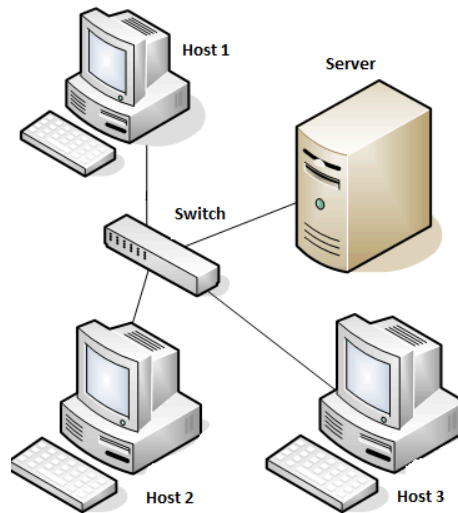
## 1 INTRODUCTION

### 1.1 Local Area Networks and their Limitations

In computer network terminology, a Local Area Network (LAN) is a network of hosts which spans a relatively small and limited geographical area. LANs are commonly seen in offices, homes and university campuses. The primary utility of LANs is for sharing of resources (data) amongst the hosts and for fast access of relevant data (stored on the server). Fig.1 shows an example of a LAN with 3 hosts and 1 server connected by a switch. An important property of a LAN is that hosts connected across the same switch are in the same "broadcast" or "collision" domain i.e broadcast traffic sent by any 1 host is received by all the other hosts and data sent by any 1 host has the likelihood of colliding with data of remaining 2 hosts who might be transmitting simultaneously.

Because of the "broadcast" domain property, the main disadvantage of using conventional physical LANs is that they do not scale well. For example, if there are 10 departments in an office, then they would each need 1 switch for interconnection of their respective computers and a router for interconnecting the switches and for blocking broadcast traffic generated from any of the switches.

The second limitation of physical LAN is that it does not allow hosts which are geographically far apart to be part of the same LAN unless additional hardware is put in place.

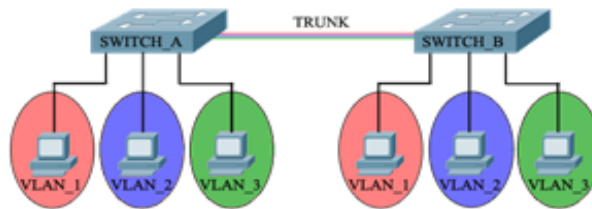


**Figure 1:** Local Area Network. All the hosts are in the same Broadcast domain

## 1.2 The Virtual Local Area Network (VLAN) Concept

As seen in the previous section, to partition different broadcast domains in a LAN, a router and multiple switched were required. This adds significantly to the network infrastructure cost.

To avoid this, the concept of Virtual Local Area Networks(VLANs) was introduced.VLANs are a way to separate a physical network into different logical networks (or broadcast domains) without the need for additional hardware or the requirement of hosts to be physically co-located. Therefore, hosts that are on separate physical networks can be logically grouped together as if they were behind the same switch. Users on different floors of the same building, or even in different buildings therefore belong to the same "network" and are "separated" from non-member hosts.



**Figure 2:** VLAN Configured Physical Network

Fig.2 shows an example of VLAN configuration in a basic network. Although, only 2 switches are used, 3 logical networks (or "broadcast" domains) have been created ,each containing two hosts on physically separate switches. Only members of the same VLAN can

---

communicate with each other and their broadcast traffic does not reach non-member hosts although they may be connected to the same switch.

Similarly, hosts connected to the same switch can be isolated from each other by assigning them to different VLANs.

The main advantages of [1] using VLANs are: -

1. **Improvement in performance:** VLANs avoid the need to use routers for creating separate broadcast domains. This increases performance since routers, being software-based, have relatively slower switching speeds as compared to hardware-based switches.
2. **Reduction in Cost:** VLANs help in reusing the existing hardware to create separate logical networks
3. **Easier Administration:** If a particular host needs to move to another physical location, unnecessary reconfiguration of routers is avoided.
4. **Security:** Only hosts of the same VLAN can access each other's data.

### 1.3 IEEE 802.1Q Standard

The 802.1Q is an "open" standard developed by IEEE [2] aimed at standardizing the operation of bridged networks. A key aspect of this standard is the specification on the use of VLANs in bridged networks.

As per this standard, the VLAN membership of a packet generated from a particular host can be designated by adding a 4-byte 802.1Q header just after the Source MAC address in the Ethernet frame. This is referred to as a "Tagging" operation. Within this 4-byte header, a 12-bit field is dedicated for denoting the VLAN id of the packet. Therefore, a maximum of 4096 VLANs are supported in this standard. Similarly, while sending a packet towards a host of a particular VLAN id, the VLAN header must be removed before outputting through the designated port. This operation is referred to as "Untagging".

The standard is further discussed in detail in Section 2.

### 1.4 SDN Implementation of 802.1Q VLAN Tagging

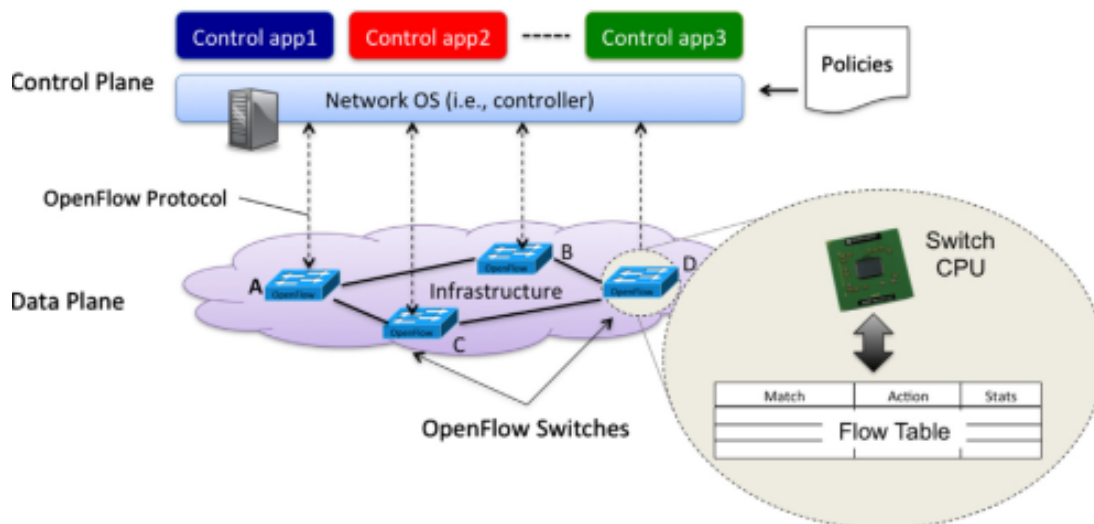
In this project, the 802.1Q operations of VLAN tagging and untagging are demonstrated using Software Defined Networking (SDN) concepts. The principal concept [3] involved in SDN is

the separation of the control and forwarding (or data) planes. The control plane may interact with several forwarding devices (OpenFlow Switches) and coordinate their actions (through Flow Tables and flow entries).

