INR Lab Assignment: Spanning Tree Protocol and 802.1Q *

Preparation

VM Name	IP Address	Netmask	Broadcast
Guest-03	145.100.108.84	255.255.255.240	145.100.108.95

Task 1. Create a new VM running the Ubuntu14.04 Trusty operating system. Give it 1GB of RAM and a public ip. Make sure you can log in using your ssh keys.

Answer:

Create the VM:

```
kotaiba@bristol:/etc/xen$ sudo xen-create-image --hostname=Guest-03 --
dist=trusty --memory=1024MB --size=10GB --swap=1024MB --vcpus=2 --
ip=145.100.108.84  --netmask=255.255.255.240  --gateway=145.100.108.81 --
lvm=VolumeGroupXen

General Information
--------------------
Hostname       :  Guest-03
Distribution   :  trusty
Mirror         :  http://archive.ubuntu.com/ubuntu
Partitions     :  swap            1024MB (swap)
                  /               10GB   (ext3)
Image type     :  full
Memory size    :  1024MB
Bootloader     :  pygrub

Networking Information
----------------------
IP Address 1   : 145.100.108.84 [MAC: 00:16:3E:3E:07:55]
Netmask        : 255.255.255.240
Gateway        : 145.100.108.81


Creating swap on /dev/VolumeGroupXen/Guest-03-swap
Done

Creating ext3 filesystem on /dev/VolumeGroupXen/Guest-03-disk
Done
Installation method: debootstrap
Done
```

```
Running hooks
Done

No role scripts were specified.  Skipping

Creating Xen configuration file
Done

No role scripts were specified.  Skipping
Setting up root password
Generating a password for the new guest.
All done


Logfile produced at:
     /var/log/xen-tools/Guest-03.log

Installation Summary
---------------------
Hostname        :  Guest-03
Distribution    :  trusty
MAC Address     :  00:16:3E:3E:07:55
IP Address(es)  :  145.100.108.84
SSH Fingerprint :  XXX
SSH Fingerprint :  XXX
SSH Fingerprint :  XXX
SSH Fingerprint :  XXX
Root Password   :  XXX
```

```
kotaiba@bristol:/etc/xen$ sudo xl create Guest-03.cfg
Parsing config from Guest-03.cfg

kotaiba@bristol:/etc/xen$ sudo xl console Guest-03



Guest-03 login: root
Password:
Last login: Fri Nov 10 13:53:23 CET 2017 on hvc0
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
root@Guest-03:~#
```

Now, I create a user for the ssh:

```
root@Guest-03:~# adduser kotaiba
Adding user `kotaiba' ...
Adding new group `kotaiba' (1000) ...
Adding new user `kotaiba' (1000) with group `kotaiba' ...
Creating home directory `/home/kotaiba' ...
Copying files from `/etc/skel' ...
```

```
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for kotaiba
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
root@Guest-03:~# usermod -aG sudo kotaiba
```

After, I configured SSH, now let's add the key and test it:

```
kotaiba@bristol:~/.ssh$ ssh-copy-id -i myKey kotaiba@145.100.108.84
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "myKey.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
kotaiba@145.100.108.84's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'kotaiba@145.100.108.84'"
and check to make sure that only the key(s) you wanted were added.
```

Test:

```
kotaiba@bristol:~/.ssh$ ssh kotaiba@145.100.108.84 -i myKey
Enter passphrase for key 'myKey':
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Nov 10 14:05:41 2017 from 145.100.108.81
kotaiba@Guest-03:~$
```

Task 2. Install the following packages: python git user-mode linux uml-utilities debootstrap vde2 screen. Afterwards, clone the old git repository: https://github.com/TeamOS3/ogopogo.git.

Answer:

Install the packages:

```
 kotaiba@Guest-03:~$ sudo apt-get install python git user-mode-linux uml-
```

```
utilities debootstrap vde2 screen
```

Clone the git repository:

```
kotaiba@Guest-03:~$ git clone https://github.com/TeamOS3/ogopogo.git
Cloning into 'ogopogo'...
remote: Counting objects: 756, done.
remote: Total 756 (delta 0), reused 0 (delta 0), pack-reused 756
Receiving objects: 100% (756/756), 113.59 KiB | 0 bytes/s, done.
Resolving deltas: 100% (360/360), done.
Checking connectivity... done.
```

Task 3. Set the values of A,B,X,Y in the rc.local script to a mod 10, b mod 10, x and y, respectively, as derived in the previous lab. These variables are used to generate the MAC addresses in this lab. Run the build_rootfs.sh, which creates a disk image, and copy this to the /tmp directory.

Answer:

A, B, X, Y values:

```
FName: Kotaiba
LName: Alachkar

A = 7 mod 10 = 7
B = 8 mod 10 = 8
X = 2
Y = 14 ( E in hex )
```

Set the values in /ogopogo/scripts/rc.local file:

```
#These should all be single digit decimal numbers
A=7
B=8

#These should all be single digit hexadecimal numbers
X="2"
Y="E"
```

Now, run /ogopogo/scripts/build_rootfs.sh, then copy disk.img to /tmp

```
root@Guest-03:/home/kotaiba/ogopogo/scripts# ./build_rootfs.sh

root@Guest-03:/home/kotaiba/ogopogo/scripts# ls
build_rootfs.sh  disk.img  rc.local
```

```
root@Guest-03:/home/kotaiba/ogopogo/scripts# cp disk.img /tmp
```

Task 4. Create the config file that starts the network using the bridge.cfg example. Keep the sniffer there. Run python ogo.py start bridge.cfg to start the environment. The sniffer should leave pcap files in its home directory, which is the /tmp directory by default .

