Q1 Operation of a basic Ethernet switch or bridge (and additional bridge questions)

4 Points

Q1.1 Setting up the bridge

1 Point

Complete this answer at the end of the "Set up the bridge" section of the "Operation of a basic Ethernet switch or bridge" experiment.

Upload `ifconfig` output

First, upload a screenshot showing the `ifconfig` output on each of the four "nodes" (node-1, node-2, node-3, and node-4). Annotate each screenshot to draw a circle or box around the MAC address on the experiment interface.

Also upload a screenshot of the `ifconfig` output on the bridge, and annotate it to draw a circle or box around the MAC address on each experiment interface.

Make sure each screenshot includes the terminal prompt + complete command + output + annotation.

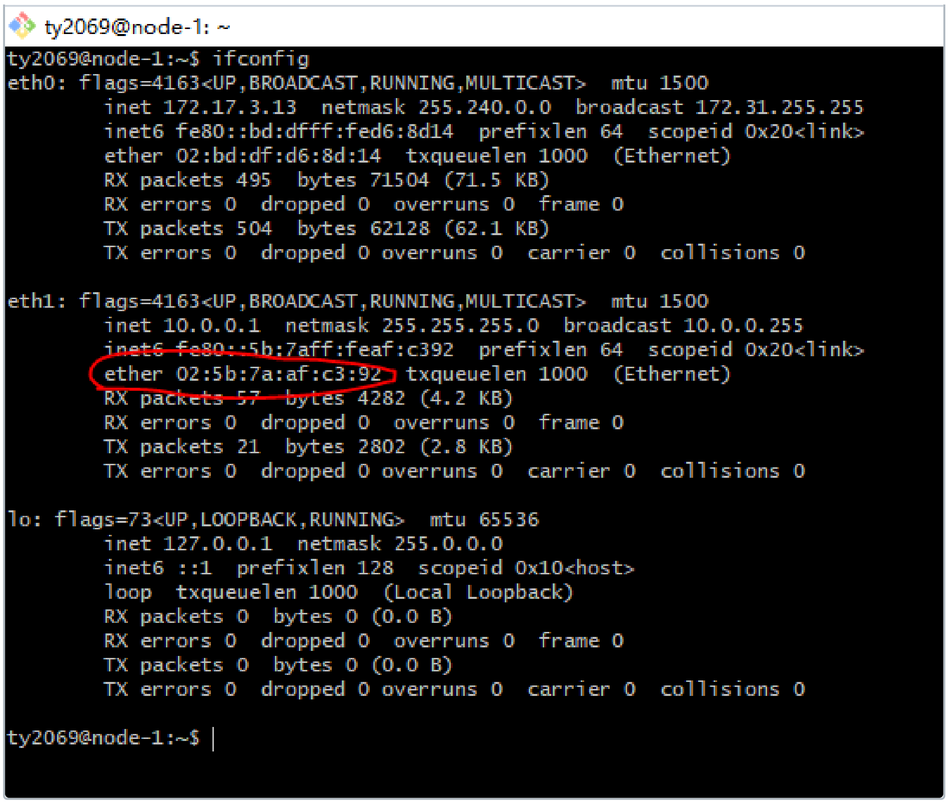
▾ bridge-ifconfig.PNG Download

ty2069@bridge: ~

```
ty2069@bridge:~$ ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::1b:c9ff:fef2:9aa4  prefixlen 64  scopeid 0x20<link>
        ether 02:1b:c9:f2:9a:a4  txqueuelen 1000  (Ethernet)
        RX packets 11  bytes 616 (616.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.3.8  netmask 255.240.0.0  broadcast 172.31.255.255
        inet6 fe80::9d:daff:feca:523d  prefixlen 64  scopeid 0x20<link>
        ether 02:9d:da:ca:52:3d  txqueuelen 1000  (Ethernet)
        RX packets 15703  bytes 38306129 (38.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3106  bytes 313073 (313.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 02:1b:c9:f2:9a:a4  txqueuelen 1000  (Ethernet)
        RX packets 23  bytes 1480 (1.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 39  bytes 3879 (3.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 02:ae:96:e1:e4:49  txqueuelen 1000  (Ethernet)
        RX packets 22  bytes 1504 (1.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 3558 (3.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 02:fa:f0:9d:3d:31  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 3668 (3.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 39  bytes 3628 (3.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 02:73:34:3c:2e:e1  txqueuelen 1000  (Ethernet)
        RX packets 25  bytes 1877 (1.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 3558 (3.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ty2069@bridge:~$ |
```

node1-ifconfig.PNG

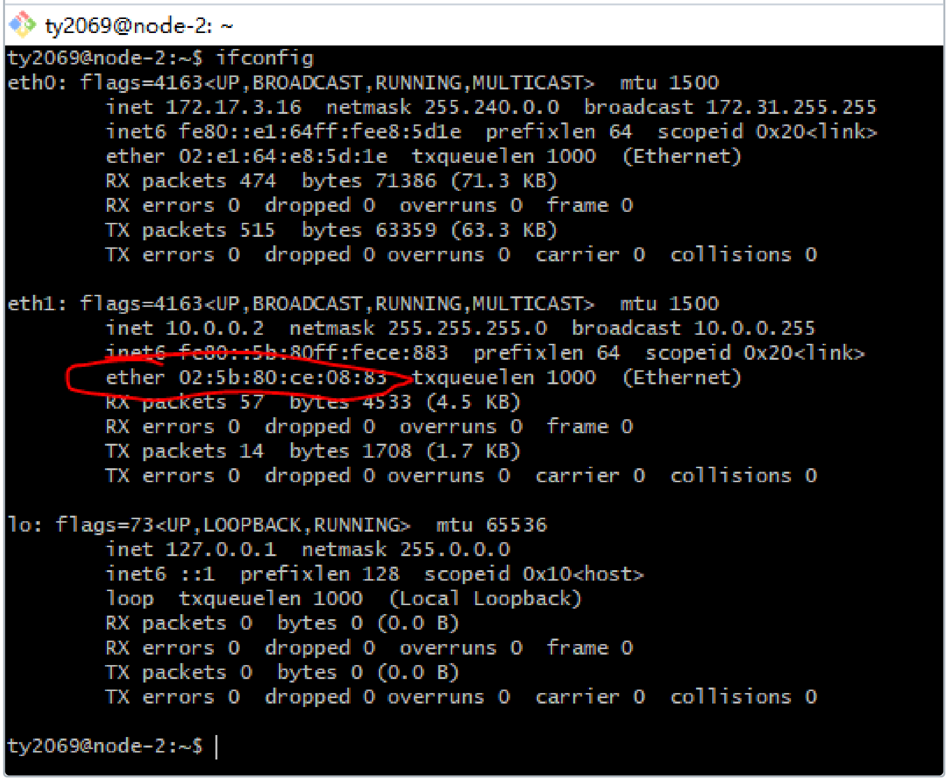
Download



ty2069@node-1: ~

```
ty2069@node-1:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.3.13  netmask 255.240.0.0  broadcast 172.31.255.255
        inet6 fe80::bd:dfff:fed6:8d14  prefixlen 64  scopeid 0x20<link>
        ether 02:bd:df:d6:8d:14  txqueuelen 1000  (Ethernet)
        RX packets 495  bytes 71504 (71.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 504  bytes 62128 (62.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::5b:7aff:feaf:c392  prefixlen 64  scopeid 0x20<link>
        ether 02:5b:7a:af:c3:92  txqueuelen 1000  (Ethernet)
        RX packets 57  bytes 4282 (4.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 21  bytes 2802 (2.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ty2069@node-1:~$
```

▾ node2-ifconfig.PNG Download



ty2069@node-2: ~

```
ty2069@node-2:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.3.16  netmask 255.240.0.0  broadcast 172.31.255.255
        inet6 fe80::e1:64ff:fee8:5d1e  prefixlen 64  scopeid 0x20<link>
        ether 02:e1:64:e8:5d:1e  txqueuelen 1000  (Ethernet)
        RX packets 474  bytes 71386 (71.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 515  bytes 63359 (63.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::5b:80ff:fece:883  prefixlen 64  scopeid 0x20<link>
        ether 02:5b:80:ce:08:83  txqueuelen 1000  (Ethernet)
        RX packets 57  bytes 4533 (4.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 1708 (1.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ty2069@node-2:~$
```

▾ node3-ifconfig.PNG Download

ty2069@node-3: ~

```
ty2069@node-3:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.3.17  netmask 255.240.0.0  broadcast 172.31.255.255
        inet6 fe80::70:a6ff:fed0:f8a0  prefixlen 64  scopeid 0x20<link>
        ether 02:70:a6:d0:f8:a0  txqueuelen 1000  (Ethernet)
        RX packets 516  bytes 71018 (71.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 494  bytes 62056 (62.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.3  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::5f:81ff:fe55:10ed  prefixlen 64  scopeid 0x20<link>
        ether 02:5f:81:55:10:ed  txqueuelen 1000  (Ethernet)
        RX packets 56  bytes 4226 (4.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 4147 (4.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ty2069@node-3:~$ |
```