OpenFlow [4] is a protocol developed by the Open Networking Foundation for interaction between the "controller" (Control Plane) and the forwarding planes. Till now, 4 versions of the OpenFlow standard have been released. This project is based on version 1.3.

RYU [5] is a python-based framework (developeped and maintained by NTT Japan) for development of OpenFlow compatible controller applications.

A sample schematic of an SDN-based network is shown Fig.3



**Figure 3:** Schematic of an SDN based Network

In this project, the aim is develop a RYU based application to simulate the 802.1Q operations on VLAN tagging and untagging. For simplicity, it is assumed that the topology of the network and the VLAN configuration is known beforehand . Therefore, topology discovery functionality is not included in this application. The VLAN configuration is hard-coded within the application before it is run.

The remainder of the document is structured as follows: Section 2 provides an overview of the 802.1Q standard specifications on VLAN tagging. Section 3 discusses the relevant OpenFlow 1.3 guidelines about VLAN tagging and supported features. Section 4 demonstrates the 802.1Q implementation in OpenvSwitch. Section 5 describes the RYU controller implementation. Finally, results and conclusions are presented in Sections 6 and 7 respectively.

---

## 2 IEEE 802.1Q STANDARD

The 802.1Q standard [2] was developed by IEEE for describing the operation of Virtual Bridged Networks. Specifications for management of VLANs are also included as part of this standard.

Section 9.1 describes the purpose of VLAN tagging as two-fold: -

1. To allow a particular VLAN ID (VID) to be conveyed throughout the network and facilitate segregation or classification based on the VID
2. To allow some type of priority to be conveyed in the Ethernet frame

### 2.1 VLAN Header (Tag) Format

Based on the above two objectives, the VLAN header format is proposed to contain following:

-

1. **Tag Protocol Identifier (TPID)**: Containing the ether type indicating the type of 802.1Q header. 3 types of VLAN tags are specified: -
  - (a) C-Tag (Customer Tag): **Ether Type 0x8100**
  - (b) S- Tag (Service Tag): **Ether Type 0x88a8**
  - (c) I-Tag (Backbone Service Instance Tag): **Ether Type 0x88e7**
2. **Tag Control Information (TCI)**: Containing Priority and VLAN ID information.



**Figure 4:** VLAN TCI Format [2]

The VLAN TCI field (Fig 4) is a 16-bit field and contains the 3-bit Priority Code Point (PCP), 1-bit Drop Eligible Indicator (DEI) field and the 12-bit VLAN ID(VID) field.

VID value of 0 (Indicating null VID), 1 (default port VLAN ID), 2 (Default service VLAN id) and FFF (for wildcard matching) are reserved values having special meaning.

3. **Additional Information as required by tag type and TCI**

---

## 2.2 VLAN ID Classification

The VLAN membership can be done in the following ways [1]: -

1. **Port-based classification:** Configuring VLAN on a per port basis. The switch maintains an association of port and its respective VLAN membership so that any incoming packet will be tagged with that particular ID.
2. **MAC-address classification:** The switch tracks the MAC address of the source and accordingly tags it with the respective ID irrespective of the ingress port.
3. **Protocol-based classification:** VLAN ID is assigned based on the type of Layer 3/4 protocol present in the packet as mentioned in the Ether Type field.
4. **IP subnet based classification:** Based on the IP address /subnet of the incoming packet.

In this project, the implementation is based on Port-based classification which is the most basic type.

## 3 OPENFLOW 1.3 VLAN RELATED SPECIFICATIONS

The Open Flow specifications prior to version 1.3 [4] had limited support for VLAN related operations and only supported single level tagging.

Version 1.3 has introduced support for 802.1Q VLAN tag addition, modification , stripping and also QinQ (double tag) operation. The new operations/fields introduced are below: -

1. **Push Tag:** This will allow a new VLAN tag to be inserted (on top an existing header if present) with a default VID of 0.
2. **Action -> Set Field:** This will allow modification of TCI fields such as VID (for assigning a VLAN ID) or PCP (for assigning or changing priority).
3. **Pop Tag:** This removes/strips the outermost tag present.
4. **Match -> VLAN VID field**

As per flow match specifications (section A.2.3.7 in [4]), following match types are supported :-

- (a) Match packets with or without a tag

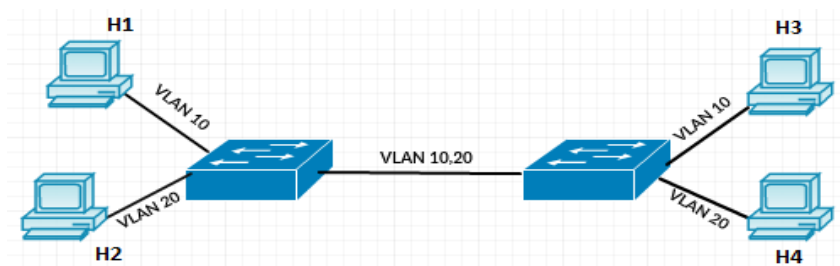


- 
- (b) Match only untagged packets
  - (c) Match only tagged packets
  - (d) Match tagged packets with specific value

For tagged packets (regardless of value), the TCI value specified is 0x1000. Similarly, for untagged packets, TCI value is 0x0000.

## 4 802.1Q IMPLEMENTATION IN OPENVSWITCH

Before writing the controller application in RYU for VLAN tagging, the implementation of VLAN Tagging in OpenvSwitch was first analyzed by manual VLAN configuration of OpenvSwitch.



**Figure 5:** Network Topology

A linear 2 switch -4 Host topology was considered (Fig 5). Two VLANs (VID 10, 20) are depicted in this network with one host member per switch. Two ports per switch need to be configured as an access port with the respective VLANs and the third port must be configured as a trunk port allowing the passage of both VLANs (10,20).