Answer:

Configure bridge.cfg file that meet Figure 1: STP Network tobology:

```
root@Guest-03:/home/kotaiba/ogopogo/examples# cat bridge.cfg
############################################################################
####
#GLOBAL CONFIG
############################################################################
####
# config that starts up an uml instance acting as a bridge
#   host1 ---- hub0 ---- bridge1 ----- hub1 ----- host2
#               \                     /
#                \------sniffer1-----/
[global]

#this defines where to store the control files and pid files for vde_switch
session_path = /tmp

#memory per uml instance
mem = 36M

#base root image
root_image= /tmp/disk.img

#number of hubs in topology.
hubs = 2

############################################################################
####
#HOST CONFIG
############################################################################
####


[H1]
role= host
home=/tmp
eth0 = 0,10.0.0.1/8,2001:0db8:0:f101::1/64
```

```
[B1]
role = bridge
home=/tmp
eth0 = 0,,
eth1 = 1,,
# The following line defines that a bridge 'bridge0' should be created in
the i$
# which contains eth0 and eth1. Note the required use of single quotes
followed$
pass_bridge0='"eth0 eth1"'

[B2]
role = bridge
home=/tmp
eth0 = 0,,
eth1 = 1,,
# The following line defines that a bridge 'bridge0' should be created in
the i$
# which contains eth0 and eth1. Note the required use of single quotes
followed$
pass_bridge0='"eth0 eth1"'

[B3]
role = bridge
home=/tmp
eth0 = 0,,
eth1 = 1,,
# The following line defines that a bridge 'bridge0' should be created in
the i$
# which contains eth0 and eth1. Note the required use of single quotes
followed$
pass_bridge0='"eth0 eth1"'

[H2]
role = host
home=/tmp
eth0 = 1,10.0.0.2/8,2001:0db8:0:f101::2/64



[sniffer1]
role = sniffer
home=/tmp
eth0 = 0,,
eth1 = 1,,
```

Now, let's start the environment:

```
root@Guest-03:/home/kotaiba/ogopogo# python ogo.py start bridge.cfg

DEBUG: vde_switch --daemon --hub --mgmt /tmp/switch-0.mgmt --sock
/tmp/switch-0.ctl
```

```
DEBUG: vde_switch --daemon --hub --mgmt /tmp/switch-1.mgmt --sock
/tmp/switch-1.ctl
INFO: Starting sniffer sniffer1
DEBUG: screen -dmS sniffer1 linux.uml umid=sniffer1 role=sniffer index=00
name=sniffer1 ubd0=/tmp/disk.img-sniffer1.cow,/tmp/disk.img
eth0=daemon,,unix,/tmp/switch-0.ctl/ctl
eth1=daemon,,unix,/tmp/switch-1.ctl/ctl mem=36M interface_count=2 home=/tmp
noswap tapdev0=
INFO: sleeping 5 seconds to allow sniffers to start first
INFO: Starting host B1
DEBUG: screen -dmS B1 linux.uml umid=B1 role=bridge index=00 name=B1
ubd0=/tmp/disk.img-B1.cow,/tmp/disk.img
eth0=daemon,,unix,/tmp/switch-0.ctl/ctl
eth1=daemon,,unix,/tmp/switch-1.ctl/ctl mem=36M interface_count=2 home=/tmp
bridge0='"eth0 eth1"' noswap tapdev0=
INFO: Starting host B2
DEBUG: screen -dmS B2 linux.uml umid=B2 role=bridge index=01 name=B2
ubd0=/tmp/disk.img-B2.cow,/tmp/disk.img
eth0=daemon,,unix,/tmp/switch-0.ctl/ctl
eth1=daemon,,unix,/tmp/switch-1.ctl/ctl mem=36M interface_count=2 home=/tmp
bridge0='"eth0 eth1"' noswap tapdev0=
INFO: Starting host B3
DEBUG: screen -dmS B3 linux.uml umid=B3 role=bridge index=02 name=B3
ubd0=/tmp/disk.img-B3.cow,/tmp/disk.img
eth0=daemon,,unix,/tmp/switch-0.ctl/ctl
eth1=daemon,,unix,/tmp/switch-1.ctl/ctl mem=36M interface_count=2 home=/tmp
bridge0='"eth0 eth1"' noswap tapdev0=
INFO: Starting host H1
DEBUG: screen -dmS H1 linux.uml umid=H1 role=host index=00 name=H1
ubd0=/tmp/disk.img-H1.cow,/tmp/disk.img
eth0=daemon,,unix,/tmp/switch-0.ctl/ctl ip0=10.0.0.1/8
ip60=2001:0db8:0:f101::1/64 mem=36M interface_count=1 home=/tmp noswap
tapdev0=
INFO: Starting host H2
DEBUG: screen -dmS H2 linux.uml umid=H2 role=host index=01 name=H2
ubd0=/tmp/disk.img-H2.cow,/tmp/disk.img
eth0=daemon,,unix,/tmp/switch-1.ctl/ctl ip0=10.0.0.2/8
ip60=2001:0db8:0:f101::2/64 mem=36M interface_count=1 home=/tmp noswap
tapdev0=
DEBUG: screen -ls
There are screens on:
	18137.H2	(11/10/17 14:41:12)	(Detached)
	17786.H1	(11/10/17 14:41:10)	(Detached)
	17456.B3	(11/10/17 14:41:08)	(Detached)
	17113.B2	(11/10/17 14:41:06)	(Detached)
	16785.B1	(11/10/17 14:41:04)	(Detached)
	16740.sniffer1	(11/10/17 14:40:57)	(Detached)
6 Sockets in /var/run/screen/S-root.
```

Witnessing the startup

Task 5. Create a diagram of the networks showing the state of all the bridge ports and the root bridge. Hint: brctl showstp (not described in manual page).

Answer:

After using “brctl showstp bridge0” on each bridge, I summarize it in the following diagram:

Task 6. Describe in detail (i.e. your own words, not verbatim dumps) what packets are sent, when and why. Explain how the root bridge was elected. Only mention the relevant packets. Upload the raw dump file (as generated by the sniffer) to your wiki, and provide a link in your log.

Answer:

First, I will move the files to my desktop:

```
kalachkar@desktop-41:~$  scp kotaiba@145.100.108.84:/tmp/sniffer1-
eth0-1110-1341.pcap Desktop/pcaps/
The authenticity of host '145.100.108.84 (145.100.108.84)' can't be
established.
ECDSA key fingerprint is SHA256:tatx/UulmdnbMMqRMDrVy4Ljr8XS0NpVmNRHQs144Qk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '145.100.108.84' (ECDSA) to the list of known
hosts.
kotaiba@145.100.108.84's password:
sniffer1-eth0-1110-1341.pcap                                       100%
133KB  32.8MB/s   00:00
kalachkar@desktop-41:~$  scp kotaiba@145.100.108.84:/tmp/sniffer1-
eth1-1110-1341.pcap Desktop/pcaps/
kotaiba@145.100.108.84's password:
sniffer1-eth1-1110-1341.pcap                                       100%
134KB  45.1MB/s   00:00
```

Describe in details:

Following the packets sequence and see how BPDUs are generated from the Root Bridge and flow outward along the active Paths and move away from the Root Bridge.