▾ node4-ifconfig.PNG Download

ty2069@node-4: ~

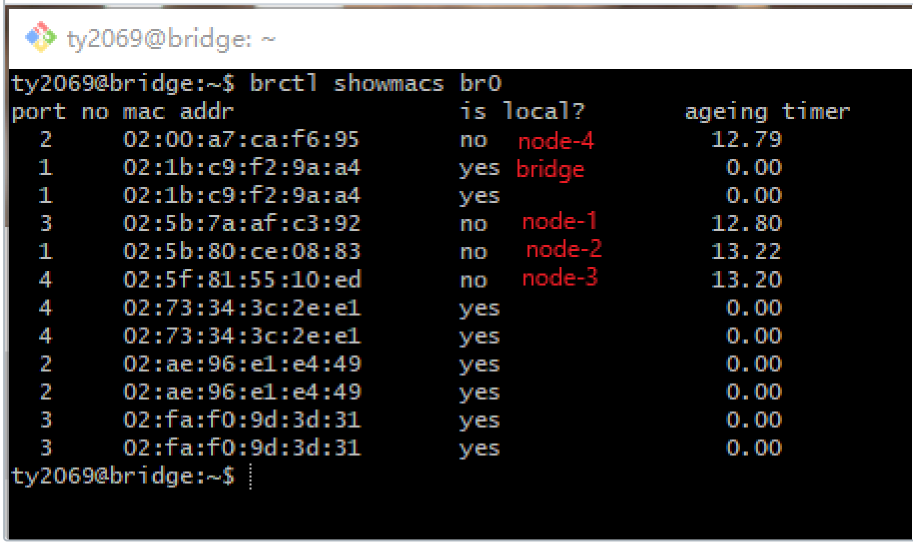
```
ty2069@node-4:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.3.18  netmask 255.240.0.0  broadcast 172.31.255.255
        inet6 fe80::2b:6bff:fe2c:dc83  prefixlen 64  scopeid 0x20<link>
        ether 02:2b:6b:2c:dc:83  txqueuelen 1000  (Ethernet)
        RX packets 480  bytes 72261 (72.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 531  bytes 64774 (64.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.4  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::a7ff:feca:f695  prefixlen 64  scopeid 0x20<link>
        ether 02:00:a7:ca:f6:95  txqueuelen 1000  (Ethernet)
        RX packets 56  bytes 4226 (4.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17  bytes 2112 (2.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ty2069@node-4:~$
```

Upload `brctl` output

Next, upload the output of `brctl showmacs br0` on the bridge, while the bridge is aware of all of the nodes' MAC addresses. (There should be four different local addresses and four different non-local addresses in this output) Annotate this screenshot - next to each line in the output, write the name of the host that this address belongs to (bridge, node-1, node-2, node-3, or node-4.) Also draw a box or a circle around the bridge port number in each line of output.

Make sure the screenshot includes the terminal prompt + complete command + output + annotation.

▾ bridge-showacs.png Download



```
ty2069@bridge: ~
ty2069@bridge:~$ brctl showmacs br0
port no mac addr                is local?       ageing timer
  2     02:00:a7:ca:f6:95       no   node-4       12.79
  1     02:1b:c9:f2:9a:a4       yes  bridge        0.00
  1     02:1b:c9:f2:9a:a4       yes                0.00
  3     02:5b:7a:af:c3:92       no   node-1       12.80
  1     02:5b:80:ce:08:83       no   node-2       13.22
  4     02:5f:81:55:10:ed       no   node-3       13.20
  4     02:73:34:3c:2e:e1       yes                0.00
  4     02:73:34:3c:2e:e1       yes                0.00
  2     02:ae:96:e1:e4:49       yes                0.00
  2     02:ae:96:e1:e4:49       yes                0.00
  3     02:fa:f0:9d:3d:31       yes                0.00
  3     02:fa:f0:9d:3d:31       yes                0.00
ty2069@bridge:~$
```

Fill in table

Finally, fill in the empty cells in the following table. For each bridge port (1, 2, 3, 4), indicate:

- which interface on the bridge - `eth1`, `eth2`, `eth3`, `eth4` - is on that port, and the MAC address of that interface.
- which host (node-1, node-2, node-3, or node-4) is on that port, and its MAC address

Bridge port	Bridge interface, MAC	Hostname, MAC
1		
2		
3		
4		

You can write your answers in the text input field, or upload a PDF, PNG, or JPG image of your table.

Bridge port Bridge interface, MAC Hostname, MAC
1 eth1, 02:1b:c9:f2:9a:a4 node-2, 02:5b:80:ce:08:83
2 eth2, 02:ae:96:e1:e4:49 node-4, 02:00:a7:ca:f6:95
3 eth3, 02:fa:f0:9d:3d:31 node-1, 02:5b:7a:af:c3:92
4 eth4, 02:73:34:3c:2e:e1 node-3, 02:5f:81:55:10:ed

table.PNG Download

Bridge port	Bridge interface, MAC	Hostname, MAC
1	eth1, 02:1b:c9:f2:9a:a4	node-2, 02:5b:80:ce:08:83
2	eth2, 02:ae:96:e1:e4:49	node-4, 02:00:a7:ca:f6:95
3	eth3, 02:fa:f0:9d:3d:31	node-1, 02:5b:7a:af:c3:92
4	eth4, 02:73:34:3c:2e:e1	node-3, 02:5f:81:55:10:ed

Q1.2 Learning MAC addresses

1 Point

Complete this answer as you run the "Learning MAC addresses" section of the "Operation of a basic Ethernet switch or bridge" experiment.

Make an ordered list of the following six events, as they occur in this section of the experiment:

- the bridge learns the MAC address of node-1,
- the bridge learns the MAC address of node-2,
- node-1 sends the first ping request (with seq 1) to node-2,
- node-2 sends the first ping reply (with seq 1) to node-1,
- node-2 receives the first ping request (with seq 1) from node-1,
- node-1 receives the first ping reply (with seq 1) from node-2.

(By "make an ordered list", I mean: note which event from the list above happened first, which happened second, etc.)

For each event, include either

- a frame from your `tcpdump` output or
- a line from the `bridge monitor fdb` output

that shows each event occurring. If the same frame appears in the `tcpdump` output on multiple hosts, indicate which host it appears on when this event occurs. (For example, a frame may appear in the `tcpdump` output of one node when it is sent, and in the `tcpdump` output of a different node when it is received - those are two different events.)

You can copy or paste from your terminal or show a screenshot, but make sure your screenshot is cropped to show only one packet from `tcpdump` or one line of `bridge monitor` output.

1st Event:

node-1 sends the first ping request (with seq 1) to node-2

Packet from `tcpdump` or line from `bridge monitor` showing this event:

03:17:14.628108 02:68:4f:e3:34:42 > 02:e8:28:ec:d4:31, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 8896, seq 1, length 64

No files uploaded

2nd Event:

the bridge learns the MAC address of node-1

Packet from `tcpdump` or line from `bridge monitor` showing this event:

02:68:4f:e3:34:42 dev eth2 master br0

No files uploaded

3rd Event:

node-2 receives the first ping request (with seq 1) from node-1,

Packet from `tcpdump` or line from `bridge monitor` showing this event:

03:17:14.628108 02:68:4f:e3:34:42 > 02:e8:28:ec:d4:31, ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.2: ICMP echo request, id 8896, seq 1, length 64

No files uploaded

4th Event:

node-2 sends the first ping reply (with seq 1) to node-1,

Packet from `tcpdump` or line from `bridge monitor` showing this event:

03:17:14.628178 02:e8:28:ec:d4:31 > 02:68:4f:e3:34:42, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 8896, seq 1, length 64

No files uploaded

5th Event:

the bridge learns the MAC address of node-2,

Packet from `tcpdump` or line from `bridge monitor` showing this event:

02:e8:28:ec:d4:31 dev eth3 master br0

No files uploaded

6th Event:

node-1 receives the first ping reply (with seq 1) from node-2.

Packet from `tcpdump` or line from `bridge monitor` showing this event:

03:17:14.628178 02:e8:28:ec:d4:31 > 02:68:4f:e3:34:42, ethertype IPv4 (0x0800), length 98: 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 8896, seq 1, length 64

No files uploaded

Q1.3 Effect of a smaller collision domain

1 Point

Run the "Effect of a smaller collision domain" section of the "Operation of a basic Ethernet switch or bridge" experiment.

iperf output with MAC learning

Upload a screenshot showing the `iperf` receiver output *with* MAC learning. Annotate it to draw a box or a circle around the data rate that was observed in each case (this should be a single value, given in kilobits per second).