The OVS commands [7] for configuring a port as an access or trunk port are listed below: -

**1. Configuring a port as access port and tag incoming packet with VID = vlan**

`ovs-vsctl set port [port] tag = vlan`

**2. Configuring a port as trunk port , allowing the VIDs= vlan1,vlan2 to pass**

`ovs-vsctl set port [port] trunks = vlan1,vlan2`

Fig 6 shows the configuration commands for the network configuration in Fig 5. (Host H1 and H3 are on eth-1, H2 and H4 on eth-2)

```
mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
mininet> sh ovs-vsctl set port s1-eth1 tag=10
mininet> sh ovs-vsctl set port s1-eth2 tag=20
mininet> sh ovs-vsctl set port s2-eth1 tag=10
mininet> sh ovs-vsctl set port s2-eth2 tag=20
mininet> sh ovs-vsctl set port s1-eth3 trunks=10,20
mininet> sh ovs-vsctl set port s2-eth3 trunks=10,20
mininet> █
```

**Figure 6:** Configuring the ports as access and trunk ports

Further, a flow entry must be added to the switch corresponding to conventional L2 switching (action= "NORMAL") so that necessary MAC addresses are learned and packets are forwarded out the correct port.

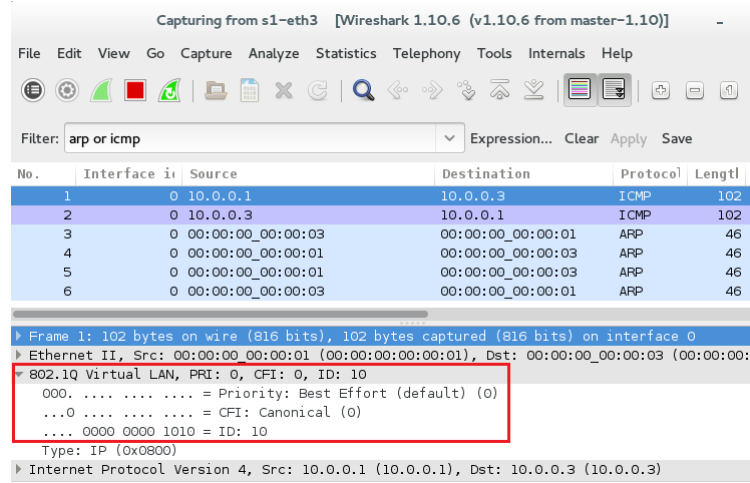
Once the above flow entry is added to both the switches (Fig 7), we can verify the ping reachability between the hosts. Based on the above configuration, it is expected that only members of the same VLAN will be able to ping each other.

Fig 7. shows the ping results which conform to the expected results.

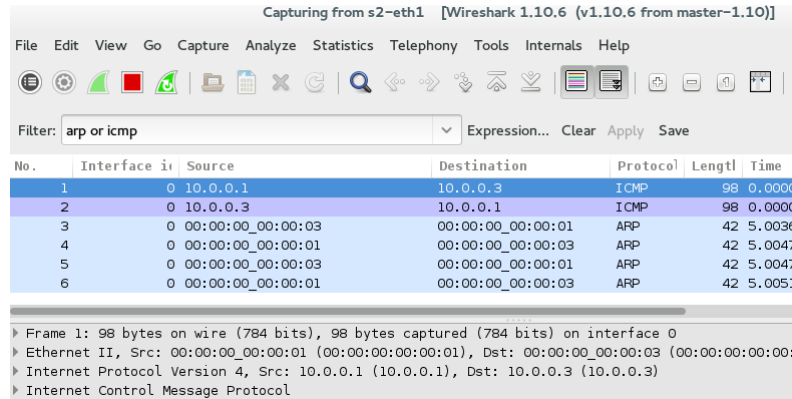
```
mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
mininet> sh ovs-ofctl add-flow --protocols=OpenFlow13 s1 action=normal
mininet> sh ovs-ofctl add-flow --protocols=OpenFlow13 s2 action=normal
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X
h2 -> X X h4
h3 -> h1 X X
h4 -> X h2 X
*** Results: 66% dropped (4/12 received)
mininet> █
```

**Figure 7:** Ping Test showing reachability of hosts. Only members of the same VLAN are able to ping each other

The tagging and untagging operation was verified using Wireshark, by taking a packet capture at port Eth -3 of switch S1 and port Eth -1 of switch S2 (for packet flow from H1 to H3).



**Figure 8:** Wireshark Capture at S1-Eth3 showing VLAN tagged packet (from H1) with VID=10



**Figure 9:** Wireshark Capture at S2-Eth1 showing that the packet from H1 is untagged (no 802.1Q header) before sending out through port 1

## 5 802.1Q IMPLEMENTATION IN RYU

The objective now is to develop a RYU controller application that will automatically add required flow entries for VLAN ID matching, tagging and untagging on a given topology consisting of OpenvSwitch (OVS) nodes. For simplicity, topology discovery will not be done and therefore the topology is assumed to be known. Also, the topologies considered are loop-free and no explicit spanning tree protocols will be run. The required VLAN configuration based on this known topology will be stored in the application. As stated before (Section 2), port based classification will be used.

---

In this implementation, we will consider 3 types of networks: -

1. **VLAN-only Network with a single trunk line per switch:** The network consists of nodes who are all members of one of the configured VLANs. So only tagged packets will arrive or depart from the trunk port of the switch.
2. **VLAN-only Network with multiple trunk lines per switch:** The network consists of nodes who are all members of one of the configured VLANs. Similar to a), only tagged packets will arrive or depart from the trunk port of the switch. But unlike in a), not all arriving packets at the trunk ports would be untagged. Some packets would need to be relayed in its current form to the next hop.
3. **Hybrid Networks with VLAN aware and Non-VLAN aware hosts and/or switches:** Not all hosts/switches are members of a particular VLAN/aware of VLAN configuration in the network. So a VLAN aware switch would need to route/relay untagged packets also.

Although three different networks have been considered here (progressively from basic to advanced), the final application will be generalized such that it is applicable for all 3 types of topologies.

In general, the flow actions to be added to the switch include the following: -

### **VLAN Tagging**

1. Match packet based on "in-port" and "tag-status" (whether tagged or untagged). If untagged, tag it with appropriate VID based on the given configuration and output through trunk port of the switch (Note: If destination is connected to the same switch and has the same VLAN membership, then no tagging needs to be done and the packet must be output through the appropriate access port)
2. If packet is already tagged, then if destination is not on the current switch, it must be "relayed" out to the next hop switch through the trunk port, keeping the packet tagged.