Since, the bridges know nothing about each others. They start to send BPDU (Hello) in order to advertise themselves as root bridge (I'm the root bridge “ID”). After that all bridges know about each other (because the bridges are directly connected to each other) if it is not that means they weren't connected to each others (directly).

```
3   0.000005    00:b1:00:00:2e:00    Spanning-tree-(for-bridges)_00    STP
52    Conf. Root = 32768/0/00:b1:bb:00:78:11  Cost = 0  Port = 0x8001

8   2.444310    00:b2:00:01:2e:00    Spanning-tree-(for-bridges)_00    STP
52    Conf. Root = 32768/0/00:b2:bb:01:78:11  Cost = 0  Port = 0x8001

18  5.463577    00:b3:00:02:2e:00    Spanning-tree-(for-bridges)_00    STP
52    Conf. Root = 32768/0/00:b3:bb:02:78:11  Cost = 0  Port = 0x8001
```

As we see above is the cost 0, (the reason is that they connected to each other virtually not through physical link)

So what happened:

“A Switch with lowest Switch ID is selected as the Root Bridge (Root Switch) . When a Network Switch receives a configuration BPDU that has a lower Root Switch (Root Bridge) ID, compared with what the Network Switch has as lower Root Switch (Root Bridge) ID, the Network Switch will consider the Switch with lowest Root Switch (Root Bridge) ID as the Root Bridge (Root Switch) and start relaying the Configuration BPDUs which are received from the new Root Bridge (Root Switch) . After the Root Bridge (Root Switch) has been identified, all other Non-Root Switches bridges do not actually generate Configuration BPDUs. Non-Root Switch only propagates the BPDUs generated by the Root Bridge (Root Switch) . The Non-Root Switch also updates certain fields in the Configuration BPDUs, such as Message Age, Root Path Cost, Sender Bridge ID etc. When a port receives a BPDU, it has a path to the Root Bridge (Root Switch), because BPDUs are originated from the Root Bridge (Root Switch). The port which receives a BPDU is normally a Root Port. For a Non-Root Bridge a port that receives a BPDU, that port leads to the Root Bridge (Root Switch).”

Pcap files:

1- Pcap1

2- Pcap2

Source:

1-
http://www.omnisecu.com/cisco-certified-network-associate-ccna/how-bpdu-is-generated-and-how-bpdu-works.php

Task 7. a. What parameters are used in the BPDU packets

Answer:

- First, what is BPDU ? “BPDU stands for bridge protocol data unit. BPDUs are data messages that are exchanged across the switches within an extended LAN that uses a spanning tree protocol topology."

- BPDU packets contain information on ports, addresses, priorities and costs and ensure that the data ends up where it was intended to go. BPDU messages are exchanged across bridges to detect loops in a network topology. The loops are then removed by shutting down selected bridge interfaces and placing redundant switch ports in a backup, or blocked, state.

The BPDU parameters are as the following:

Source:

1- https://learningnetwork.cisco.com/thread/10067, 2- https://adelzalok.wordpress.com/2011/09/29/anatomy-of-a-mac-address-bpdu-bid-and-the-802-1q-ethernet-frame-tag/, 3- http://www.omnisecu.com/cisco-certified-network-associate-ccna/bridge-protocol-data-unit-bpdu-frame-format.php

Task 7. b. What is the role of each parameter?

Answer:

The format and roles of IEEE 802.1D Bridge Protocol Data Unit (BPDU) is:

- **Protocol ID (2 bytes):** Contains the value 0000 for IEEE 802.1D
- **Version ID (1 byte):** Contains the value zero.
- **BPDU Message Type (1 byte):** Configuration or TCN BPDU
- **Flags (1 byte):** The Topology Change (TC) bit signals a topology change. The Topology Change Acknowledgment (TCA) bit is set to acknowledge receipt of a configuration message.
  - 1 : Topology Change Flag
  - 2 : unused 0
  - 3 : unused 0
  - 4 : unused 0
  - 5 : unused 0
  - 6 : unused 0
  - 7 : unused 0
  - 8 : Topology Change Ack
- **Root Bridge (Root Switch) ID (8 bytes):** Identifies the root bridge by listing its 2-byte priority number followed by its 6-byte MAC address.
- **Root Path Cost (4 bytes) :** Contains the cost of the path from the bridge sending the configuration message to the Root Bridge (Root Switch) .
- **Sender Bridge (Switch) ID (8 bytes):** Identifies the Sender bridge by listing its 2-byte priority number followed by its 6-byte MAC address.
- **Port ID 2 bytes:** Identifies the port from which the configuration message was sent.
- **Message Age (2 bytes):** Specifies the amount of time elapsed since the Root Bridge (Root Switch) sent the configuration message on which the current configuration message is based.
- **Maximum Age (2 bytes):** Indicates when the current configuration message should be

deleted.
- **Hello time (2 bytes):** Provides the time period between Root Bridge (Root Switch) configuration messages.
- **Forward Delay (2 bytes):** Provides the length of time that bridges should wait before transitioning to a new state after a topology change.

Source:

1- http://www.omnisecu.com/cisco-certified-network-associate-ccna/bridge-protocol-data-unit-bpdu-frame-format.php

Task 7. c. What are they set to?

Answer:

I will choose a random STP packet and I will put it here (as a sample for value set):

Creating problems

Task 8. What happens if you shutdown the root bridge Describe all the events that take place from the moment the bridge goes down until the network has converged again.

Answer:

Before the max age time runs out all (non-root bridges) will have no idea that the root bridge is down. However, after that time all the bridges will detect that the root bridge is down which leads to topology change. In this case the reelection process will start. "First a switch which has a Best bridge ID (lower bridge ID) will be elected as a root bridge of a Network. Then, once root bridge is elected all other non-root bridges will elect one root port which is best path towards a root bridge and that can be done by using link cost of cable.STP have its own cost for a bandwidth of a link for example: 100mbps - 19 10mbps- 100 1gbps-4 10gbps-2. After that once root port is elected all swithces on a network elect one designated port (which have best cost to reach a root bridge on that subnet) per segment basis. The port which are not a root port and designated port will be in blocking state.which we call non designated port."