▾ iperf-with-MAC.PNG Download

```
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-94.3 sec  10.8 MBytes   957 Kbits/sec
```

iperf output without MAC learning (bridge acts like a hub)

Upload a screenshot showing the `iperf` receiver output *without* MAC learning. Annotate it to draw a box or a circle around the data rate that was observed in each case (this should be a single value, given in kilobits per second).

▾ iperf-without-MAC.PNG Download

```
[  4] local 10.0.0.4 port 5001 connected with 10.0.0.2 port 47624
[  4]  0.0-95.0 sec  4.88 MBytes  431 Kbits/sec
```

Additional experiment

Then, run the following variation:

With MAC learning turned ON, run an `iperf` receiver on node-4:

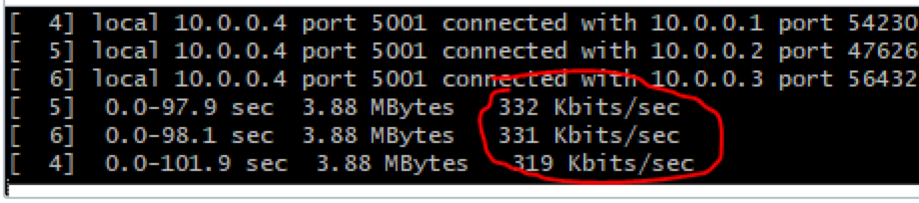
```
iperf -s
```

and on node-1, node-2, and node-3, simultaneously run `iperf` transmitters to send traffic to node-4:

```
iperf -c 10.0.0.4 -t 90
```

What network capacity is "seen" by each of the three transmitters? Upload a screenshot showing the `iperf` receiver output. Annotate it to draw a box or a circle around the data rate that was observed (this should be a single value, given in kilobits per second).

▾ iperf-with-MAC-one-node.PNG   Download



```
[  4] local 10.0.0.4 port 5001 connected with 10.0.0.1 port 54230
[  5] local 10.0.0.4 port 5001 connected with 10.0.0.2 port 47626
[  6] local 10.0.0.4 port 5001 connected with 10.0.0.3 port 56432
[  5]  0.0-97.9 sec  3.88 MBytes   332 Kbits/sec
[  6]  0.0-98.1 sec  3.88 MBytes   331 Kbits/sec
[  4]  0.0-101.9 sec  3.88 MBytes   319 Kbits/sec
```

Explain how and why this scenario is different from your `iperf` output above, when MAC learning is also turned on.

As part of your answer, make sure to explain:

- When MAC learning is turned on in this additional experiment, are all frames flooded on all bridge ports, regardless of their destination? (This means that there is a single collision domain.) Or, are frames only forwarded on the bridge port that their destination is attached to? (That means that there are separate collision domains on each network segment attached to a separate bridge port.)
- What is the total volume of traffic delivered to node-4 in the first case (from your `iperf` output above when MAC learning is turned on)? Compute this as the sum of the data rates of all `iperf` flows *going to node-4*.
- What is the total volume of traffic delivered to node-4 in this additional experiment? Compute this as the sum of the data rates of all `iperf` flows going to node-4.

With MAC learning turned on
[4] 0.0-94.3 sec 10.8 MBytes 957 Kbits/sec
With MAC learning turned off
[4] 0.0-95.0 sec 4.88 MBytes 431 Kbits/sec
With MAC learning turned on
[5] 0.0-97.9 sec 3.88 MBytes 332 Kbits/sec
[6] 0.0-98.1 sec 3.88 MBytes 331 Kbits/sec

[4] 0.0-101.9 sec 3.88 MBytes 319 Kbits/sec

In the additional experiment, all frames only forwarded on the bridge port that their destination is attached to. In order to avoid the collision. The bridge uses separate collision domains, so each node evenly split the network capacity so each node get 330 Kbits/sec.

The total volume of traffic delivered to node-4 in the first case is 94.3 * 957 = 90245.1Kbits
the total volume of traffic delivered to node-4 in this additional experiment is
332*97.9+331*98.1+319*101.9 = 97480Kbits

Q1.4 Additional bridge questions

1 Point

What are the source and destination IP addresses and source and destination MAC addresses of a packet that went from node-1 to the bridge?

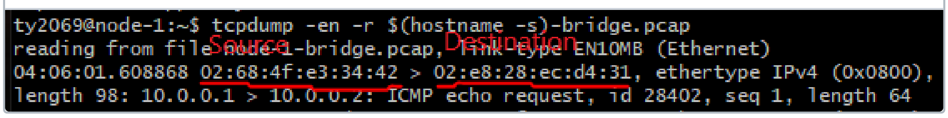
source: 02:68:4f:e3:34:42
dest: 02:e8:28:ec:d4:31

Does node-1 put the bridge's MAC address or node-2's MAC address in the destination address field of the Ethernet header?

Yes

Show an annotated screenshot from your `tcpdump` replay on node-1, with the source and destination IP addresses and source and destination MAC addresses clearly labeled. (Draw a box or circle around each of the four addresses in the selected packet, and label each as "source" or "destination".)

▾ tcpdump.png Download



What are source and destination IP addresses and source and destination MAC addresses of a packet that went from the bridge to node-2?

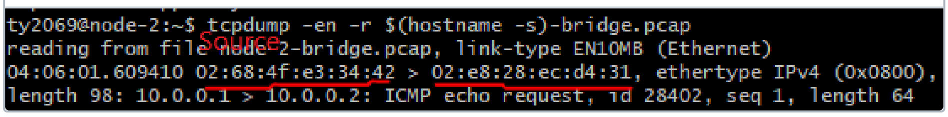
source: 02:68:4f:e3:34:42
dest: 02:e8:28:ec:d4:31

Does the bridge modify the source or destination MAC address in the Ethernet header?

No

Show an annotated screenshot *of the same packet* from your `tcpdump` replay on node-2, with the source and destination IP addresses and source and destination MAC addresses clearly labeled. (Draw a box or circle around each of the four addresses in the packet, and label each as "source" or "destination".)

▾ tcpdump2.png Download



Q2 Spanning tree

6 Points

Q2.1 Broadcast storm

1 Point

Why does a broadcast storm occur specifically when there is a loop in the network? Why does the loop in the network "amplify" the broadcast traffic, so that even when only a few broadcast packets are sent, the load on the network is very high?

Since the broadcast packets will send to all switches except the sender. Then when Romeo sent out a broadcast packet. All other hosts will receive the packet and then transfer to other hosts include Romeo. Then the loop will never stop and make a broadcast storm.

Why does this occur specifically with broadcast packets, not unicast packets?

Since unicast packets won't be transferred by other hosts to others. When the packet reaches its destination. A reply packet will send back. So no loop will occur.

Q2.2 Set up bridges to use spanning tree algorithm

3 Points

In this question, you will show how the bridges in the loop form a spanning tree.

You will need screenshots of the final output of `brctl showstp br0` from each bridge (after the spanning tree is complete). These screenshots should show the terminal prompt (important - it must show which bridge the output is from!), the command, and the complete output. You will also need the BPDUs collected on each network segment (open these in Wireshark).

Elect the root bridge

The first step in the spanning tree algorithm is electing a root bridge, the bridge with the lowest bridge ID.

Annotate each of your `brctl` screenshots by drawing a circle or a box around the bridge ID, and another circle or box around the ID of the "designated root". Also, make a special marking on the screenshot from the root bridge itself (i.e. the one where the bridge ID and designated root are the same).

▾ brctl_bridge1.PNG Download

ty2069@bridge-1: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.028cc1b78e00
 designated root        8000.023891955ae7
 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                  110.89
 flags


eth1 (1)
 port id                8001                    state                 forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer          19.18
 designated port        8001                    forward delay timer         0.00
 designated cost           0                    hold timer                  0.00
 flags

eth2 (2)
 port id                8002                    state                 forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer           0.00
 designated port        8002                    forward delay timer         0.00
 designated cost         100                    hold timer                  0.86
 flags
```

brctl_bridge2.PNG

Download

ty2069@bridge-2: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023891955ae7       →root
 designated root        8000.023891955ae7
 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               1.96                 tcn timer                   0.00
 topology change timer     0.00                 gc timer                   86.98
 flags


eth1 (1)
 port id                8001                    state                 forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer           0.00
 designated port        8001                    forward delay timer         0.00
 designated cost           0                    hold timer                  0.95
 flags

eth2 (2)
 port id                8002                    state                 forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer           0.00
 designated port        8002                    forward delay timer         0.00
 designated cost           0                    hold timer                  0.95
 flags
```

brctl_bridge3.PNG

Download

ty2069@bridge-3: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.02bf2338a439
 designated root        8000.023891955ae7
 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                  74.07
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer         18.20
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.00
 flags
```

▾ brctl_bridge4.PNG Download

ty2069@bridge-4: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023b0d234c7c
 designated root        8000.023891955ae7
 root port                 1                    path cost                  200
 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                  62.71
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer         19.06
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                  blocking
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer         19.06
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.00
 flags
```

Is the root bridge bridge-1, bridge-2, bridge-3, or bridge-4?

bridge-2

To elect the root bridge, each bridge initially considers *itself* the root bridge. Then, bridges exchange spanning tree configuration

BPDUs, including a "root identifier" field where they list the ID of the bridge they consider to be the root bridge.