### **VLAN Matching and Untagging**

1. If the packet is tagged, Match packet VID and Destination address. Untag the packet before sending it through output port (based on MAC address table)

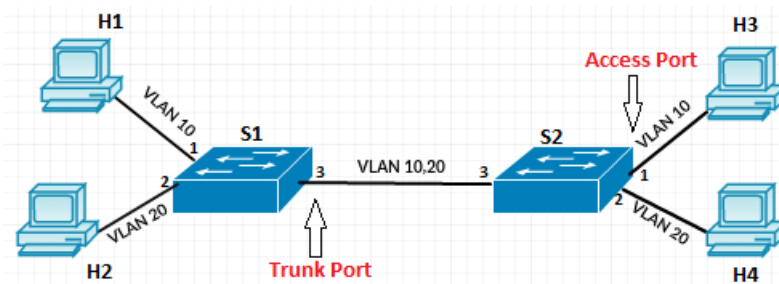
---

For the above mentioned flow actions, specialized classes are available in the RYU Open-Flow 1.3 protocol API [8] which are listed below: -

1. **OFFActionPushVlan**(*ethertype=33024, type=None, len=None*): Pushes a new VLAN tag into the packet with default VID = 0
2. **OFFActionPopVlan**(*type=None, len=None*): Removes the outermost VLAN tag
3. **OFFMatch**(*vlan\_vid=0x1000 | "vid"*): This matches a tagged packet containing VID = "vid" (integer value). The value 0x1000 comes from the OpenFlow 1.3 specifications mentioned in Section 3.
4. **OFFActionSetField**(*vlan\_vid="vid"*): Sets the VID value in the TCI field as "vid" (Integer value)

## 5.1 VLAN-Only Network with Single Trunk Line per Switch

We will consider the same basic topology seen in Fig 5.

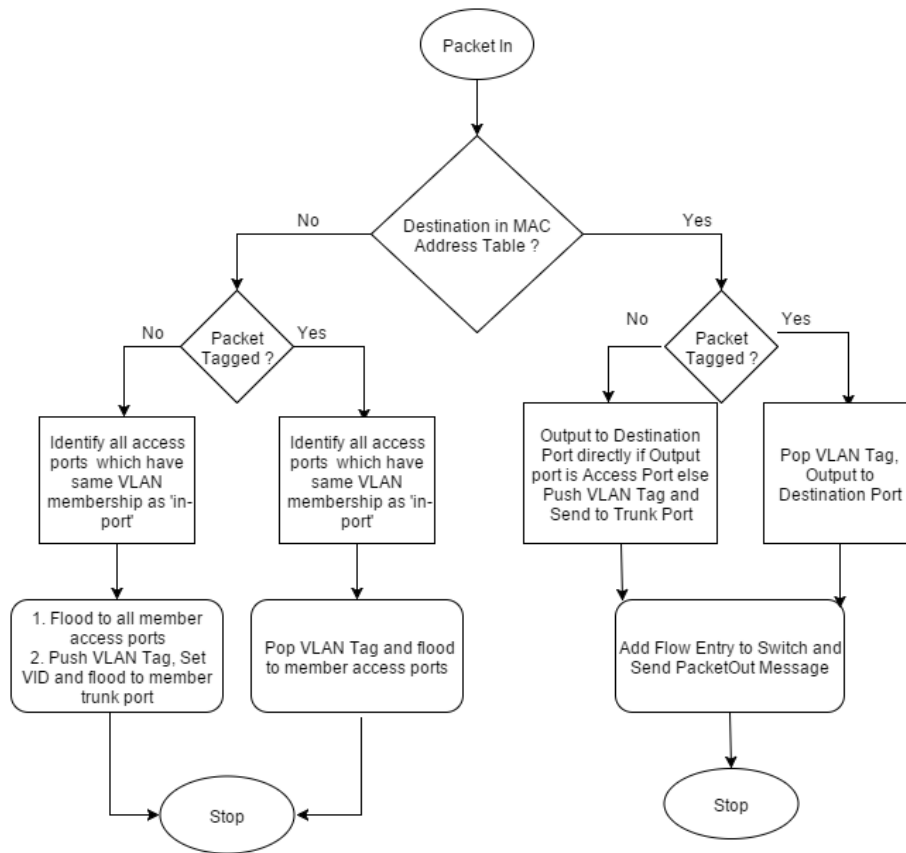


**Figure 10:** Network Topology (Single Trunk Line per Switch)

Fig 11 shows the decision flow of the controller. Broadcast flow entries have not been added in this flow for purpose of generality (for avoiding broadcast loops in non-linear topologies).

Since there is only 1 trunk line per switch, any arriving packet at a trunk port will be in tagged state and is therefore untagged before sending it through the output port.

The topology can be extended by adding more number of hosts per switch. The details of the flow entries added are discussed in the results section.



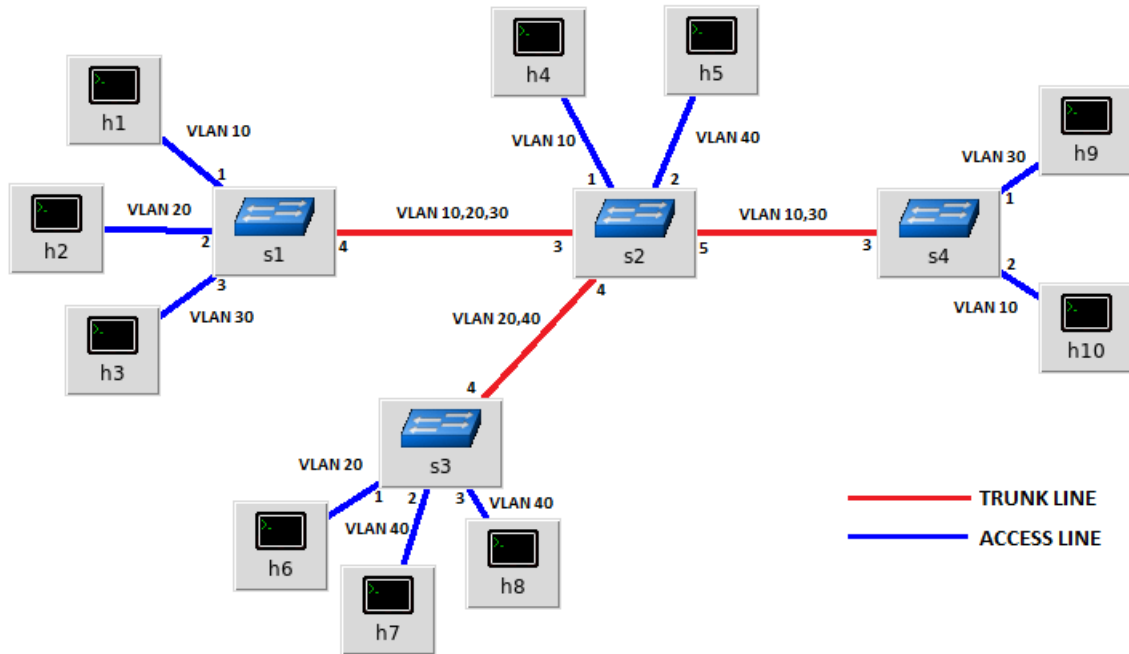
**Figure 11:** Flowchart Showing the Decision Flow of the Controller

## 5.2 VLAN-Only Network with Multiple Trunk Lines per Switch

The topology considered for this case is shown in Fig 12.