I will test this by shutting down the root bridge and then look at the pcap file:

```
root@B1:~# ip link set bridge0 down
bridge0: port 2(eth1) entered disabled state
bridge0: port 1(eth0) entered disabled state
```

Check what I said above:

```
00:b1:00:00:2e:00   Spanning-tree-(for-bridges)_00    STP    52    Conf.
Root = 32768/0/00:b1:bb:00:78:11  Cost = 0  Port = 0x8001
00:b1:00:00:2e:00   Spanning-tree-(for-bridges)_00    STP    52    Conf. TC
+ Root = 32768/0/00:b1:bb:00:78:11  Cost = 0  Port = 0x8001
00:b1:00:00:2e:00   Spanning-tree-(for-bridges)_00    STP    52    Conf. TC
+ Root = 32768/0/00:b1:bb:00:78:11  Cost = 0  Port = 0x8001
00:b3:00:02:2e:00   Spanning-tree-(for-bridges)_00    STP    21    Topology
Change Notification
```

Source:

1- https://learningnetwork.cisco.com/thread/57164

Task 9. Restore the original situation. Then start a ping from H1 to H2. Next, on a bridge machine that has disabled ports, bring down the bridge, disable STP and then bring up the bridge again. Describe what happens mentioning the state of the ports on all bridges. After how much time do the other bridges notice your disruptive actions? What is the downtime experienced by the ping session? Based on the STP protocol variables explain this downtime.

Answer:

In order to restore the original situation, I will power on the B1 again:

```
root@B1:~# ip link set bridge0 up
bridge0: port 2(eth1) entered listening state
bridge0: port 2(eth1) entered listening state
bridge0: port 1(eth0) entered listening state
bridge0: port 1(eth0) entered listening state
IPv6: ADDRCONF(NETDEV_UP): bridge0: link is not ready
root@B1:~# bridge0: port 1(eth0) received tcn bpdu
bridge0: topology change detected, propagating
bridge0: port 2(eth1) entered learning state
bridge0: port 1(eth0) entered learning state
bridge0: topology change detected, propagating
bridge0: port 2(eth1) entered forwarding state
bridge0: topology change detected, propagating
bridge0: port 1(eth0) entered forwarding state
IPv6: ADDRCONF(NETDEV_CHANGE): bridge0: link becomes ready


root@B1:~# brctl showstp bridge0
bridge0
```

```
 bridge id              8000.00b1bb007811
 designated root        8000.00b1bb007811
```

Now, I will start a ping from H1 to H2. Then on a bridge machine that has disabled ports (B3), I will bring down the bridge, disable STP and then bring up the bridge again.

```
root@B3:~# ip link set bridge0 down
bridge0: port 2(eth1) entered disabled state
bridge0: port 1(eth0) entered disabled state
bridge0: topology change detected, propagating
root@B3:~# brctl stp bridge0 off
root@B3:~# ip link set bridge0 up
bridge0: port 2(eth1) entered forwarding state
bridge0: port 2(eth1) entered forwarding state
bridge0: port 1(eth0) entered forwarding state
bridge0: port 1(eth0) entered forwarding state
root@B3:~# bridge0: received packet on eth0 with own address as source
address
bridge0: received packet on eth0 with own address as source address
bridge0: received packet on eth1 with own address as source address
bridge0: received packet on eth0 with own address as source address
bridge0: received packet on eth1 with own address as source address
bridge0: received packet on eth0 with own address as source address
bridge0: received packet on eth1 with own address as source address
bridge0: received packet on eth0 with own address as source address
bridge0: received packet on eth1 with own address as source address
bridge0: received packet on eth0 with own address as source address
bridge0: port 2(eth1) entered forwarding state
bridge0: port 1(eth0) entered forwarding state
```

Ping:

```
root@H1:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=117 ttl=64 time=1.08 ms
64 bytes from 10.0.0.2: icmp_req=118 ttl=64 time=1.13 ms
64 bytes from 10.0.0.2: icmp_req=119 ttl=64 time=1.06 ms
64 bytes from 10.0.0.2: icmp_req=120 ttl=64 time=1.05 ms
64 bytes from 10.0.0.2: icmp_req=121 ttl=64 time=1.27 ms
64 bytes from 10.0.0.2: icmp_req=122 ttl=64 time=1.15 ms
64 bytes from 10.0.0.2: icmp_req=123 ttl=64 time=1.01 ms
64 bytes from 10.0.0.2: icmp_req=124 ttl=64 time=0.989 ms
64 bytes from 10.0.0.2: icmp_req=125 ttl=64 time=0.971 ms
64 bytes from 10.0.0.2: icmp_req=126 ttl=64 time=1.20 ms
64 bytes from 10.0.0.2: icmp_req=127 ttl=64 time=11.3 ms
64 bytes from 10.0.0.2: icmp_req=128 ttl=64 time=1.03 ms
64 bytes from 10.0.0.2: icmp_req=129 ttl=64 time=1.10 ms
64 bytes from 10.0.0.2: icmp_req=130 ttl=64 time=1.11 ms
64 bytes from 10.0.0.2: icmp_req=131 ttl=64 time=1.31 ms
64 bytes from 10.0.0.2: icmp_req=132 ttl=64 time=0.875 ms
64 bytes from 10.0.0.2: icmp_req=133 ttl=64 time=1.08 ms
```

```
64 bytes from 10.0.0.2: icmp_req=134 ttl=64 time=0.958 ms
64 bytes from 10.0.0.2: icmp_req=135 ttl=64 time=0.962 ms
64 bytes from 10.0.0.2: icmp_req=136 ttl=64 time=1.07 ms
64 bytes from 10.0.0.2: icmp_req=137 ttl=64 time=1.29 ms
64 bytes from 10.0.0.2: icmp_req=138 ttl=64 time=0.878 ms
64 bytes from 10.0.0.2: icmp_req=139 ttl=64 time=0.971 ms
64 bytes from 10.0.0.2: icmp_req=140 ttl=64 time=1.44 ms
^C
--- 10.0.0.2 ping statistics ---
140 packets transmitted, 93 received, 33% packet loss, time 140095ms
rtt min/avg/max/mdev = 0.650/12.081/998.560/102.858 ms
```

B1:

```
root@B1:~# brctl showstp bridge0
bridge0
 bridge id              8000.00b1bb007811
 designated root        8000.00b1bb007811
 root port                 0                    path cost                  0
 max age                  20.00                 bridge max age
20.00
 hello time                2.00                 bridge hello time
2.00
 forward delay            15.00                 bridge forward delay
15.00
 ageing time             300.00
 hello timer               1.48                 tcn timer
0.00
 topology change timer     0.00                 gc timer
105.00
 flags


eth0 (1)
 port id                8001                    state
forwarding
 designated root        8000.00b1bb007811       path cost                100
 designated bridge      8000.00b1bb007811       message age timer
0.00
 designated port        8001                    forward delay timer
0.00
 designated cost           0                    hold timer
0.48
 flags

eth1 (2)
 port id                8002                    state
blocking
 designated root        8000.00b1bb007811       path cost                100
 designated bridge      8000.00b1bb007811       message age timer
19.52
```

```
 designated port        8001                        forward delay timer
0.00
 designated cost           0                        hold timer
0.00
 flags
```