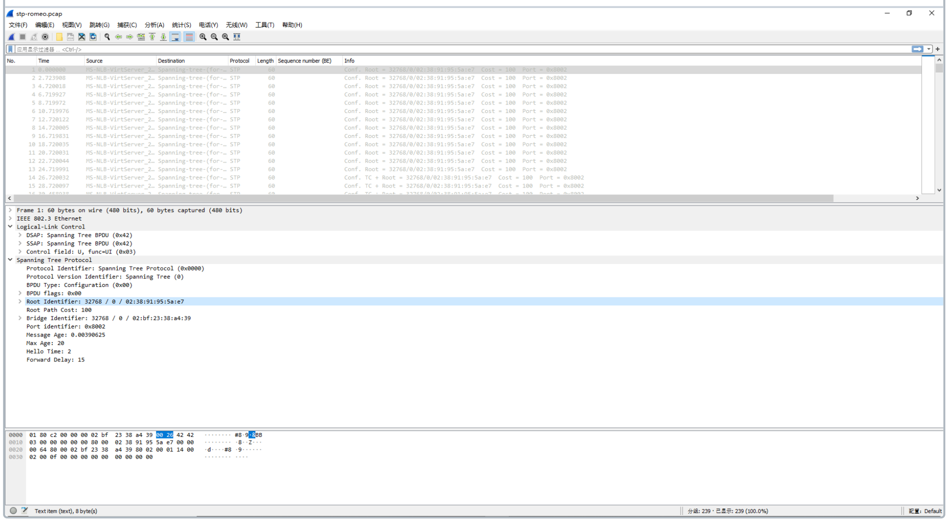
From among the BPDUs you collected, find one where the root bridge in the "root identifier" field is *not* the same as the final root bridge you identified above. (You're likely to find this near the beginning of a packet capture.)

Note: in Wireshark, the first part of the bridge ID, which is a priority value, is shown in decimal digits in the Packet Details pane, while in the `brctl` output, it is in hex digits. You can see the ID in hex in Wireshark by highlighting the field in the Packet Details pane and then looking at the Packet Bytes pane.

Upload a screenshot showing the Packet Details pane and Packet Bytes pane in Wireshark, with the Root Identifier in this BPDU highlighted. Also note which bridge this BPDU was sent from (use the Bridge ID field in this BPDU to identify the source). Was it sent from bridge-1, bridge-2, bridge-3, or bridge-4?

bridge-3

▾ wireshark.PNG Download



Upon receiving BPDUs from neighboring bridges, a bridge may see that its neighbor has a different root bridge than itself. If its neighbor's root bridge ID is smaller than the ID of the bridge it currently considers the root, it will update its root bridge to the new, smaller value.

Compare the root bridge ID in the BPDU you showed above, and the ID of the root bridge elected by all the bridges eventually. Give these two values in hex. Which one is greater? Why does the bridge whose BPDU you showed above eventually change its root bridge?

> by comparing 02bf2338a439 and 023891955ae7, the second number is smaller so the bridge updates its root bridge.

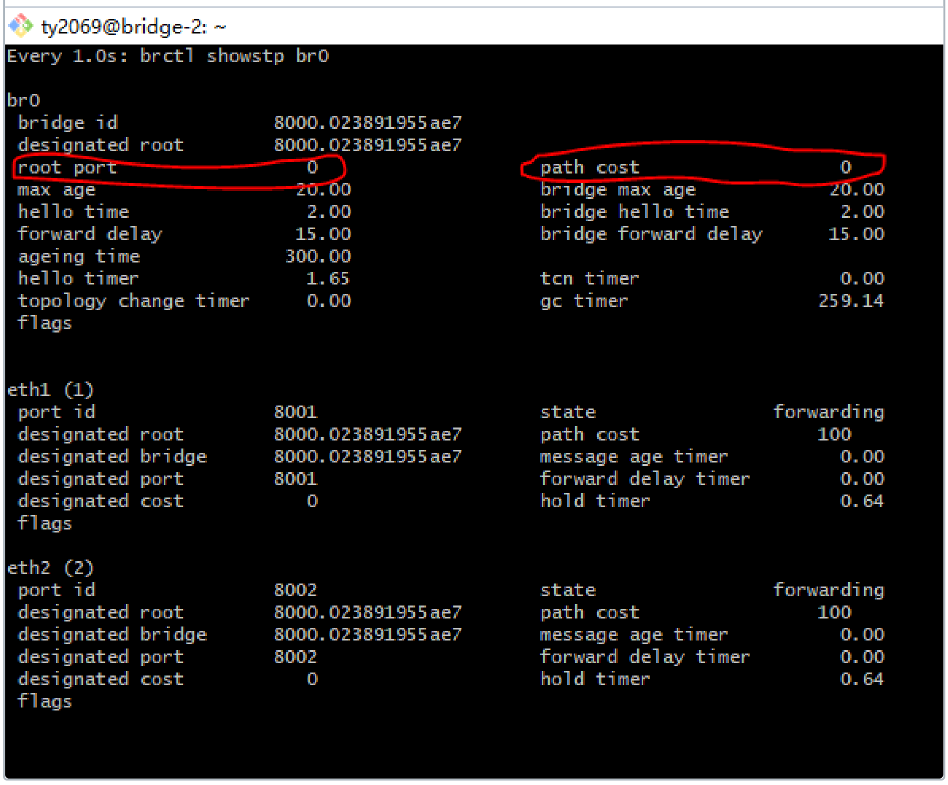
Elect a root port on each non-root bridge

In the second step, each bridge (except the root bridge) computes the root path cost, i.e. cost of the path to the root bridge, through each port. Then, the root port is elected - the one with the lowest root path cost. This is the port that is "facing" the root bridge.

Find the `brctl` screenshots from your root bridge, and annotate it by drawing a circle or a box around the *root port*, which is listed near the top of the output. For the root bridge, the root port will be 0 (there is no root port). Also draw a circle or a box around the *path cost* for the bridge overall, which is shown right next to the root port, near the top of the output.

Upload this annotated screenshot here:

▾ root-bridge-root-port.PNG Download



```
ty2069@bridge-2: ~
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023891955ae7
 designated root        8000.023891955ae7
 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               1.65                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                 259.14
 flags


eth1 (1)
 port id                8001                    state               forwarding
 designated root        8000.023891955ae7       path cost                100
 designated bridge      8000.023891955ae7       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.64
 flags

eth2 (2)
 port id                8002                    state               forwarding
 designated root        8000.023891955ae7       path cost                100
 designated bridge      8000.023891955ae7       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.64
 flags
```

Next, find the `brctl` screenshots from the two bridges that have one port on the same network segment as the root bridge. This is the port that will be selected as the root port. Annotate these two screenshots by drawing a circle or a box around the *root port*, which is listed near the top of the output. Also draw a circle or a box around the *path cost* for the bridge overall, which is shown right next to the root port, near the top of the output.