The modified decision flow that would work with such networks is shown in Fig 13. In this case, when an "untagged" broadcast packet is received, then it must be flooded to all access ports and trunk ports which are members of the same VLAN. Before sending to the trunk port, it must be tagged with appropriate VID. For example, when S3 receives a broadcast packet from H8 (which is untagged), it is flooded to H7 as is and sent to the trunk port (towards S2) after tagging.

Similarly, when a "tagged" broadcast packet is received, before flooding to an access port, the tag must be popped. So, when the broadcast packet is received by S2 from host H8, the outer tag is popped and sent out to H5. (which has VLAN membership of 40)



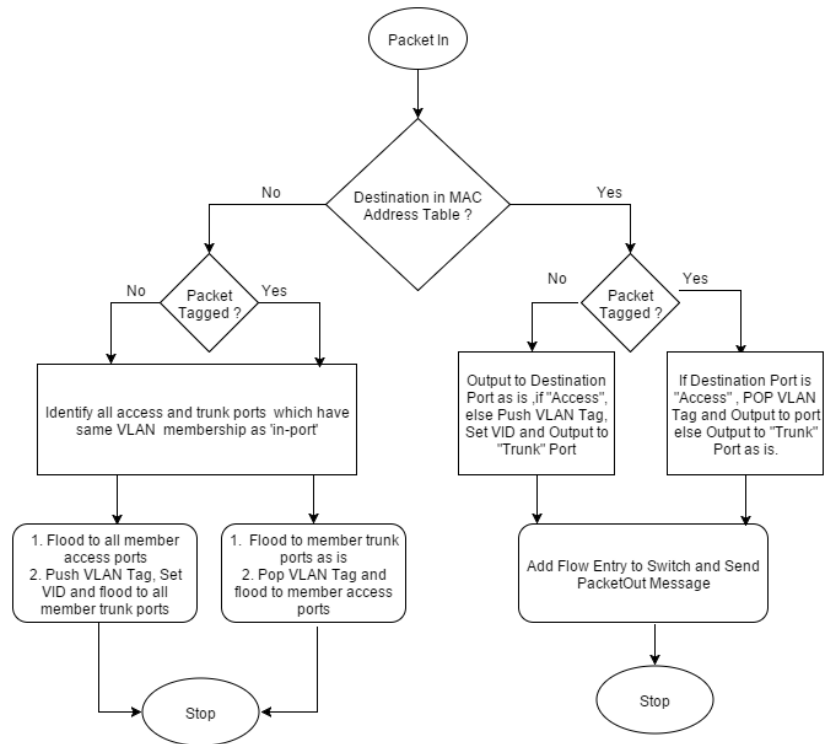
**Figure 12:** VLAN-Only Network with Multiple Trunk Lines

### 5.3 Hybrid Network with VLAN and Non-VLAN aware ("Native" VLAN) Nodes

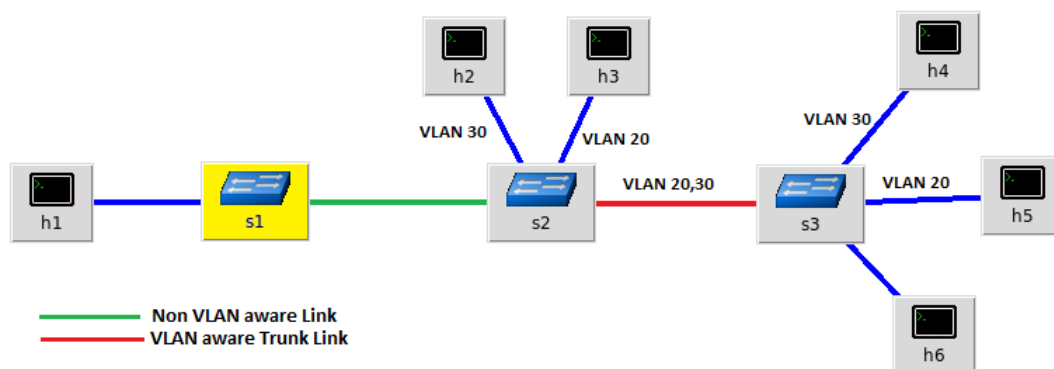
The topology considered for this case is shown in Fig 14. The "yellow" highlighted node is non-VLAN aware and therefore will not generate any "tagged" traffic. So Host H1 and H6 do not belong to any VLAN (can be considered as "native" VLAN). But their traffic will need to be relayed through VLAN aware switches before reaching the destination node. By implementing this topology, we can simulate a scenario where, in a large network, a small sub-network is to be designed and partitioned using VLANs. If the application is able to function in such a scenario flawlessly, it would indicate good flexibility.

The modified decision flow which is compatible with above type of networks is shown in Fig 15. Note that this decision flow can be regarded as the final decision flow for the entire application as it is general and applicable to all 3 networks discussed till now.

For this flow to execute, the VLAN configuration in the application must also include the native VLAN configuration of a switch if it contains VLAN configured ports (Such as S2, S3). But it need not include configuration information of Non-VLAN aware nodes (such as S1 in the above topology). Also topology information is needed only for the VLAN aware subset of the network (S2 -S3 network).