B2:

```
root@B2:~# brctl showstp bridge0
bridge0
 bridge id              8000.00b2bb017811
 designated root        8000.00b1bb007811
 root port                 1                        path cost                100
 max age                  20.00                     bridge max age
20.00
 hello time                2.00                     bridge hello time
2.00
 forward delay            15.00                     bridge forward delay
15.00
 ageing time             300.00
 hello timer               0.00                     tcn timer
0.00
 topology change timer     0.00                     gc timer
55.43
 flags


eth0 (1)
 port id                8001                        state
forwarding
 designated root        8000.00b1bb007811           path cost                100
 designated bridge      8000.00b1bb007811           message age timer
18.87
 designated port        8001                        forward delay timer
0.00
 designated cost           0                        hold timer
0.00
 flags

eth1 (2)
 port id                8002                        state
blocking
 designated root        8000.00b1bb007811           path cost                100
 designated bridge      8000.00b1bb007811           message age timer
18.87
 designated port        8001                        forward delay timer
0.00
 designated cost           0                        hold timer
0.00
 flags
```

B3:

```
root@B3:~# brctl showstp bridge0
bridge0
 bridge id              8000.00b3bb027811
 designated root        8000.00b3bb027811
 root port                 0                    path cost                  0
 max age                  20.00                 bridge max age
20.00
 hello time                2.00                 bridge hello time
2.00
 forward delay            15.00                 bridge forward delay
15.00
 ageing time             300.00
 hello timer               1.87                 tcn timer
0.00
 topology change timer     0.00                 gc timer
115.03
 flags


eth0 (1)
 port id                8001                    state
forwarding
 designated root        8000.00b3bb027811       path cost                100
 designated bridge      8000.00b3bb027811       message age timer
0.00
 designated port        8001                    forward delay timer
0.00
 designated cost           0                    hold timer
0.87
 flags

eth1 (2)
 port id                8002                    state
forwarding
 designated root        8000.00b3bb027811       path cost                100
 designated bridge      8000.00b3bb027811       message age timer
0.00
 designated port        8002                    forward delay timer
0.00
 designated cost           0                    hold timer
0.87
 flags
```

As we see from the output above, since we powered off B3. both of it's ports changed to forwarding state. However, B1 and B2 still assume that B1 is the root bridge. If you we go back to our topology above we noticed that B1 ports was in forwarding state and B2 ports one on blocking and one in forwarding. But now B1 and B2 eth1 one port in blocking state. That happened because B3 doesn't have enabled STP. Which means if B1 left both interfaces in forwarding state it will enter a loop problem.

Now, let's check the pcap file:

```
138397  276588.488006    00:b1:00:00:2e:00    Spanning-tree-(for-bridges)_00
STP    52    Conf. Root = 32768/0/00:b1:bb:00:78:11  Cost = 0  Port = 0x8001
138398  276588.488139    00:b1:00:00:2e:11    Spanning-tree-(for-bridges)_00
STP    52    Conf. Root = 32768/0/00:b1:bb:00:78:11  Cost = 0  Port = 0x8002
138399  276588.490064    00:b2:00:01:2e:11    Spanning-tree-(for-bridges)_00
STP    21    Topology Change Notification
```

The time do the other bridges notice the disruptive actions and the downtime experienced by the ping session was almost immediately noticed the change in the topology. To be more specific, it took 2 second to notice the change. Why it is two seconds ? (because two second the hello timer. Which means B1 will only send new BPDU packets every two seconds. In addition to that, bridges will react to the received BPDU immediately as valid information since the the message age time is 0.

Task 10. Enable STP again on all bridges. Does the network come back to the initial state? Why?

Answer:

Enable the stp on B3:

```
root@B3:~# brctl stp bridge0 on
```

As we see in the test below, after we enable stp again on B3 it immediately changed it's topology and assume that B1 is the root bridge (since it recived BPDu from b1 which is sent every 2 seconds) and since B1 has the lowest MAC address.

Test:

```
root@B3:~# brctl showstp bridge0
bridge0
 bridge id              8000.00b3bb027811
 designated root        8000.00b1bb007811
 root port                 1                    path cost                  100
 max age                  20.00                 bridge max age
20.00
 hello time                2.00                 bridge hello time
2.00
 forward delay            15.00                 bridge forward delay
15.00
 ageing time             300.00
 hello timer               0.00                 tcn timer
0.00
 topology change timer     0.00                 gc timer
20.63
 flags
```

```
eth0 (1)
 port id                8001                    state
forwarding
 designated root        8000.00b1bb007811       path cost                100
 designated bridge      8000.00b1bb007811       message age timer
19.67
 designated port        8001                    forward delay timer
0.00
 designated cost           0                    hold timer
0.00
 flags

eth1 (2)
 port id                8002                    state
blocking
 designated root        8000.00b1bb007811       path cost                100
 designated bridge      8000.00b1bb007811       message age timer
19.67
 designated port        8002                    forward delay timer
0.00
 designated cost           0                    hold timer
0.00
 flags
```

802.1Q aka VLANs

Task 12. What command do you need to type to add a VLAN id 5 to eth0?

Answer:

Add a VLAN id 5 to eth0.5 (0.5 since it's Vlan)

```
ip link add link eth0 name eth0.5 type vlan id 5
```

Source:

1- https://www.cyberciti.biz/tips/howto-configure-linux-virtual-local-area-network-vlan.html

Task 13. Start the vlan.conf configuration in the pogo repository and configure each host manually, as follows:

- H1: eth0.10,10.0.10.1/24 ; eth0.30,10.0.30.1/24
- H2: eth0.10,10.0.10.2/24 ; eth0.20,10.0.20.2/24 ; eth0.30,10.0.30.2/24

- B1: bridge0 containing eth0.10, eth1.10
- B2: eth1.20,10.0.20.1/24
- B3: bridge0 containing eth0.30, eth1.30

Note: ethX.Y,Z/T means ethX with VLAN id Y configured having IP address Z with subnetmask T. Hint: you might want to write down the commands in a text file first, to make sure you got everything right.