Upload these annotated screenshots here:

same-seg-with-root1.PNG Download

ty2069@bridge-1: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.028cc1b78e00
 designated root        8000.023891955ae7
 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                 141.16
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer        20.01
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 1.70
 flags
```

same-seg-with-root2.PNG Download

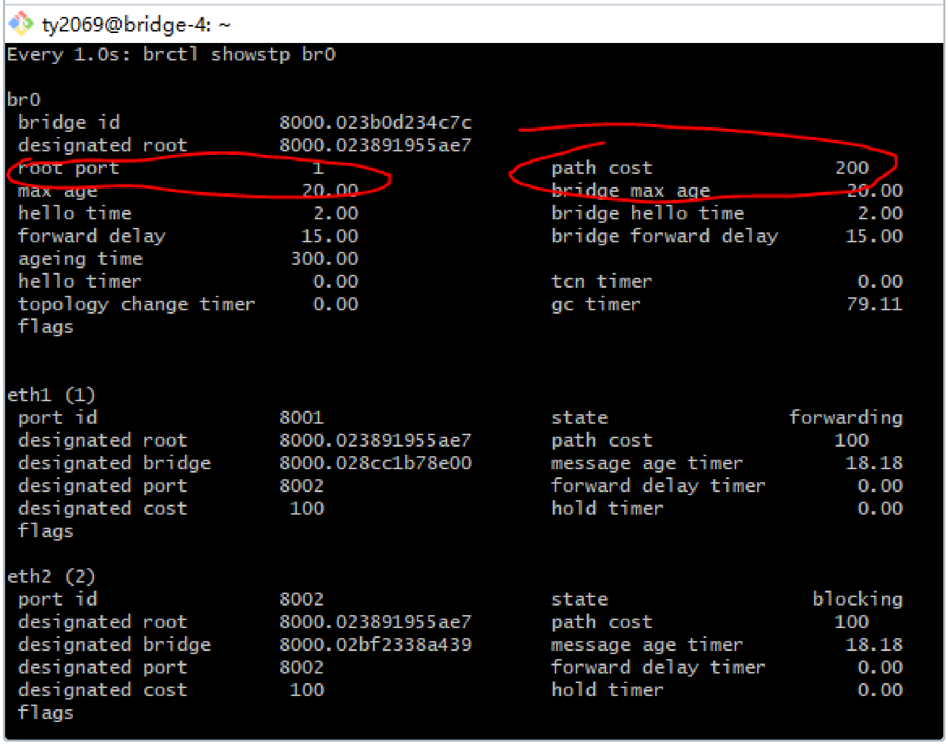
ty2069@bridge-3: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.02bf2338a439
 designated root        8000.023891955ae7
 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                 115.88
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer        18.59
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.31
 flags
```

Then, find the `brctl` screenshots from the bridge that is *farthest* from the root bridge. Annotate these two screenshots by drawing a circle or a box around the *root port*, which is listed near the top of the output. Also draw a circle or a box around the *path cost* for the bridge overall, which is shown right next to the root port, near the top of the output. Then, draw a circle or box around the *state* of the root port.

Upload this annotated screenshot here:

▾ farest-from-root.PNG Download



```
ty2069@bridge-4: ~
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023b0d234c7c
 designated root        8000.023891955ae7
 root port                 1                    path cost                  200
 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                  79.11
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer        18.18
 designated port        8002                    forward delay timer       0.00
 designated cost         100                    hold timer                0.00
 flags

eth2 (2)
 port id                8002                    state                  blocking
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer        18.18
 designated port        8002                    forward delay timer       0.00
 designated cost         100                    hold timer                0.00
 flags
```

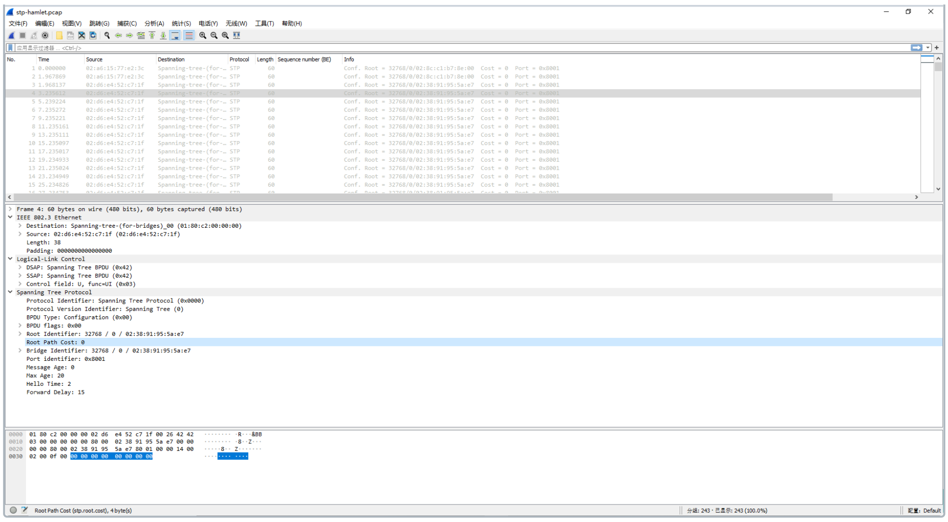
The bridges find out the path cost (to elect their root ports) by exchanging BPDUs.

From among the BPDUs you collected, find one where the root path cost is zero. Upload a screenshot showing the Packet Details pane and Packet Bytes pane in Wireshark, with the Root Path Cost in this BPDU highlighted.

Also make a note of the bridge ID and root ID in this BPDU - is this BPDU sent by the root bridge, or a non-root bridge?

root bridge

▾ wireshark-cost0.PNG Download

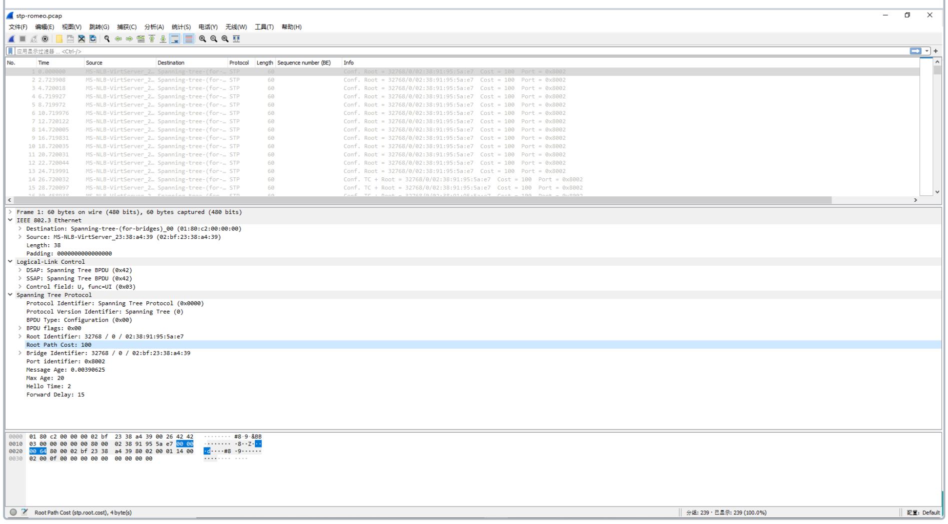


From among the BPDUs you collected, find one where the root path cost is *greater than* zero. Upload a screenshot showing the Packet Details pane and Packet Bytes pane in Wireshark, with the Root Path Cost in this BPDU highlighted.

Also make a note of the bridge ID and root ID in this BPDU - is this BPDU sent by the root bridge, or a non-root bridge?

non-root bridge

wireshark-cost100.PNG Download



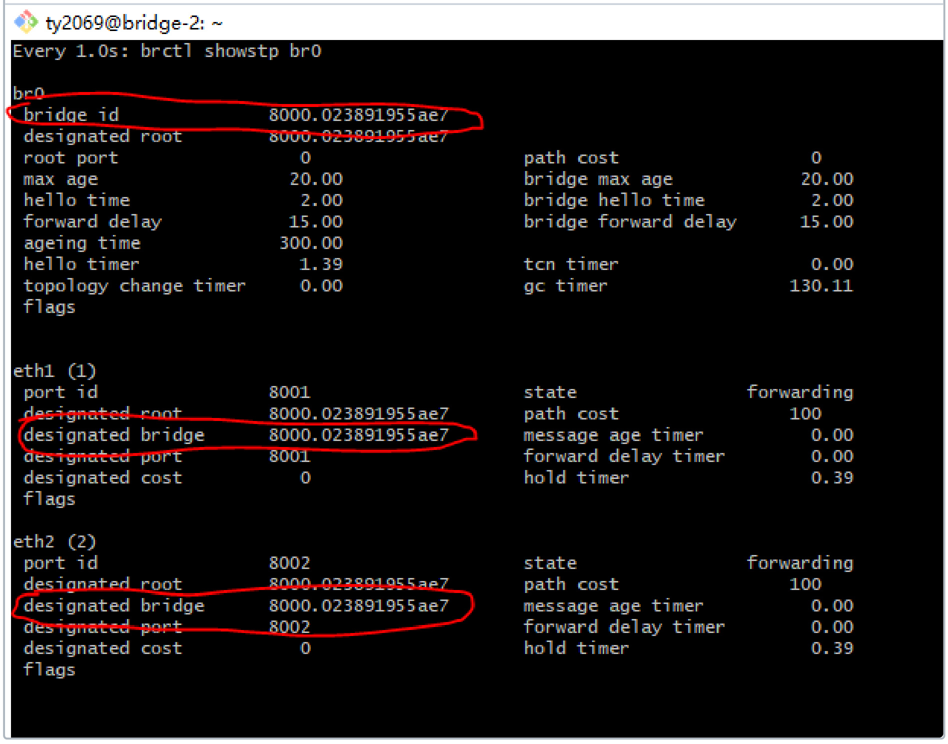
Select a designated bridge and port on each network segment

In the third step of the spanning tree protocol, the bridge on each network segment with the lowest root path cost will be selected as the designated bridge, and the port that connects that bridge to the network segment is the designated port.