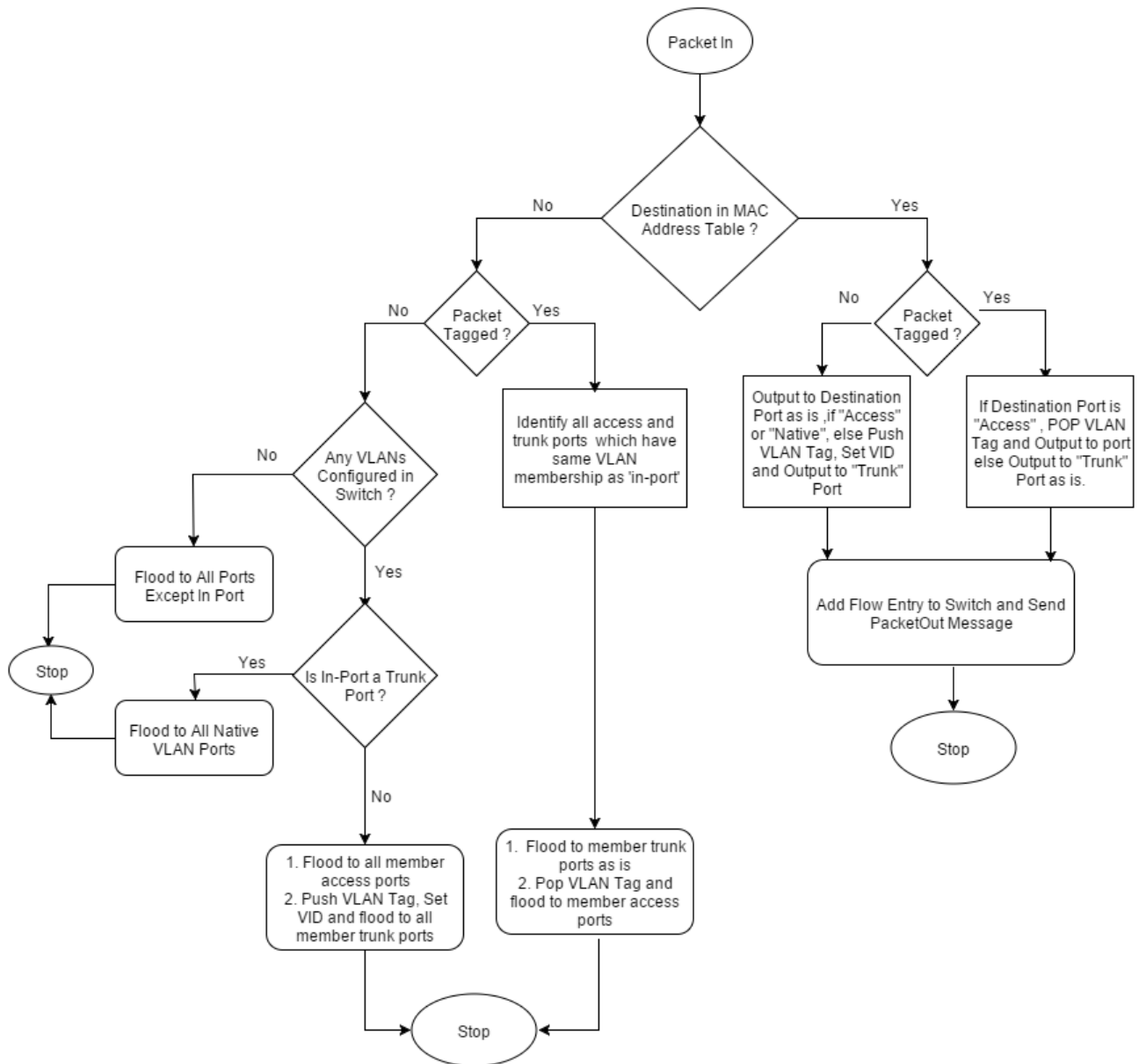


**Figure 13:** Decision Flow for Multiple Trunk Line Network



**Figure 14:** Network Topology for Hybrid Type Network





**Figure 15:** Decision Flow for Hybrid Network Containing Native VLAN as well as VLAN ports

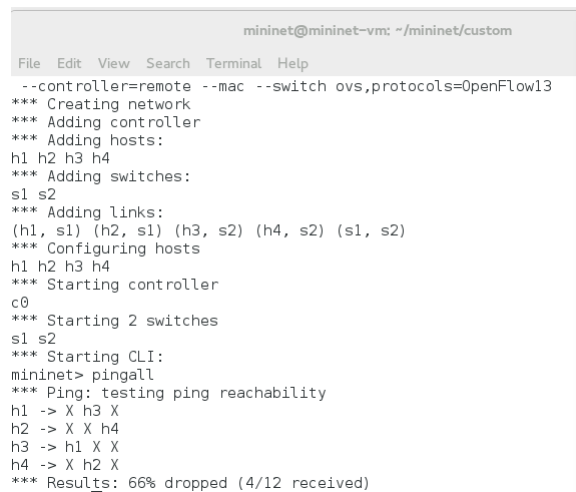
---

## 6 RESULTS

### 6.1 VLAN-Only Network with Single Trunk Line per Switch

The RYU application was executed on a mininet session consisting of the basic network topology given in Fig 10.

#### 6.1.1 Ping Reachability Test



```
mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
--controller=remote --mac --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X
h2 -> X X h4
h3 -> h1 X X
h4 -> X h2 X
*** Results: 66% dropped (4/12 received)
```

**Figure 16:** Ping Test (H1 and H3 are on VLAN 10, H2 and H4 are on VLAN 20)

The ping results show that only hosts which are members of the same VLAN are able to reach each other, which is expected

#### 6.1.2 Flow Entries in Switch

By executing the "dump-flows" command on each switch, it is verified whether or not the expected flow entries have been added.

Fig 17 shows the output of "dump-flows" on S1.

We can see that in each switch, there are two Push VLAN Tag actions (One for each Host) and corresponding two Pop VLAN Tag actions. The tag ids are set as per the desired configuration. In each of the Pop VLAN actions, the incoming VLAN ID is matched and then output through the required port. This ensures that only member hosts can reach each other.

Each of the flows has had a "match" or a "hit" since the number of packets are non-zero.

```

mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
mininet> sh ovs-ofctl --protocols=OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
  cookie=0x0, duration=579.388s, table=0, n_packets=2, n_bytes=140, priority=1,in
  _port=2,d_l_dst=00:00:00:00:00:04 actions=push_vlan:0x8100,set_field:20->vlan_vid
  ,output:3
  cookie=0x0, duration=588.443s, table=0, n_packets=2, n_bytes=140, priority=1,in
  _port=1,d_l_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:10->vlan_vid
  ,output:3
  cookie=0x0, duration=579.39s, table=0, n_packets=3, n_bytes=250, priority=1,in
  _port=3,d_l_vlan=20,d_l_dst=00:00:00:00:00:02 actions=strip_vlan,output:2
  cookie=0x0, duration=588.458s, table=0, n_packets=3, n_bytes=250, priority=1,in
  _port=3,d_l_vlan=10,d_l_dst=00:00:00:00:00:01 actions=strip_vlan,output:1
  cookie=0x0, duration=595.106s, table=0, n_packets=33, n_bytes=1554, priority=0
  actions=CONTROLLER:65535
mininet>

```

**Figure 17: Flow Entries on Switch S1**

```

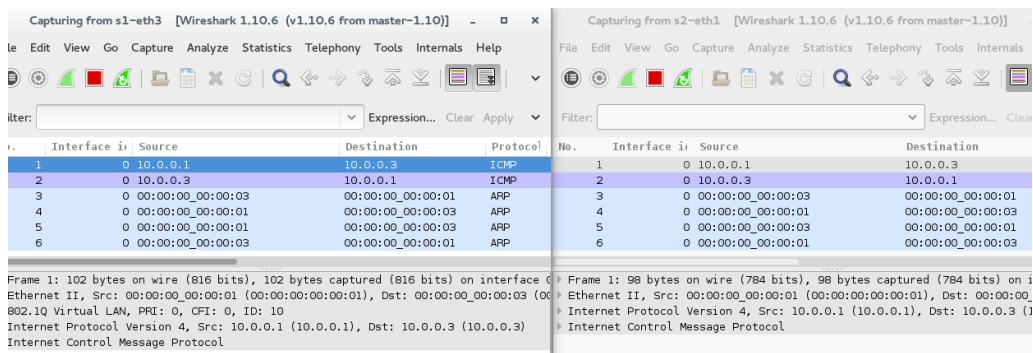
mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
mininet> sh ovs-ofctl --protocols=OpenFlow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
  cookie=0x0, duration=1155.982s, table=0, n_packets=2, n_bytes=148, priority=1,i
  n_port=3,d_l_vlan=20,d_l_dst=00:00:00:00:00:04 actions=strip_vlan,output:2
  cookie=0x0, duration=1165.036s, table=0, n_packets=2, n_bytes=148, priority=1,i
  n_port=3,d_l_vlan=10,d_l_dst=00:00:00:00:00:03 actions=strip_vlan,output:1
  cookie=0x0, duration=1155.988s, table=0, n_packets=3, n_bytes=238, priority=1,i
  n_port=2,d_l_dst=00:00:00:00:00:02 actions=push_vlan:0x8100,set_field:20->vlan_vi
  d,output:3
  cookie=0x0, duration=1165.059s, table=0, n_packets=3, n_bytes=238, priority=1,i
  n_port=1,d_l_dst=00:00:00:00:00:01 actions=push_vlan:0x8100,set_field:10->vlan_vi
  d,output:3
  cookie=0x0, duration=1171.645s, table=0, n_packets=33, n_bytes=1562, priority=0
  actions=CONTROLLER:65535
mininet>

```

**Figure 18: Flow Entries on Switch S2**

### 6.1.3 Wireshark Packet Capture

Finally, the packets at the switch are captured using Wireshark to verify the flow entry actions. The outgoing packet at S1-Eth3 contains the 802.1Q tag whereas it has been popped at the egress of S2-Eth1, which is the expected output.



**Figure 19: Wireshark Capture at S1-Eth3 and S2-Eth1 (For H1 Ping H3)**

---

## 6.2 VLAN-Only Network with Multiple Trunk Lines per Switch

The results for the multi-trunk line topology (Fig 12.) are presented below

### 6.2.1 Ping Reachability Test

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X h4 X X X X h10
h2 -> X X X h6 X X X
h3 -> X X X X X X h9 X
h4 -> h1 X X X X X h10
h5 -> X X X X h7 h8 X X
h6 -> X h2 X X X X X X
h7 -> X X X h5 X h8 X X
h8 -> X X X h5 X h7 X X
h9 -> X X h3 X X X X X
h10 -> h1 X X h4 X X X X
*** Results: 82% dropped (16/90 received)
mininet>
```

**Figure 20:** Ping Reachability Test

Based on the network connectivity and VLAN configuration of Fig 12., the ping test shows the expected results.

### 6.2.2 Flow Entries in Switch

We shall verify the flow entries on S1 and S2 by executing the "dump-flows" command. Fig 21 and 22 show the respective outputs.



```
mininet@mininet-vm: ~/mininet/custom
File Edit View Search Terminal Help
mininet> sh ovs-ofctl --protocols=OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
  cookie=0x0, duration=1652.004s, table=0, n_packets=0, n_bytes=0, priority=1,in_
  port=1,d_l_dst=00:00:00:00:00:0a actions=push_vlan:0x8100,set_field:10->vlan_vid,
  output:4
  cookie=0x0, duration=1852.855s, table=0, n_packets=2, n_bytes=140, priority=1,i
  n_port=1,d_l_dst=00:00:00:00:00:04 actions=push_vlan:0x8100,set_field:10->vlan_vi
  d,output:4
  cookie=0x0, duration=1825.674s, table=0, n_packets=2, n_bytes=140, priority=1,i
  n_port=2,d_l_dst=00:00:00:00:00:06 actions=push_vlan:0x8100,set_field:20->vlan_vi
  d,output:4
  cookie=0x0, duration=1792.544s, table=0, n_packets=2, n_bytes=140, priority=1,i
  n_port=3,d_l_dst=00:00:00:00:00:09 actions=push_vlan:0x8100,set_field:30->vlan_vi
  d,output:4
  cookie=0x0, duration=1792.555s, table=0, n_packets=3, n_bytes=250, priority=1,i
  n_port=4,d_l_vlan=30,d_l_dst=00:00:00:00:00:03 actions=strip_vlan,output:3
  cookie=0x0, duration=1825.682s, table=0, n_packets=3, n_bytes=250, priority=1,i
  n_port=4,d_l_vlan=20,d_l_dst=00:00:00:00:00:02 actions=strip_vlan,output:2
  cookie=0x0, duration=1852.864s, table=0, n_packets=8, n_bytes=588, priority=1,i
  n_port=4,d_l_vlan=10,d_l_dst=00:00:00:00:00:01 actions=strip_vlan,output:1
  cookie=0x0, duration=1898.822s, table=0, n_packets=180, n_bytes=8216, priority=
  0 actions=CONTROLLER:65535
mininet>
```

**Figure 21:** Flow Entries on Switch S1



---

## 6.3 Hybrid Network with VLAN and Non-VLAN aware ("Native" VLAN) Nodes

The results for the hybrid topology (Fig 14.) are presented below. In this hybrid case, we want the switch to behave like a VLAN switch towards a "tagged" packet and like a normal learning switch (like the Simple Switch) for an "un-tagged" packet.

### 6.3.1 Ping Reachability Test

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X h6
h2 -> X X h4 X X
h3 -> X X X h5 X
h4 -> X h2 X X X
h5 -> X X h3 X X
h6 -> h1 X X X X
*** Results: 80% dropped (6/30 received)
mininet>
```

**Figure 24:** Ping Reachability Test

Based on the network connectivity and VLAN configuration of Fig 14., the ping test shows the expected results. We can note here that host H6 and H1 belong to the "native" VLAN and are able to ping each other.

### 6.3.2 Flow Entries in Switch

We shall verify the flow entries on S2 and S3 by executing the "dump-flows" command (Fig 24)

```
File Edit View Search Terminal Help
mininet> sh ovs-ofctl --protocols=openFlow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=421.91s, table=0, n_packets=7, n_bytes=546, priority=1,in_port=4,d_l_
lan=20,d_l_dst=00:00:00:00:00:03 actions=strip_vlan,output:2
 cookie=0x0, duration=436.974s, table=0, n_packets=7, n_bytes=546, priority=1,in_port=4,d_l_
vlan=30,d_l_dst=00:00:00:00:00:02 actions=strip_vlan,output:1
 cookie=0x0, duration=421.901s, table=0, n_packets=6, n_bytes=420, priority=1,in_port=2,d_l_
dst=00:00:00:00:00:05 actions=push_vlan:0x8100,set_field:20->vlan_vid,output:4
 cookie=0x0, duration=443.019s, table=0, n_packets=9, n_bytes=602, priority=1,in_port=4,d_l_
dst=00:00:00:00:00:01 actions=output:3
 cookie=0x0, duration=436.972s, table=0, n_packets=6, n_bytes=420, priority=1,in_port=1,d_l_
dst=00:00:00:00:00:04 actions=push_vlan:0x8100,set_field:30->vlan_vid,output:4
 cookie=0x0, duration=443.008s, table=0, n_packets=8, n_bytes=504, priority=1,in_port=3,d_l_
dst=00:00:00:00:00:06 actions=output:4
 cookie=0x0, duration=452.287s, table=0, n_packets=165, n_bytes=7298, priority=0 actions=CO
NTROLLER:65535
mininet>
```

**Figure 25:** Flow Entries on Switch S2

In both the switches, flow entries covering VLAN action and conventional L2 forwarding (for "native" VLAN packets) have been inserted and matched. This way any packet regardless

```

mininet> sh ovs-ofctl --protocols=OpenFlow13 dump-flows s3
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=205.528s, table=0, n_packets=6, n_bytes=444, priority=1,in
 _port=4,dl_vlan=20,dl_dst=00:00:00:00:00:05 actions=strip_vlan,output:2
 cookie=0x0, duration=220.574s, table=0, n_packets=6, n_bytes=444, priority=1,in
 _port=4,dl_vlan=30,dl_dst=00:00:00:00:00:04 actions=strip_vlan,output:1
 cookie=0x0, duration=226.6s, table=0, n_packets=8, n_bytes=504, priority=1,in_p
 ort=4,dl_dst=00:00:00:00:00:06 actions=output:3
 cookie=0x0, duration=226.622s, table=0, n_packets=9, n_bytes=602, priority=1,in
 _port=3,dl_dst=00:00:00:00:00:01 actions=output:4
 cookie=0x0, duration=205.535s, table=0, n_packets=7, n_bytes=518, priority=1,in
 _port=2,dl_dst=00:00:00:00:00:03 actions=push_vlan:0x8100,set_field:20->vlan_vid
 ,output:4
 cookie=0x0, duration=220.588s, table=0, n_packets=7, n_bytes=518, priority=1,in
 _port=1,dl_dst=00:00:00:00:00:02 actions=push_vlan:0x8100,set_field:30->vlan_vid
 ,output:4
 cookie=0x0, duration=241.868s, table=0, n_packets=160, n_bytes=7096, priority=0
 actions=CONTROLLER:65535
mininet> █

```

**Figure 26:** Flow Entries on Switch S3

of whether it is tagged can be forwarded by the switch, thus allowing VLAN and Non-VLAN enabled networks to co-exist.

## 7 SUMMARY AND CONCLUSIONS

In this project, 802.1Q VLAN tagging has been successfully implemented through SDN using the RYU framework on OpenFlow. The RYU application was executed for 3 different types of network topologies: -

1. VLAN-only Network with a single trunk line per switch.
2. VLAN-only Network with multiple trunk lines per switch.
3. Hybrid Networks with VLAN aware and Non-VLAN aware switches.

Although a relatively small network was considered in the above scenarios for testing, the decision flow of the application has been designed in such a way that it is scalable for any number of nodes and any number of VLANs. The successful implementation in Hybrid Networks also shows that the application can be seamlessly introduced in an already developed Non-VLAN aware network.

This project also helped to understand how SDN works in practice i.e. how the control and data planes are decoupled. The use of SDN for implementing VLAN functionality brings the following benefits: -

1. The VLANs are centrally configured in the controller (provided the topology is known). So the trouble of logging in to each device and setting the configuration is avoided.
2. SDN allows easier and faster enhancements or modifications to the existing configuration. This is again due to centralized control.

- 
3. Debugging of control plane issues is more convenient. In conventional distributed scenario, it is generally difficult to pin down or isolate the problem source (which could be a particular node running a faulty software).

Despite the above advantages gained through SDN, the application has certain **limitations** which need to be addressed: -

1. **Currently, the application works only with a loop-free technology as there is no inclusion of Spanning Tree Protocol (STP).** Once a loop is introduced in the network, one or more Spanning Tree Protocol Instances (Multiple Spanning Tree Protocol), possibly corresponding to each VLAN must run, so as to create a logical loop-free topology.
2. **Topology discovery functionality is not included.** This is essential in case of MAC-based VLAN classification wherein a host may change location (port). The new location can then be dynamically discovered using topology updates.



---

## REFERENCES

- [1] R. Jain, "Virtual Local Area Networks," 14 August 1997. [Online]. Available: [http://www.cse.wustl.edu/~jain/cis788-97/ftp/virtual\\_lans/](http://www.cse.wustl.edu/~jain/cis788-97/ftp/virtual_lans/)
- [2] IEEE Computer Society *IEEE Standard for Local and Metropolitan Area Networks (802.1Q)*, IEEE, New York, 2014.
- [3] Open Networking Foundation, "Software Defined Networking : The New Norm for Networks," 12 April 2012. [Online]. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers>
- [4] Open Networking Foundation, "OpenFlow Switch Specification (Version 1.3)," 25 June 2012.[Online]. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specific>
- [5] Nippon Telegraph and Telephone Corporation, "Getting Started -RYU 3.2 Documentation," 2014. [Online]. Available: [http://ryu-zhdoc.readthedocs.org/en/latest/getting\\_started.html](http://ryu-zhdoc.readthedocs.org/en/latest/getting_started.html).
- [6] M. Kobayash, S. Seetharaman and G. Parulkar *"Maturing of OpenFlow and Software-defined Networking through Deployments,"* Computer Networks, no. <http://dx.doi.org/10.1016/j.bjp.2013.10.011>, 2013.
- [7] OpenvSwitch.org, "OpenvSwitch Manual," 2014. [Online]. Available: <http://openvswitch.org/support/dist-docs/ovs-vsctl.8.txt>
- [8] Nippon Telegraph and Telephone Corporation, "OpenFlow v 1.3 Messages and Structures-RYU 3.29 Documentation," 2014. [Online]. Available: [http://ryu-zhdoc.readthedocs.org/en/latest/ofproto\\_v1\\_3\\_ref.html](http://ryu-zhdoc.readthedocs.org/en/latest/ofproto_v1_3_ref.html)