Answer:

When I'm trying to run create vlan.cfg I get a lot of errors, so I will destroy the whole image and create it again.

```
kotaiba@bristol:~$ sudo xen-create-image --hostname=Guest-02 --dist=zesty --
size=10Gb --swap=1024Mb --lvm=VolumeGroupXen --fs=ext3 --vcpus=2 --
ip=145.100.108.83 --netmask=255.255.255.240 --broadcast=145.100.108.95 --
memory=4096Mb --gateway=145.100.108.81

kotaiba@bristol:~$ sudo xl create /etc/xen/Guest-02.cfg
Parsing config from /etc/xen/Guest-02.cfg
```

As in Lab 3, I fixed the source.list and dns problem and I installed again all the required packages. Also, I created a user for the SSH login.

Now, lets get pogo again from gitlab:

```
root@Guest-02:~#  git clone https://gitlab.os3.nl/Networking/pogo.git
Cloning into 'pogo'...
remote: Counting objects: 91, done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 91 (delta 44), reused 44 (delta 22)
Unpacking objects: 100% (91/91), done.
```

I got some errors so I tried to figure out what is happening (I forgot to add stuff also from Lab 3)

So I try again now:

```
root@Guest-02:~/pogo# python pogo.py create vlan.cfg
INFO: Creating ogo-ovs0 switch
ovs-vsctl: cannot create a bridge named ogo-ovs0 because a bridge named ogo-
ovs0 already exists
ovs-vsctl: multiple rows in Mirror match "ogo-ovs0mirror"
INFO: Creating ogo-ovs1 switch
ovs-vsctl: cannot create a bridge named ogo-ovs1 because a bridge named ogo-
ovs1 already exists
ovs-vsctl: multiple rows in Mirror match "ogo-ovs1mirror"
INFO: Creating host host1. This will take some time.
INFO: Creating host host2. This will take some time.
INFO: Creating host bridge1. This will take some time.
```

```
INFO: Creating host bridge2. This will take some time.
INFO: Creating host bridge3. This will take some time.

root@Guest-02:~/pogo# lxc list
+---------+---------+------+------+------------+-----------+
|  NAME   |  STATE  | IPV4 | IPV6 |    TYPE    | SNAPSHOTS |
+---------+---------+------+------+------------+-----------+
| bridge1 | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| bridge2 | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| bridge3 | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| host1   | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| host2   | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| ogobase | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
```

CREDITS TO ARNO TO HELP ME IN THIS.

I forgot to start the pogo configuration that's why there was no screen attached:

```
root@Guest-02:~/pogo# python pogo.py start vlan.cfg
INFO: Starting sniffer on ogo-ovs0 switch
INFO: Starting sniffer on ogo-ovs1 switch
INFO: Starting host host1.
INFO: Starting host host2.
INFO: Starting host bridge1.
INFO: Starting host bridge2.
INFO: Starting host bridge3.
There are screens on:
	18284.bridge3	(11/14/17 12:11:54)	(Detached)
	17356.bridge2	(11/14/17 12:11:50)	(Detached)
	16749.bridge1	(11/14/17 12:11:47)	(Detached)
	16358.host2	(11/14/17 12:11:44)	(Detached)
	15943.host1	(11/14/17 12:11:41)	(Detached)
5 Sockets in /run/screen/S-root.


root@Guest-02:~/pogo# lxc list
+---------+---------+------+------+------------+-----------+
|  NAME   |  STATE  | IPV4 | IPV6 |    TYPE    | SNAPSHOTS |
+---------+---------+------+------+------------+-----------+
| bridge1 | RUNNING |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| bridge2 | RUNNING |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| bridge3 | RUNNING |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| host1   | RUNNING |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| host2   | RUNNING |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
| ogobase | STOPPED |      |      | PERSISTENT | 0         |
+---------+---------+------+------+------------+-----------+
```

Now, I will configure each host manually:

host1:

```
root@host1:~# ip link add link eth0 name eth0.10 type vlan id 10
root@host1:~# ip addr add 10.0.10.1/24 brd 10.0.10.255 dev eth0.10
root@host1:~# ip link add link eth0 name eth0.30 type vlan id 30
root@host1:~# ip addr add 10.0.30.1/24 brd 10.0.30.255 dev eth0.30
root@host1:~# ip link set dev eth0.10 up
root@host1:~# ip link set dev eth0.30 up
root@host1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:33:5f:04
          inet6 addr: fe80::216:3eff:fe33:5f04/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:44 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3464 (3.4 KB)  TX bytes:2178 (2.1 KB)

eth0.10   Link encap:Ethernet  HWaddr 00:16:3e:33:5f:04
          inet addr:10.0.10.1  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe33:5f04/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:656 (656.0 B)

eth0.30   Link encap:Ethernet  HWaddr 00:16:3e:33:5f:04
          inet addr:10.0.30.1  Bcast:10.0.30.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe33:5f04/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:656 (656.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:520 (520.0 B)  TX bytes:520 (520.0 B)
```

host2:

```
root@host2:~# ip link add link eth0 name eth0.10 type vlan id 10
root@host2:~# ip addr add 10.0.10.2/24 brd 10.0.10.255 dev eth0.10
root@host2:~# ip link add link eth0 name eth0.20 type vlan id 20
root@host2:~# ip addr add 10.0.20.2/24 brd 10.0.20.255 dev eth0.20
root@host2:~# ip link add link eth0 name eth0.30 type vlan id 30
root@host2:~# ip addr add 10.0.30.2/24 brd 10.0.30.255 dev eth0.30
root@host2:~# ip link set dev eth0.10 up
root@host2:~# ip link set dev eth0.20 up
root@host2:~# ip link set dev eth0.30 up
root@host2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:d7:46:50
          inet6 addr: fe80::216:3eff:fed7:4650/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:47 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3674 (3.6 KB)  TX bytes:2764 (2.7 KB)

eth0.10   Link encap:Ethernet  HWaddr 00:16:3e:d7:46:50
          inet addr:10.0.10.2  Bcast:10.0.10.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fed7:4650/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:656 (656.0 B)

eth0.20   Link encap:Ethernet  HWaddr 00:16:3e:d7:46:50
          inet addr:10.0.20.2  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fed7:4650/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:586 (586.0 B)

eth0.30   Link encap:Ethernet  HWaddr 00:16:3e:d7:46:50
          inet addr:10.0.30.2  Bcast:10.0.30.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fed7:4650/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:586 (586.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:520 (520.0 B)  TX bytes:520 (520.0 B)
```