Find the `brctl` screenshots from your root bridge, and annotate it by drawing a circle or a box around the *designated bridge* for each bridge port. Also draw a circle or a box around the bridge ID of this bridge (near the top of the output). The root bridge will be the designated bridge on any network segment it is on.

Upload this annotated screenshot here:

▾ root-designated.PNG Download



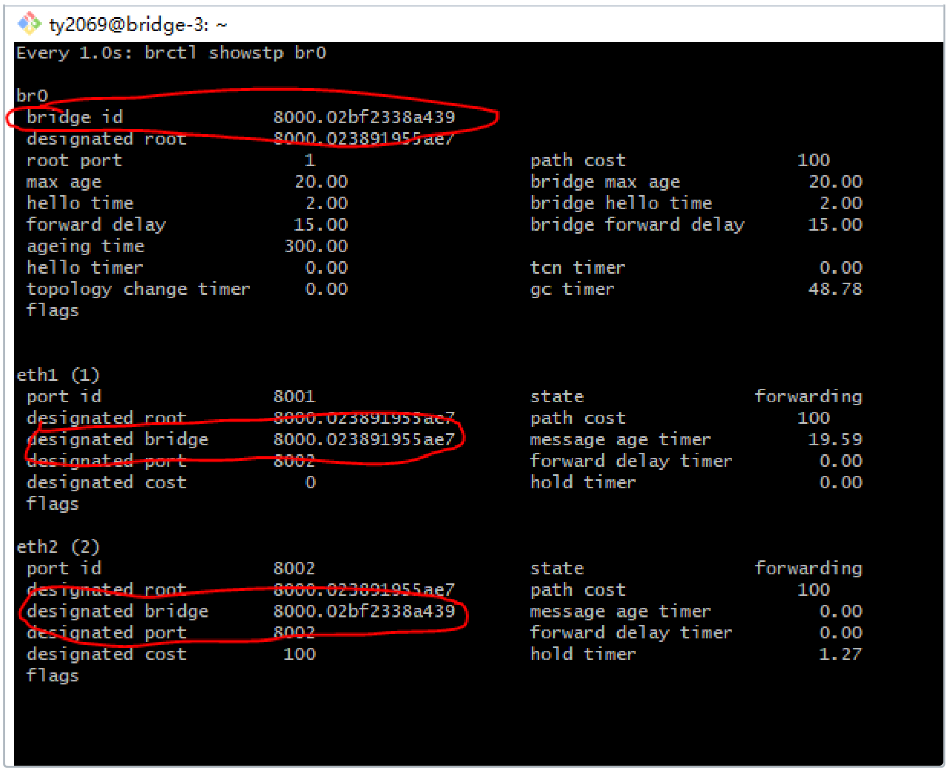
```
ty2069@bridge-2: ~
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023891955ae7
 designated root        8000.023891955ae7
 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               1.39                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                 130.11
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                100
 designated bridge      8000.023891955ae7       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.39
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023891955ae7       path cost                100
 designated bridge      8000.023891955ae7       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.39
 flags
```

Next, find the `brctl` screenshots from the two bridges that have one port on the same network segment as the root bridge. Draw a circle or a box around the bridge ID of this bridge (near the top of the output). Then draw a circle or a box around the *designated bridge* for each bridge port, and if this bridge is the designated bridge on the network segment (i.e. the designated bridge is the same as the bridge ID), also draw a circle or a box around the *designated port*.

For the bridge port on the same network segment as the root bridge, you should see that the root bridge is the designated bridge, since it has the lowest path cost. For the other port, this bridge and port should be the designated bridge and port, since it has a lower path cost than the other bridge connected to this network segment.

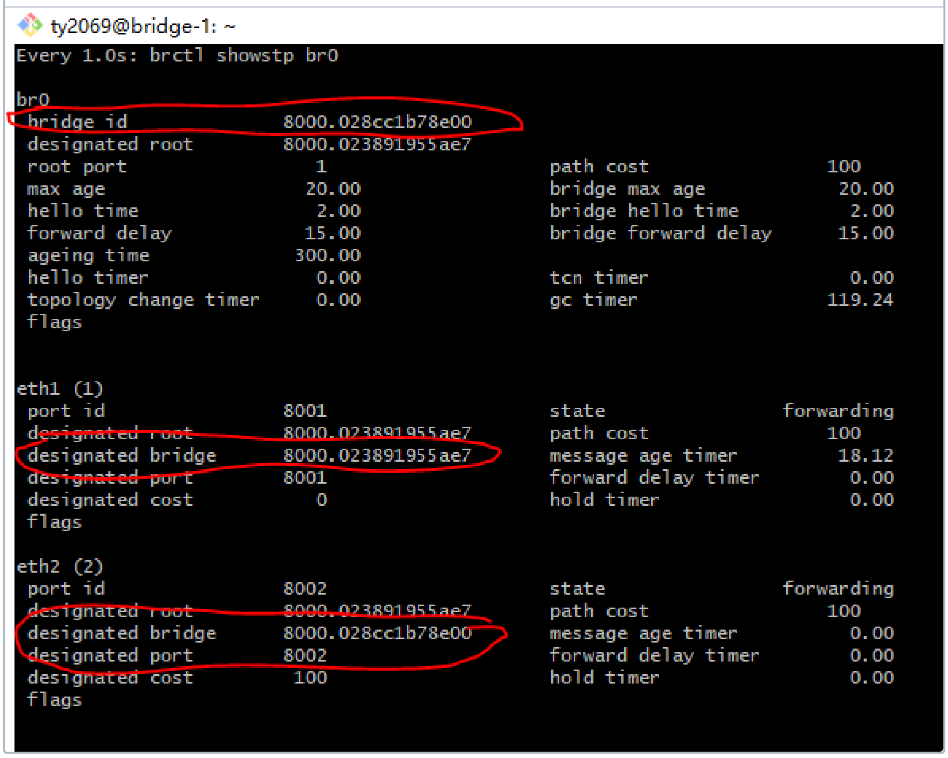
Upload these annotated screenshots here:

▾ bridge1-designated (1).PNG Download



```
ty2069@bridge-3: ~
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.02bf2338a439
 designated root        8000.023891955ae7
 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                    48.78
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer           19.59
 designated port        8002                    forward delay timer          0.00
 designated cost           0                    hold timer                   0.00
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer            0.00
 designated port        8002                    forward delay timer          0.00
 designated cost         100                    hold timer                   1.27
 flags
```

bridge3-designated (2).PNG Download



```
ty2069@bridge-1: ~
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.028cc1b78e00
 designated root        8000.023891955ae7
 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                   119.24
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer           18.12
 designated port        8001                    forward delay timer          0.00
 designated cost           0                    hold timer                   0.00
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer            0.00
 designated port        8002                    forward delay timer          0.00
 designated cost         100                    hold timer                   0.00
 flags
```

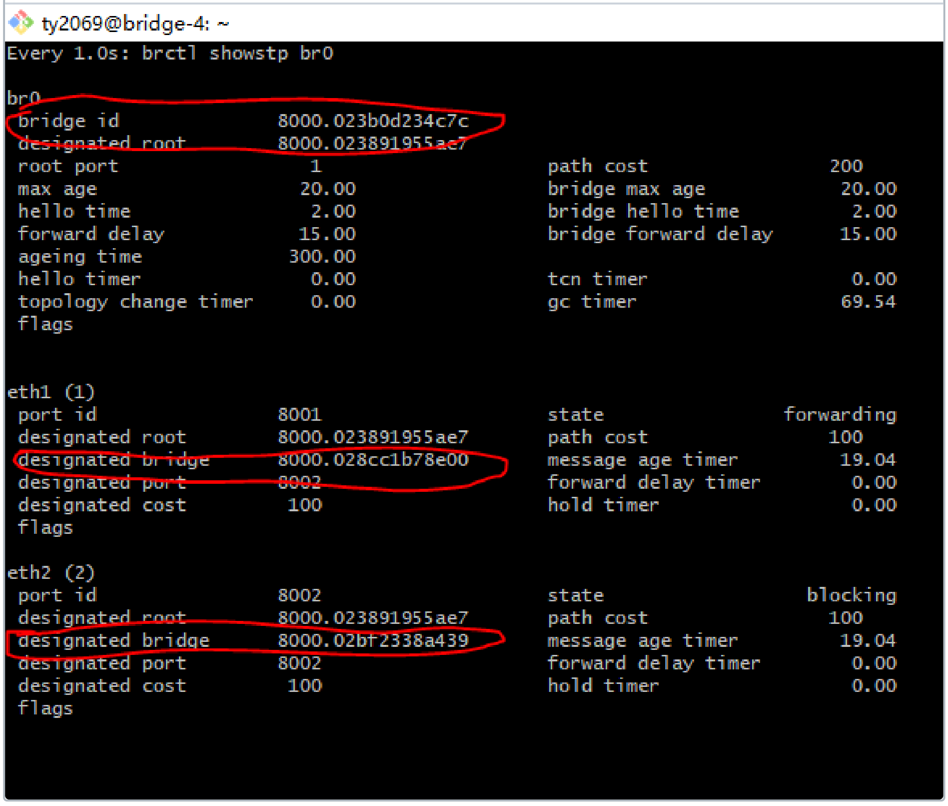
Then, find the `brctl` screenshots from the bridge that is *farthest* from the root bridge. Annotate it by drawing a circle or a box around the *designated bridge* for each bridge port. Also draw a circle or a box around the bridge ID of this bridge (near the top of the output).

Is this bridge a designated bridge on any network segment? Answer yes or no.

No

Upload the annotated screenshot here:

▾ bridge4-designated.PNG Download



```
ty2069@bridge-4: ~
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023b0d234c7c
 designated root        8000.023891955ae7
 root port                 1                    path cost                  200
 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                  69.54
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer         19.04
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                  blocking
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer         19.04
 designated port        8002                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.00
 flags
```

Set bridge ports' states

In the fourth step, on a bridge that is not the root, any bridge port that is neither a root port nor a designated port will be put in the blocked state.

Find the `brctl` screenshots from the two bridges that have one port on the same network segment on the root bridge. Annotate these by drawing a circle or a box around the *status* of each port. Next to each port, indicate whether it is a root port, a designated port on a network segment where this bridge is the designated bridge, or in the blocking state.

▾ bridge1-state.png Download

ty2069@bridge-1: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.028cc1b78e00
 designated root        8000.023891955ae7
 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                    40.04
 flags

eth1 (1)
 port id                8001                    state                  forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer           19.24
 designated port        8001                    forward delay timer          0.00
 designated cost           0                    hold timer                   0.00
 flags

eth2 (2)
 port id                8002                    state                  forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer            0.00
 designated port        8002                    forward delay timer          0.00
 designated cost         100                    hold timer                   0.96
 flags
```

root-port (eth1)

not root-port (eth2)

bridge3-state.png Download

ty2069@bridge-3: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.02bf2338a439
 designated root        8000.023891955ae7
 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                   120.86
 flags

eth1 (1)
 port id                8001                    state                  forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.023891955ae7       message age timer           19.04
 designated port        8002                    forward delay timer          0.00
 designated cost           0                    hold timer                   0.00
 flags

eth2 (2)
 port id                8002                    state                  forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer            0.00
 designated port        8002                    forward delay timer          0.00
 designated cost         100                    hold timer                   0.74
 flags
```

root-port (eth1)

not root-port (eth2)

Then, find the `brctl` screenshots from the bridge that is *farthest* from the root bridge. Annotate it by drawing a circle or a box around the *status* of each port. Next to each port, indicate whether it is a root port, a designated port, or in the blocking state.

bridge4-state.png Download

ty2069@bridge-4: ~

```
Every 1.0s: brctl showstp br0

br0
 bridge id              8000.023b0d234c7c
 designated root        8000.023891955ae7
 root port                 1                    path cost                  200
 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                    40.43
 flags


eth1 (1)
 port id                8001                    state                  forwarding
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.028cc1b78e00       message age timer           19.69
 designated port        8002                    forward delay timer          0.00
 designated cost         100                    hold timer                   0.00
 flags

eth2 (2)
 port id                8002                    state                    blocking
 designated root        8000.023891955ae7       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer           19.69
 designated port        8002                    forward delay timer          0.00
 designated cost         100                    hold timer                   0.00
 flags
```

root-port

in blocking state

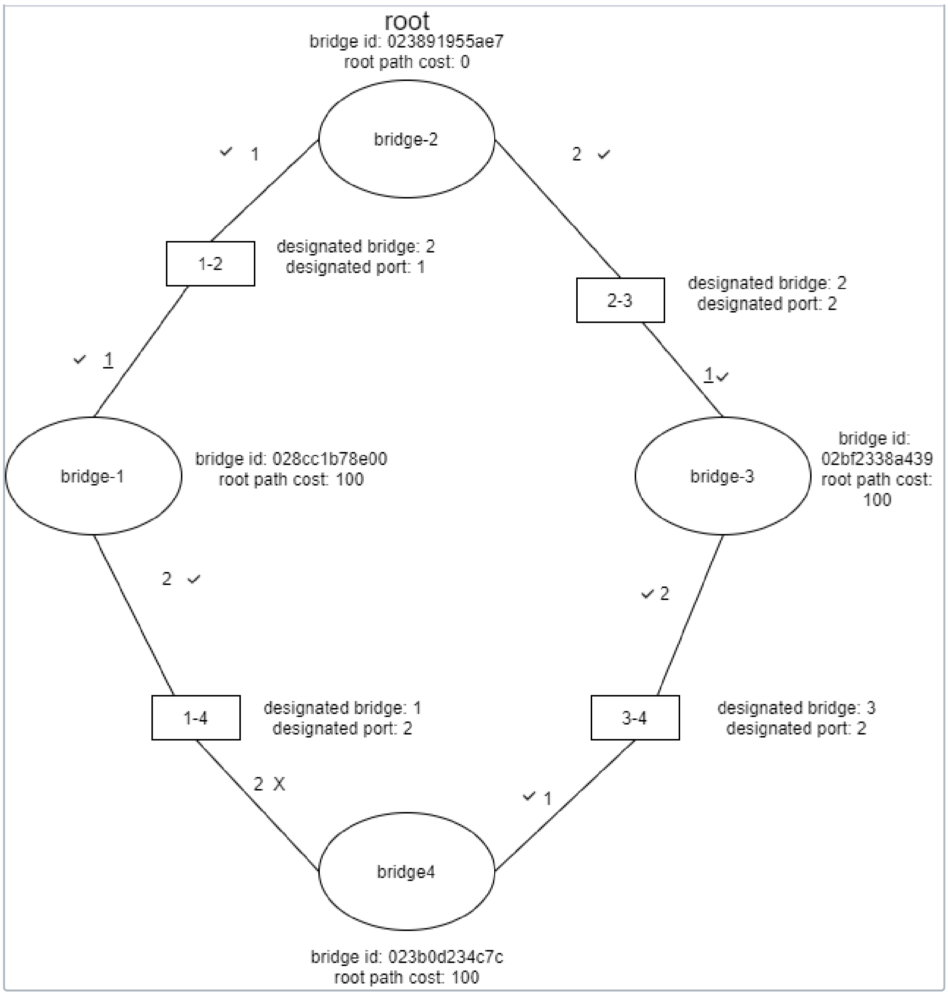
Draw the spanning tree

Finally, draw the spanning tree from this section:

- Put the root bridge at the top of your drawing. Draw a circle around the root bridge, and label it "Root". Then, draw each of the other bridges. On each bridge, write its hostname (e.g. "bridge-1", "bridge-2", etc.) Draw links connecting the bridges; label each network segment (e.g. "1-2", "2-3", etc.)
- Label each bridge with its bridge ID, and each port with its port ID (1 or 2).
- If a port is the root port for that bridge, underline its port ID.
- Next to each bridge port, draw a check mark if it is in the forwarding state. If a port is in the blocked state, then draw an X next to it.
- Next to each network segment (1-2, 2-3, 3-4, 1-4), write the designated bridge and the designated port on that bridge (1 or 2) for that network segment.
- Next to each bridge, write the root path cost for that bridge.

Upload your spanning tree diagram as a PNG, PDF, or JPG file.

span-tree.png Download



Q2.3 Reacting to changes in the topology

1 Point

Show the output of the `brctl showstp br0` command on each bridge at the end of the section where you practiced "Reacting to changes in the topology". These screenshots should show the terminal prompt (important - it must show which bridge the output is from!), the command, and the complete output.

▾ changed-bridge1.PNG Download

ty2069@bridge-1: ~

```
ty2069@bridge-1:~$ brctl showstp br0
br0
 bridge id              8000.028cc1b78e00
 designated root        8000.023b0d234c7c
 root port                 2                    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                 151.85
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023b0d234c7c       path cost                100
 designated bridge      8000.028cc1b78e00       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.32
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023b0d234c7c       path cost                100
 designated bridge      8000.023b0d234c7c       message age timer         19.11
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags
```

changed-bridge2.PNG

Download

ty2069@bridge-2: ~

```
ty2069@bridge-2:~$ brctl showstp br0
br0
 bridge id              8000.023891955ae7
 designated root        8000.023891955ae7
 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               0.00                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                   0.00
 flags


eth1 (1)
 port id                8001                    state                  disabled
 designated root        8000.023891955ae7       path cost                100
 designated bridge      8000.023891955ae7       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                  disabled
 designated root        8000.023891955ae7       path cost                100
 designated bridge      8000.023891955ae7       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags
```

changed-bridge3.PNG

Download

ty2069@bridge-3: ~

```
ty2069@bridge-3:~$ brctl showstp br0
br0
 bridge id              8000.02bf2338a439
 designated root        8000.023b0d234c7c
 root port                 2                    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                 143.27
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023b0d234c7c       path cost                  100
 designated bridge      8000.02bf2338a439       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost         100                    hold timer                 0.00
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023b0d234c7c       path cost                  100
 designated bridge      8000.023b0d234c7c       message age timer         18.46
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags
```

▾ changed-bridge4.PNG Download

ty2069@bridge-4: ~

```
ty2069@bridge-4:~$ brctl showstp br0
br0
 bridge id              8000.023b0d234c7c
 designated root        8000.023b0d234c7c
 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               1.50                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                 143.14
 flags


eth1 (1)
 port id                8001                    state                forwarding
 designated root        8000.023b0d234c7c       path cost                  100
 designated bridge      8000.023b0d234c7c       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.49
 flags

eth2 (2)
 port id                8002                    state                forwarding
 designated root        8000.023b0d234c7c       path cost                  100
 designated bridge      8000.023b0d234c7c       message age timer          0.00
 designated port        8002                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.48
 flags
```

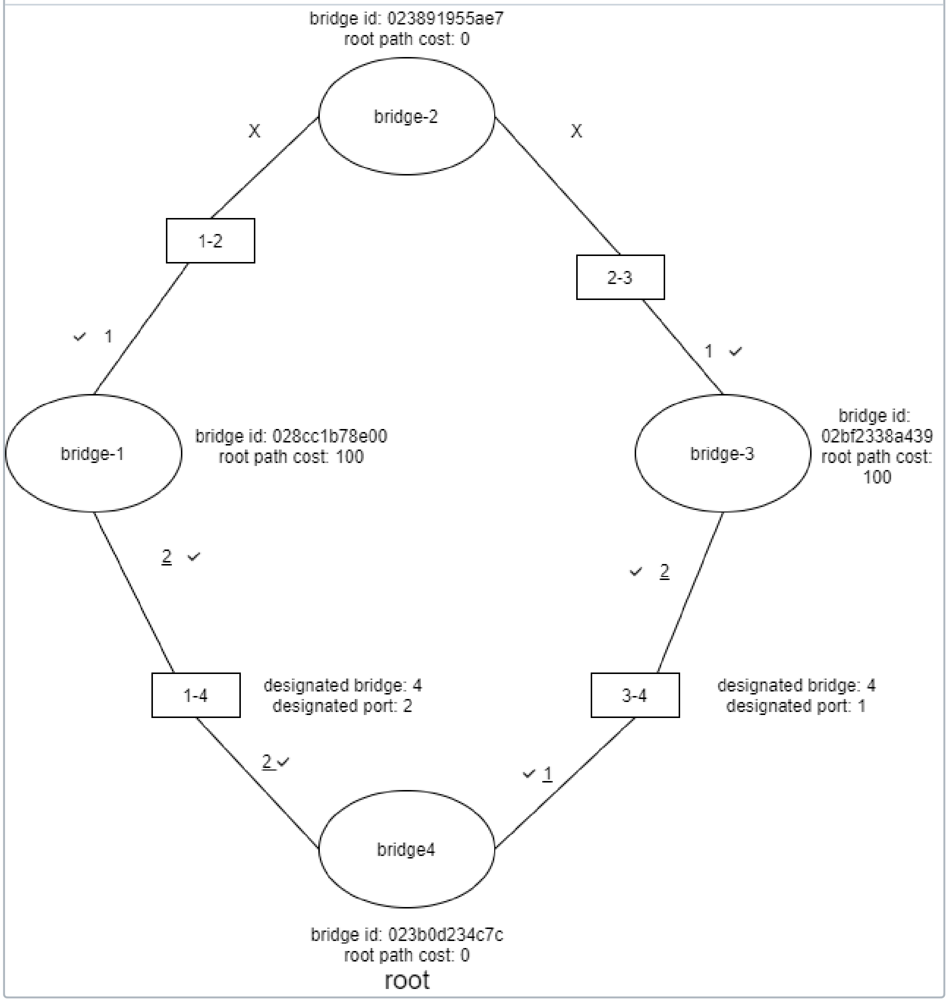
Also draw the network from the section where you practiced "Reacting to changes in the topology" (i.e. the new spanning tree after you brought down the root bridge), following the same specifications:

- Put the root bridge at the top of your drawing. Draw a circle around the root bridge, and label it "Root". Then, draw each of the other bridges. On each bridge, write its hostname (e.g. "bridge-1", "bridge-2", etc.) Draw links connecting the bridges; label each network segment (e.g. "1-2", "2-3", etc.)

- Label each bridge with its bridge ID, and each port with its port ID (1 or 2).
- If a port is the root port for that bridge, underline its port ID.
- Next to each bridge port, draw a check mark if it is in the forwarding state. If a port is in the blocked state, then draw an X next to it.
- Next to each network segment (1-2, 2-3, 3-4, 1-4), write the designated bridge and the designated port on that bridge (1 or 2) for that network segment.
- Next to each bridge, write the root path cost for that bridge.

Upload your spanning tree diagram as a PNG, PDF, or JPG file.

▾ changed-span-tree.png Download



Q2.4 Change in topology

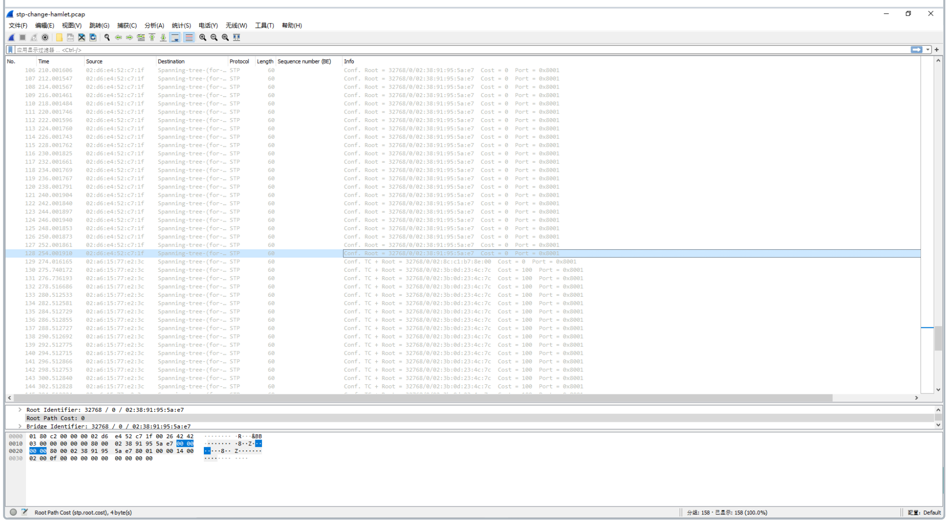
1 Point

When you changed the topology, how much time elapsed between the last ping request arriving at the target before you brought the root bridge down, and the first ping request arriving at the target after you brought the root bridge down? (Use the packet capture from the network segment on which the target node was located.)

20 sec

Show evidence from your packet captures to support your answer. For example, show a Wireshark screenshot, and annotate it to circle the latest time that a ping request arrives at the target before you brought the root bridge down, and the earliest time that a ping request arrives at the target after you brought the root bridge down.

wireshark-change.PNG Download



Q3 Delete your resources, please

0 Points

Did you delete your resources in the GENI Portal? After you have finished submitting your answers to the questions above, delete your resources so that they will be available to other experimenters.

- [x] Yes, I deleted my resources.

Lab 3: Bridges and LANs

UNGRADED

STUDENT

Tingyu Yang

TOTAL POINTS

- / 10 pts

QUESTION 1

Operation of a basic Ethernet switch or bridge (and additional bridge questions) 4 pts

1.1	Setting up the bridge	1 pt
1.2	Learning MAC addresses	1 pt
1.3	Effect of a smaller collision domain	1 pt
1.4	Additional bridge questions	1 pt

QUESTION 2

Spanning tree 6 pts

2.1	Broadcast storm	1 pt
2.2	Set up bridges to use spanning tree algorithm	3 pts
2.3	Reacting to changes in the topology	1 pt
2.4	Change in topology	1 pt

QUESTION 3

Delete your resources, please 0 pts