brdige1:

```
root@bridge1:~# brctl addbr bridge0
root@bridge1:~# brctl stp bridge0 on
root@bridge1:~# ip link add link eth0 name eth0.10 type vlan id 10
root@bridge1:~# ip link add link eth1 name eth1.10 type vlan id 10
root@bridge1:~# brctl addif bridge0 eth0.10
root@bridge1:~# brctl addif bridge0 eth1.10
root@bridge1:~# ip link set dev eth1.10 up
root@bridge1:~# ip link set dev eth0.10 up
root@bridge1:~# ip link set dev bridge0 up
root@bridge1:~# ifconfig
bridge0   Link encap:Ethernet  HWaddr 00:16:3e:72:f4:fa
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:16:3e:ef:e7:da
          inet6 addr: fe80::216:3eff:feef:e7da/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:63 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4890 (4.8 KB)  TX bytes:1452 (1.4 KB)

eth1      Link encap:Ethernet  HWaddr 00:16:3e:72:f4:fa
          inet6 addr: fe80::216:3eff:fe72:f4fa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:72 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5616 (5.6 KB)  TX bytes:1522 (1.5 KB)

eth0.10   Link encap:Ethernet  HWaddr 00:16:3e:ef:e7:da
          inet6 addr: fe80::216:3eff:feef:e7da/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:516 (516.0 B)

eth1.10   Link encap:Ethernet  HWaddr 00:16:3e:72:f4:fa
```

```
          inet6 addr: fe80::216:3eff:fe72:f4fa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:586 (586.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:536 (536.0 B)  TX bytes:536 (536.0 B)
```

bridge2:

```
root@bridge2:~# ip link add link eth1 name eth1.20 type vlan id 20
root@bridge2:~# ip addr add 10.0.20.1/24 brd 10.0.20.255 dev eth1.20
root@bridge2:~# ip link set dev eth1.20 up
root@bridge2:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:c1:9f:2c
          inet6 addr: fe80::216:3eff:fec1:9f2c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:73 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5686 (5.6 KB)  TX bytes:936 (936.0 B)

eth1      Link encap:Ethernet  HWaddr 00:16:3e:cb:fc:44
          inet6 addr: fe80::216:3eff:fecb:fc44/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:83 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6482 (6.4 KB)  TX bytes:1202 (1.2 KB)

eth1.20   Link encap:Ethernet  HWaddr 00:16:3e:cb:fc:44
          inet addr:10.0.20.1  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fecb:fc44/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:266 (266.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:536 (536.0 B)  TX bytes:536 (536.0 B)
```

bridge3:

```
root@bridge3:~# brctl addbr bridge0
root@bridge3:~# brctl stp bridge0 on
root@bridge3:~# ip link add link eth0 name eth0.30 type vlan id 30
root@bridge3:~# ip link add link eth1 name eth1.30 type vlan id 30
root@bridge3:~# brctl addif bridge0 eth0.30
root@bridge3:~# brctl addif bridge0 eth1.30
root@bridge3:~# ip link set dev eth0.30 up
root@bridge3:~# ip link set dev eth1.30 up
root@bridge3:~# ip link set dev bridge0 up
root@bridge3:~# ifconfig
bridge0   Link encap:Ethernet  HWaddr 00:16:3e:0f:91:2e
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:16:3e:3a:31:37
          inet6 addr: fe80::216:3eff:fe3a:3137/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:71 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5450 (5.4 KB)  TX bytes:1574 (1.5 KB)

eth1      Link encap:Ethernet  HWaddr 00:16:3e:0f:91:2e
          inet6 addr: fe80::216:3eff:fe0f:912e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7042 (7.0 KB)  TX bytes:1574 (1.5 KB)

eth0.30   Link encap:Ethernet  HWaddr 00:16:3e:3a:31:37
          inet6 addr: fe80::216:3eff:fe3a:3137/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:638 (638.0 B)

eth1.30   Link encap:Ethernet  HWaddr 00:16:3e:0f:91:2e
          inet6 addr: fe80::216:3eff:fe0f:912e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:638 (638.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:536 (536.0 B)  TX bytes:536 (536.0 B)
```

Task 14. Once you have have configured everything perform the following ping tests:

- from 10.0.10.1 to 10.0.10.2
- from 10.0.30.1 to 10.0.30.2

Answer:

from 10.0.10.1 to 10.0.10.2:

```
root@host1:~# ping 10.0.10.2
PING 10.0.10.2 (10.0.10.2) 56(84) bytes of data.
64 bytes from 10.0.10.2: icmp_seq=1 ttl=64 time=2.30 ms
64 bytes from 10.0.10.2: icmp_seq=2 ttl=64 time=0.117 ms
^C
--- 10.0.10.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.117/1.212/2.307/1.095 ms
```

from 10.0.30.1 to 10.0.30.2:

```
root@host1:~# ping 10.0.30.2
PING 10.0.30.2 (10.0.30.2) 56(84) bytes of data.
64 bytes from 10.0.30.2: icmp_seq=1 ttl=64 time=1.55 ms
64 bytes from 10.0.30.2: icmp_seq=2 ttl=64 time=0.146 ms
^C
--- 10.0.30.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.146/0.850/1.554/0.704 ms
```

Task 15. Explain the structure of a Ethernet frame that contains an ICMP echo request with source 10.0.20.1 captured by the sniffer on the righthand side of the network (between H2 and B2). Explain all the VLAN related fields in detail. Next you will configure H2 to act as a “router-on-a-stick”.

Answer:

The source of 10.0.20.1 IP address is bridge2:

```
root@bridge2:~# ifconfig eth1.20
eth1.20   Link encap:Ethernet  HWaddr 00:16:3e:cb:fc:44
          inet addr:10.0.20.1  Bcast:10.0.20.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fecb:fc44/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:168 (168.0 B)  TX bytes:1006 (1.0 KB)
```

Now, I will ping then check the the pcap file (ogo-ovs1.pcap since it's eth1.*):

```
root@bridge2:~# ping 10.0.20.2
PING 10.0.20.2 (10.0.20.2) 56(84) bytes of data.
64 bytes from 10.0.20.2: icmp_seq=1 ttl=64 time=1.34 ms
64 bytes from 10.0.20.2: icmp_seq=2 ttl=64 time=0.103 ms
64 bytes from 10.0.20.2: icmp_seq=3 ttl=64 time=0.097 ms
64 bytes from 10.0.20.2: icmp_seq=4 ttl=64 time=0.101 ms
64 bytes from 10.0.20.2: icmp_seq=5 ttl=64 time=0.102 ms
64 bytes from 10.0.20.2: icmp_seq=6 ttl=64 time=0.105 ms
64 bytes from 10.0.20.2: icmp_seq=7 ttl=64 time=0.098 ms
64 bytes from 10.0.20.2: icmp_seq=8 ttl=64 time=0.103 ms
^C
--- 10.0.20.2 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7146ms
rtt min/avg/max/mdev = 0.097/0.257/1.347/0.411 ms
```

Get the pcap file:

```
kalachkar@desktop-41:~$ scp kotaiba@145.100.108.83:/tmp/ogo-ovs1.pcap
Desktop/pcaps/
kotaiba@145.100.108.83's password:
ogo-ovs1.pcap                                                      100%
75KB  20.7MB/s   00:00
```

The structure of a Ethernet frame that contains an ICMP echo request with source 10.0.20.1:

- **Source:** “00:16:3e:cb:fc:44”, MAC address of the sender interface “10.0.20.1”.
- **Destination:** “00:16:3e:d7:46:50”, MAC address of the receiver “10.0.20.2”.
- **EtherType:** “0x8100”, used to indicate which protocol is encapsulated in the payload of the frame.

- **PCP (Priority Code Point):** “0”, a 3-bit field which refers to the IEEE 802.1p class of service and maps to the frame priority level “0 (best effort, default)”.
- **CFI (Canonical Format Indicator):** “0”, indicates whether the following 12 bits of VLAN identifier conform to Ethernet or not. For Ethernet frames, this bit is always set to 0.
- **VID (VLAN Identifier):** “20”, a 12-bit field specifying the VLAN to which the frame belongs.

Source:

1- https://en.wikipedia.org/wiki/EtherType, 2- https://en.wikipedia.org/wiki/IEEE_802.1Q 3- https://www.juniper.net/documentation/en_US/junos/topics/concept/layer-2-networking-ethernet-frame-forwarding-802-1q-vlan-tag-mx-solutions.html

Task 16. Add static routes on H1, H2, B2 to allow B2 to reach H1 over both VLANs. Display the routing tables on H1, H2 and B2 and the output of ping from B2 to 10.0.10.1 and 10.0.30.1.

Answer:

First let's add static routes on H1, H2, B2:

host1:

First, we add in /etc/iproute2/rt_tables:

```
1 vlan10
2 vlan30
```

Then:

```
root@host1:~# ip route add 10.0.10.0/24 dev eth0.10 src 10.0.10.1 table
vlan10
root@host1:~# ip route add 10.0.20.0/24 via 10.0.10.2 dev eth0.10 table
vlan10
root@host1:~# ip route add 10.0.30.0/24 dev eth0.30 src 10.0.30.1 table
vlan30
root@host1:~# ip route add 10.0.20.0/24 via 10.0.30.2 dev eth0.30 table
vlan30
root@host1:~# ip rule add from 10.0.30.1/32 table vlan30
root@host1:~# ip rule add to 10.0.30.1/32 table vlan30
root@host1:~# ip rule add from 10.0.10.1/32 table vlan10
root@host1:~# ip rule add to 10.0.10.1/32 table vlan10
```

host2:

I will enable IPv4 routing:

```
root@host2:~# sysctl net.ipv4.ip_forward
```

```
net.ipv4.ip_forward = 1
```

bridge2:

```
root@bridge2:~# ip route add 10.0.10.0/24 via 10.0.20.2
root@bridge2:~# ip route add 10.0.30.0/24 via 10.0.20.2
```

Now, Let's test:

```
root@bridge2:~# ping 10.0.10.1
PING 10.0.10.1 (10.0.10.1) 56(84) bytes of data.
64 bytes from 10.0.10.1: icmp_seq=1 ttl=63 time=1.80 ms
64 bytes from 10.0.10.1: icmp_seq=2 ttl=63 time=0.159 ms
64 bytes from 10.0.10.1: icmp_seq=3 ttl=63 time=0.172 ms
64 bytes from 10.0.10.1: icmp_seq=4 ttl=63 time=0.156 ms
64 bytes from 10.0.10.1: icmp_seq=5 ttl=63 time=0.205 ms
^C
--- 10.0.10.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4056ms
rtt min/avg/max/mdev = 0.156/0.498/1.800/0.651 ms


root@bridge2:~# ping 10.0.30.1
PING 10.0.30.1 (10.0.30.1) 56(84) bytes of data.
64 bytes from 10.0.30.1: icmp_seq=1 ttl=63 time=1.46 ms
64 bytes from 10.0.30.1: icmp_seq=2 ttl=63 time=0.159 ms
64 bytes from 10.0.30.1: icmp_seq=3 ttl=63 time=0.149 ms
64 bytes from 10.0.30.1: icmp_seq=4 ttl=63 time=0.158 ms
^C
--- 10.0.30.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3031ms
rtt min/avg/max/mdev = 0.149/0.482/1.463/0.566 ms
```

Now, Let's check the routing tables:

host1:

```
root@host1:~# ip route list table vlan10
10.0.10.0/24 dev eth0.10  scope link  src 10.0.10.1
10.0.20.0/24 via 10.0.10.2 dev eth0.10
root@host1:~# ip route list table vlan30
10.0.20.0/24 via 10.0.30.2 dev eth0.30
10.0.30.0/24 dev eth0.30  scope link  src 10.0.30.1
root@host1:~# ip route
10.0.10.0/24 dev eth0.10  proto kernel  scope link  src 10.0.10.1
10.0.30.0/24 dev eth0.30  proto kernel  scope link  src 10.0.30.1
```

host2:

```
root@host2:~# ip route
10.0.10.0/24 dev eth0.10  proto kernel  scope link  src 10.0.10.2
```

```
10.0.20.0/24 dev eth0.20  proto kernel  scope link  src 10.0.20.2
10.0.30.0/24 dev eth0.30  proto kernel  scope link  src 10.0.30.2
```

bridge2:

```
root@bridge2:~# ip route
10.0.10.0/24 via 10.0.20.2 dev eth1.20
10.0.20.0/24 dev eth1.20  proto kernel  scope link  src 10.0.20.1
10.0.30.0/24 via 10.0.20.2 dev eth1.20
```

Task 17. What is the maximum number of VLAN IDs active on a network segment? Be precise!

Answer:

Under IEEE 802.1Q, the maximum number of VLANs on a given Ethernet network is 4,094 (the 4,096 provided for by the 12-bit VID field minus reserved values 0x000 and 0xFFF).

Source:

1- https://en.wikipedia.org/wiki/Virtual_LAN

Task 18. Draw a network diagram that depicts the configuration. Make sure the VLANs are clearly marked.

Answer: