

IT Technology
Networking Assignment 4



.....

University College

Authors

Aleksis Kairiss
Alexander Bjørk Andersen
Dainty Lyka Bihay Olsen
Daniel Rasmussen
Jacob Suurballe Petersen
Nikolaj Hult-Christensen
Sheriff Dibba

alka28793@edu.ucl.dk
aban28821@edu.ucl.dk
dlbo28887@edu.ucl.dk
dara28918@edu.ucl.dk
jspe28926@edu.ucl.dk
nihu28952@edu.ucl.dk
shdi28860@edu.ucl.dk

Wednesday 28 October 2020

Table of Contents

Introduction	1
Audience	1
Inventory	1
Topology diagram	2
brctl and configuration	2
Commands	8
Sudo brctl showmacs br0	8
brctl show	9
Sudo brctl stp br0 off/on	9
IP addresses and Subnet masks	10
TCPdumps from one computer to another	11
Commands	12
tcpdump -D	12
Tcpdump -i <interface name>	12
Tcpdump -n -i <interface name>	13
Show ARP tables	13
MAC table of switch	14
Conclusion	14

1 Introduction

This report will demonstrate the students' capability to set up a switch and two virtual machines inside VMware workstation. The students will also describe a layer 2 switch, a host MAC table and what the mechanics of ARP are.

The student will also demonstrate setting up static IP addresses on vm machines, show connectivity between machines and the result of a TCP dump traffic monitoring.

2 Audience

This document is meant for teachers and fellow students alike, with the intention of receiving peer review from these parties.

3 Inventory

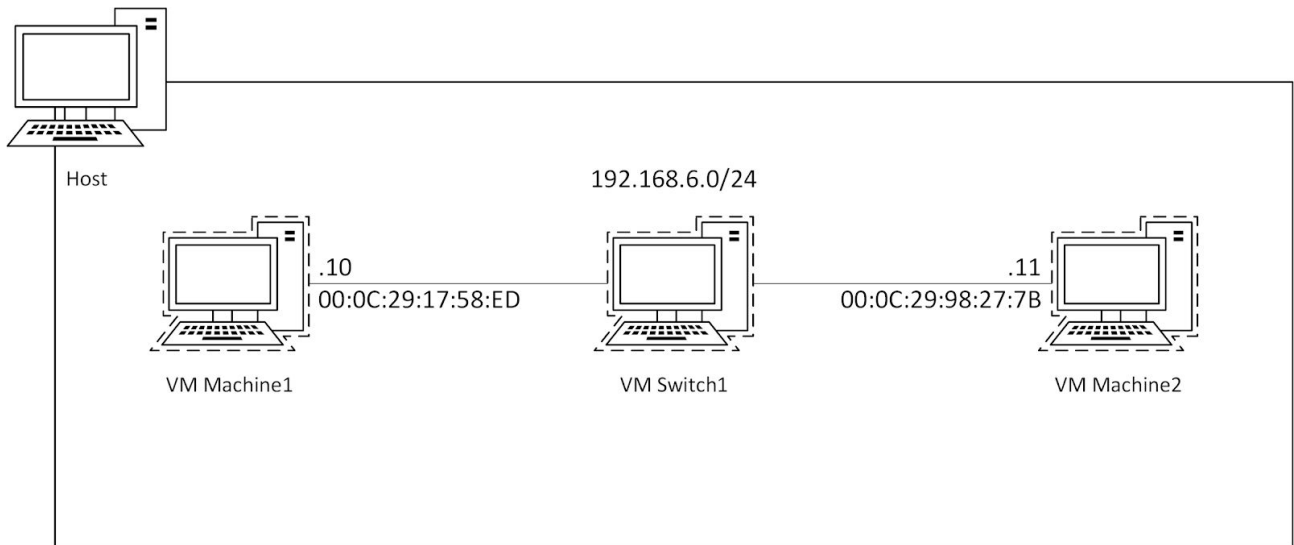
Software:

- VMWare Workstation Pro
 - Xubuntu
 - Brctl
 - Ifupdown
 - TCPdump
 - Terminal
- Visio

Hardware:

- Host machine

4 Topology diagram



In the above diagram we see 4 computers (1 physical host, 2 virtual machines and a virtual switch, hosted on the physical computer).

The virtual switch is hosted on a network id 192.168.6.0, the two virtual machines are connected to each other through the vm switch. The virtual switch is on subnet mask 255.255.255.0 (24 cidr notation).

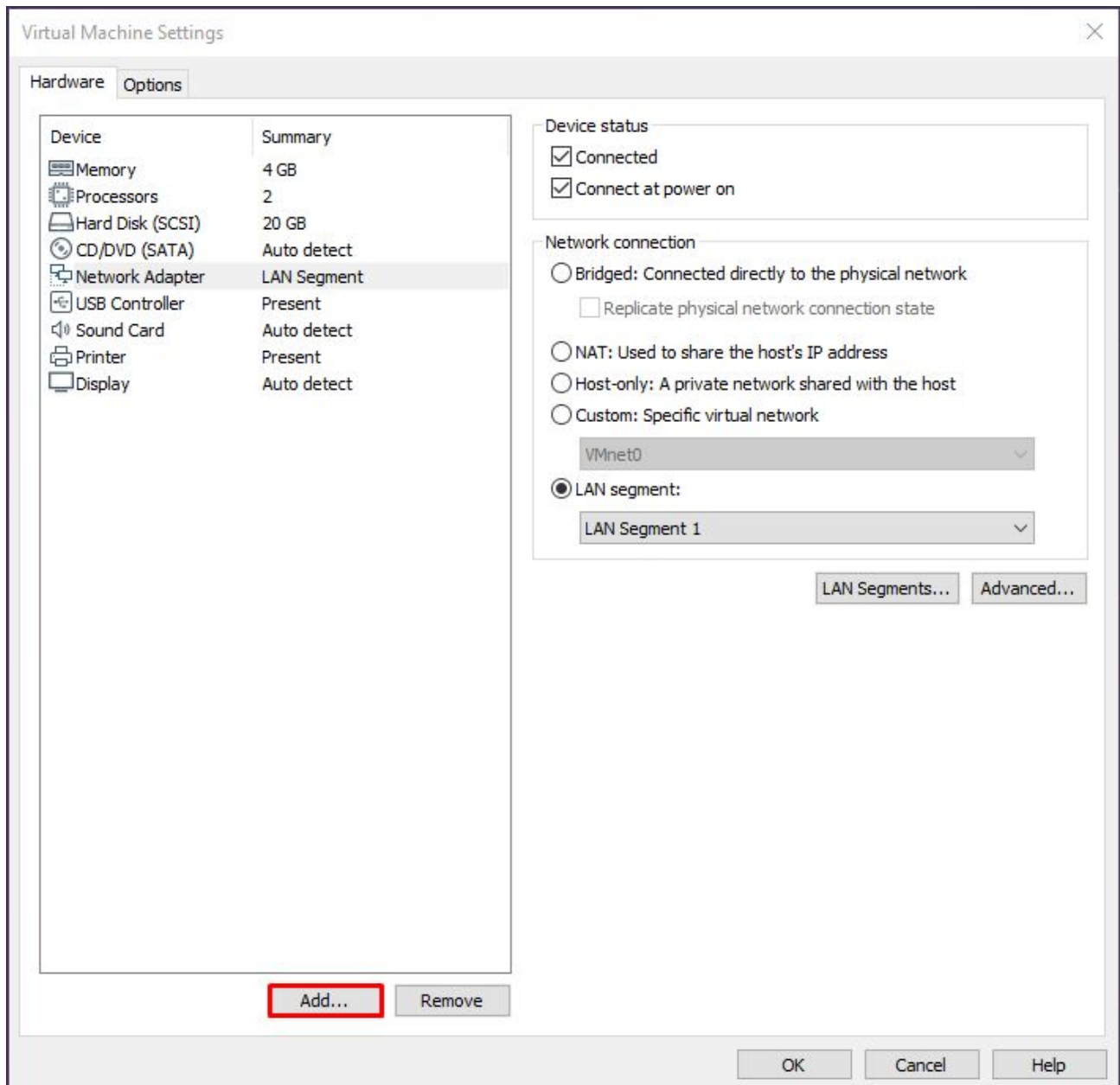
Each virtual machine has been assigned their own ip addresses, and MAC addresses. Vm machine1 has been given the ip address 192.168.6.10 and the MAC address 00:0C:29:17:58:ED. Vm machine2 has been given the ip address 192.168.6.11 and the MAC address 00:0C:29:98:27:7B.

5 brctl and configuration

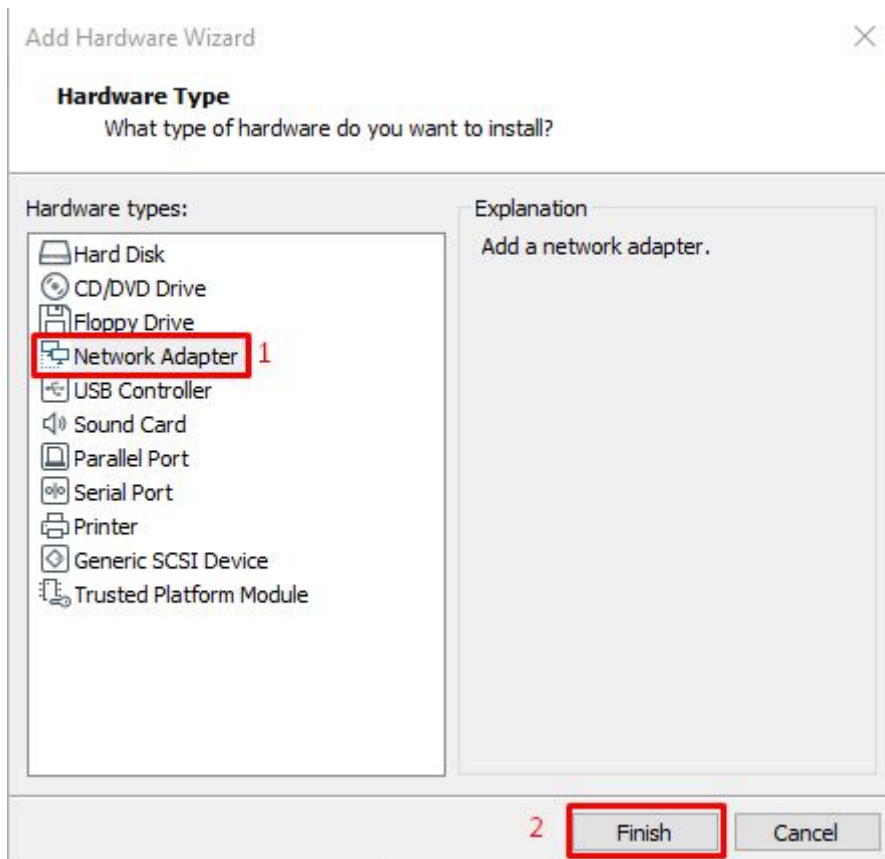
```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help
strindoi@ubuntu:~$ sudo brctl showmacs br0
port no mac addr is local? ageing timer
1 00:0c:29:17:58:ed no 161.38
1 00:0c:29:c0:83:7e yes 0.00
1 00:0c:29:c0:83:7e yes 0.00
2 00:0c:29:c0:83:88 yes 0.00
2 00:0c:29:c0:83:88 yes 0.00
strindoi@ubuntu:~$
```

brctl is a program used to bridge interfaces, in the picture above you can see two interfaces (port 1 and port 2) being bridged. In the example above, this is done on bridge 0 (henceforth referred to as br0).

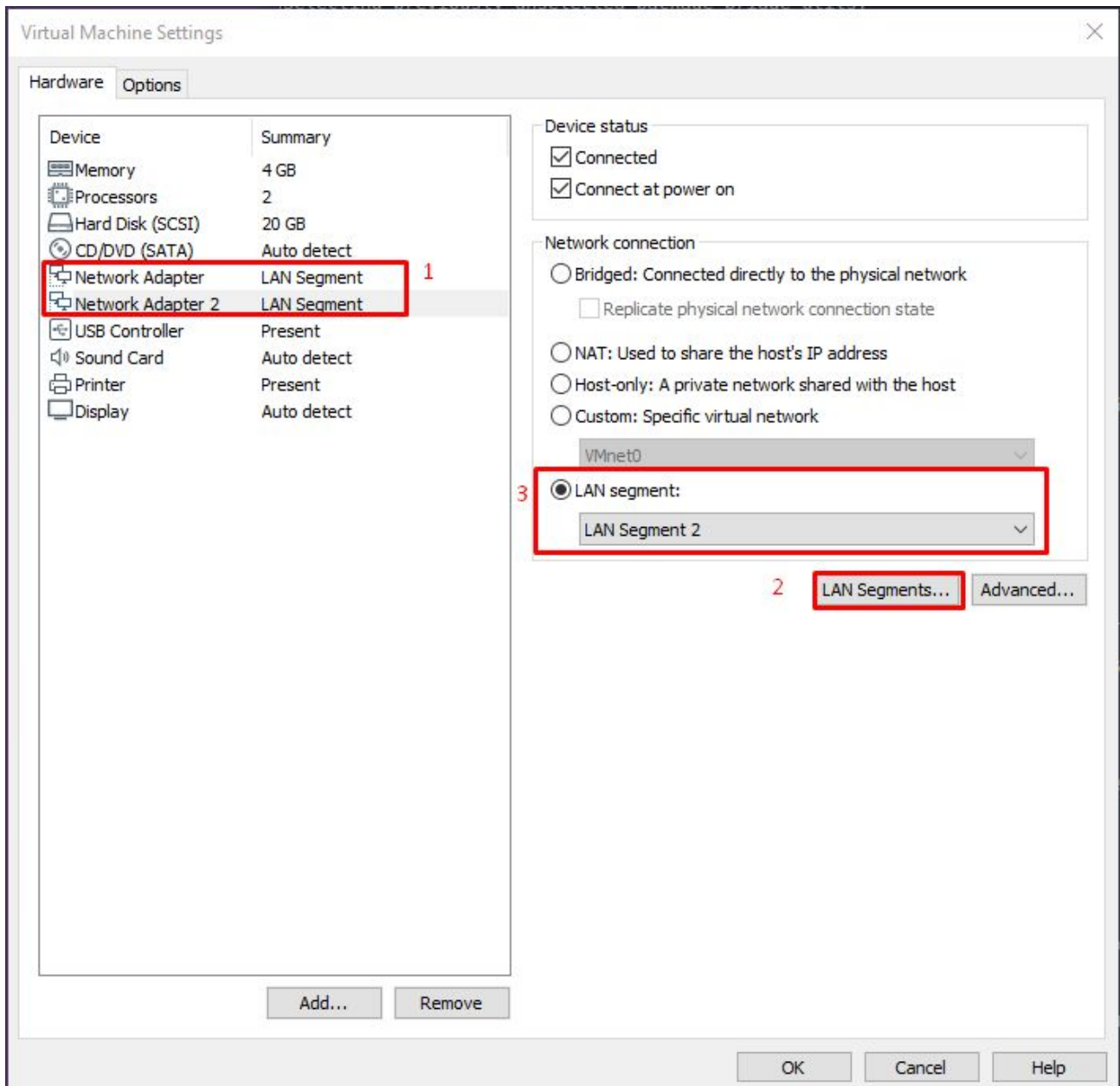
To configure the switch for switching/bridging you first need to add multiple network adapters to your virtual switch. When using VMware workstation pro, this is done using the virtual machine settings, as seen in the pictures below.



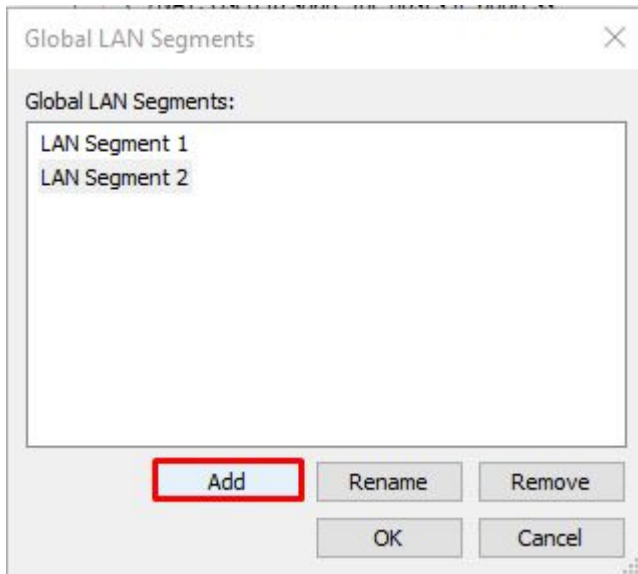
First you press add to bring up the add hardware wizard.



Then you press “network adapter” to add more network adapters. Repeat this step until you have the desired amount of network adapters.



The next step is to configure the LAN Segments, this is done by going to the Global LAN Segments window and adding LAN Segments (see the picture below), and then setting each network adaptor's network connection to each of their own LAN segments.



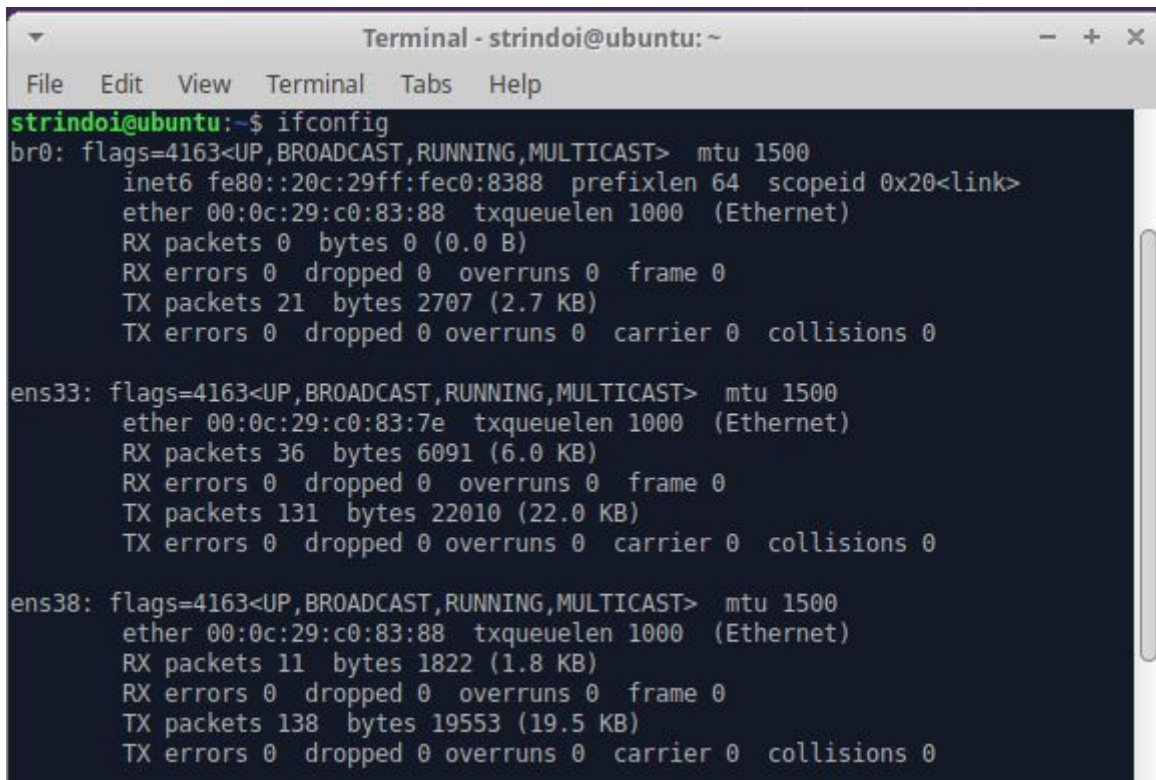
Add a LAN Segment for each network adaptor you have added to the virtual switch.

When this is done, you can turn on the switch, or reboot it if it was already on.

Now we need to complete the setup in the linux environment on the virtual switch. To do this we first need to know what interfaces we want to add to our bridge.

We can find these by typing the following in the terminal, and looking for all of the ens interfaces.

```
ifconfig
```

A terminal window titled "Terminal - strindoi@ubuntu: ~" showing the output of the 'ifconfig' command. The output lists three network interfaces: br0, ens33, and ens38. Each interface shows its flags, MTU, IP address (for br0), MAC address, and statistics for RX and TX packets, bytes, errors, dropped, overruns, carrier, and collisions.

```
strindoi@ubuntu:~$ ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::20c:29ff:fec0:8388 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:c0:83:88 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 21 bytes 2707 (2.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

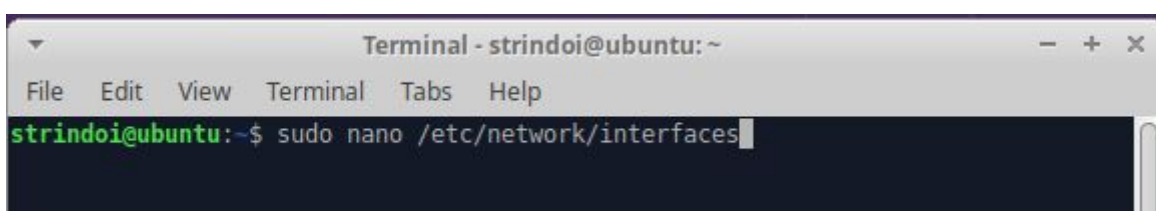
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 00:0c:29:c0:83:7e txqueuelen 1000 (Ethernet)
    RX packets 36 bytes 6091 (6.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 131 bytes 22010 (22.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 00:0c:29:c0:83:88 txqueuelen 1000 (Ethernet)
    RX packets 11 bytes 1822 (1.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 138 bytes 19553 (19.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

In this case, the interfaces we want to add to our bridge are ens33 and ens38.

To add ens33 and ens38 to our bridge, we need to nano into /etc/network/interfaces. We do this by writing the following in the terminal.

```
sudo nano /etc/network/interfaces
```

A terminal window titled "Terminal - strindoi@ubuntu: ~" showing the command 'sudo nano /etc/network/interfaces' being entered at the prompt.

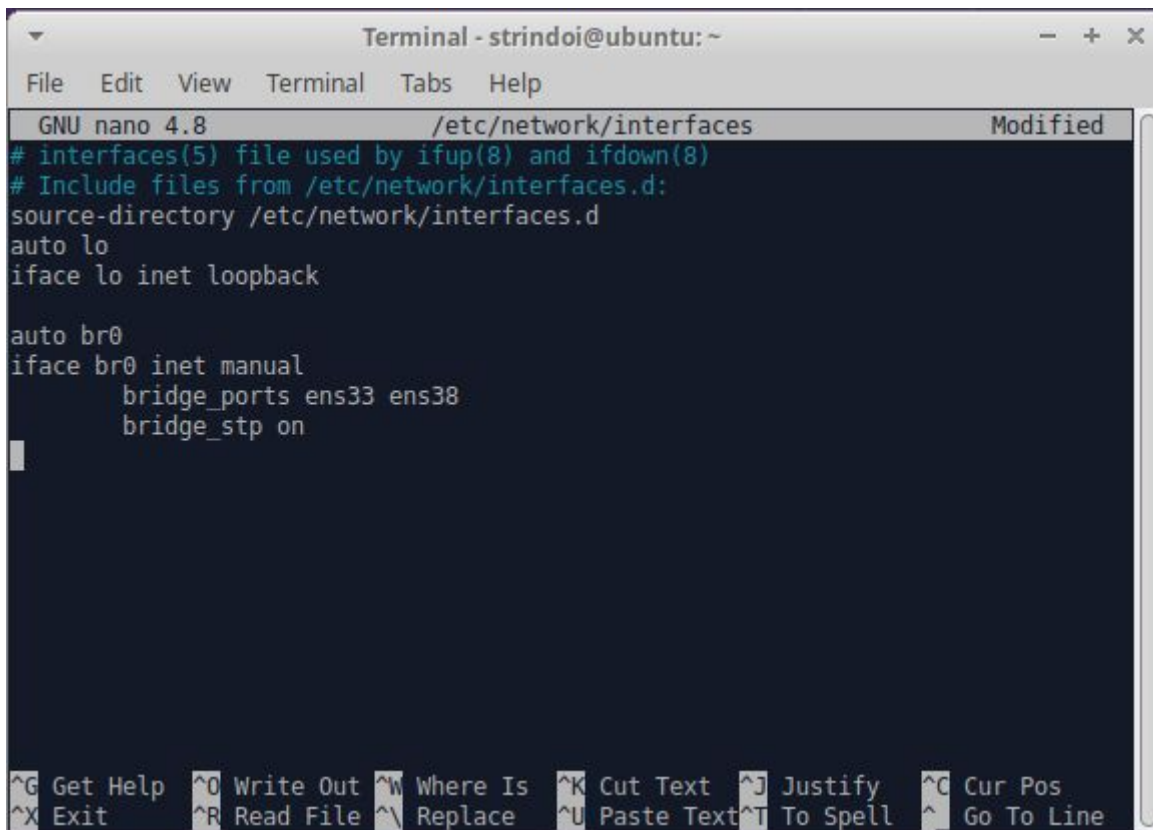
```
strindoi@ubuntu:~$ sudo nano /etc/network/interfaces
```

This opens up the network interfaces file. Add the following to the file:

```
auto lo
iface lo inet loopback

Auto br0
Iface br0 inet manual
    bridge _ports ens33 ens38
    Bridge_stp on
```

Replace ens33 and ens38 with your inet interfaces.



```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help
GNU nano 4.8 /etc/network/interfaces Modified
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto lo
iface lo inet loopback

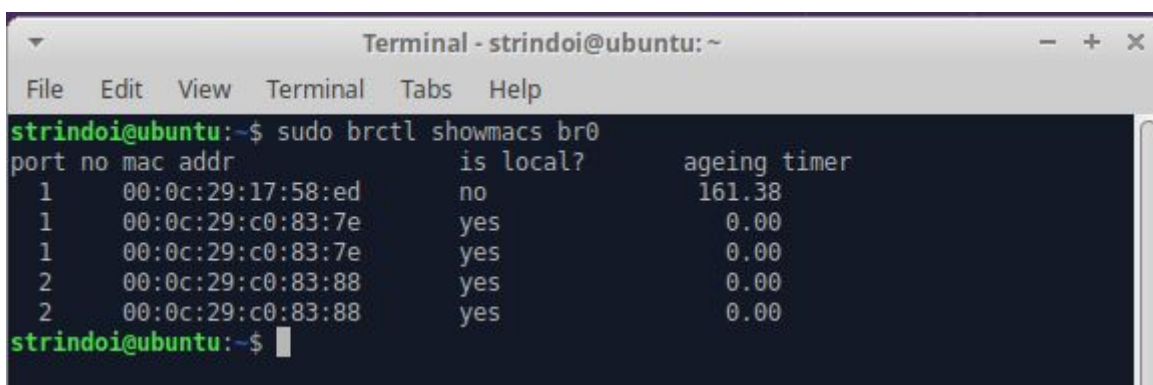
auto br0
iface br0 inet manual
    bridge_ports ens33 ens38
    bridge_stp on
```

Save the file, and then the virtual switch is ready for switching/bridging.

Commands

We used several different commands when using brctl, some of which we have provided explanations for in the following subsections.

Sudo brctl showmacs br0



```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help
strindoi@ubuntu:~$ sudo brctl showmacs br0
port no mac addr          is local?    ageing timer
1      00:0c:29:17:58:ed      no           161.38
1      00:0c:29:c0:83:7e      yes          0.00
1      00:0c:29:c0:83:7e      yes          0.00
2      00:0c:29:c0:83:88      yes          0.00
2      00:0c:29:c0:83:88      yes          0.00
strindoi@ubuntu:~$
```

Sudo brctl showmacs br0 is used to display what MAC addresses are present on bridge 0, and what ports they are connected to.

brctl show

```
Terminal - strindoi@ubuntu: ~  
File Edit View Terminal Tabs Help  
strindoi@ubuntu:~$ brctl show  
bridge name      bridge id        STP enabled      interfaces  
br0               8000.000c29c0837e  yes              ens33  
                  ens38  
strindoi@ubuntu:~$
```

brctl show displays all ethernet bridges and their status.

Sudo brctl stp br0 off/on

```
Terminal - strindoi@ubuntu: ~  
File Edit View Terminal Tabs Help  
strindoi@ubuntu:~$ sudo brctl stp br0 off  
strindoi@ubuntu:~$ sudo brctl stp br0 on  
strindoi@ubuntu:~$
```

Enable or disable the spanning tree protocol within a specific bridge interface (in this case, br0)

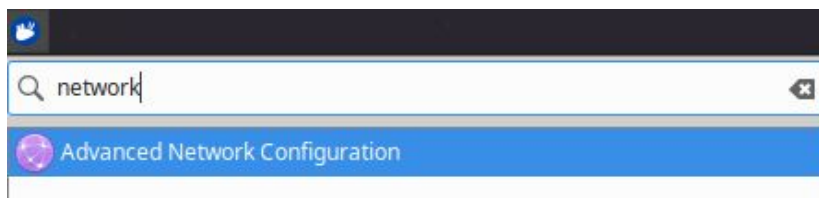


6 IP addresses and Subnet masks

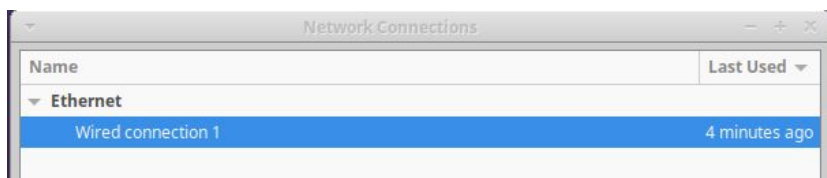
Many situations can call for a static IP address, for example, to establish stable reusable remote access to a certain device, or when you have configured your network adapter to a LAN Segment.

In this example we are using Linux Xubuntu.

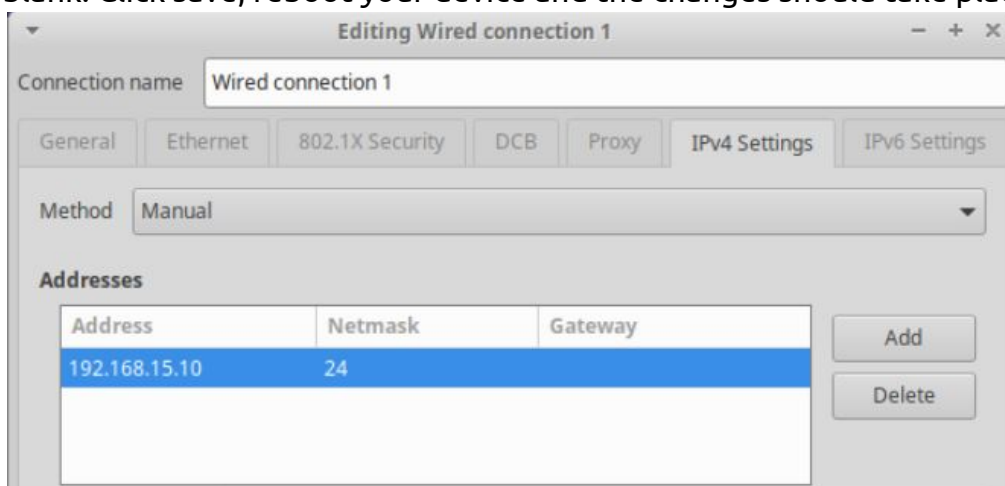
I) Open Network settings as shown in the picture below. In different Ubuntu versions, it might be called Networking instead. In Xubuntu Linux, it is Advanced Network Configuration.



II) Click on your connection type, in this case, Wired connection 1, and navigate to the gear icon located in the left corner of the menu.

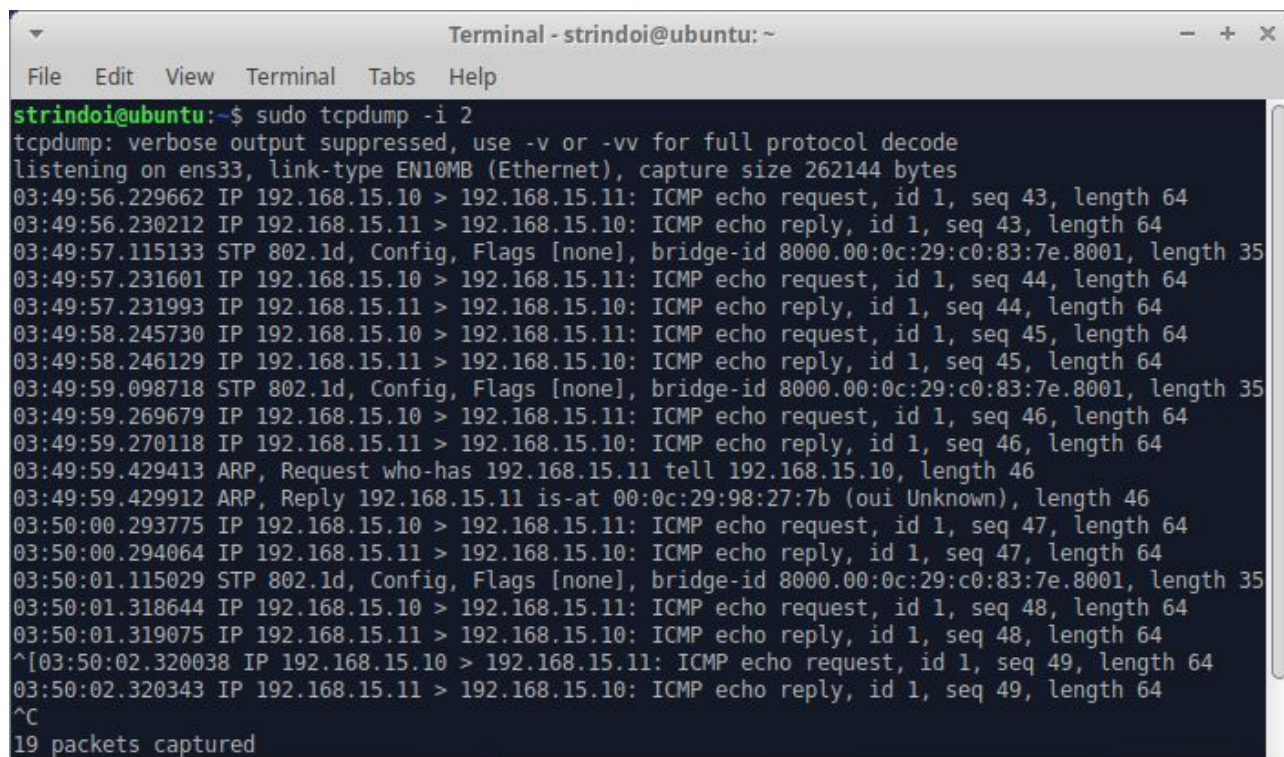


III) Navigate to the IPv4 Settings, change the Method from DHCP to Manual. Click the add button. In the address bar type in your desired address, in this case, 192.168.15.10, netmask stays as 24 (here you can change your subnet mask), and leave the gateway blank. Click save, reboot your device and the changes should take place.



IV) Repeat step 1-3 on the second device.

7 TCPdumps from one computer to another



```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help
strindoi@ubuntu:~$ sudo tcpdump -i 2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
03:49:56.229662 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 43, length 64
03:49:56.230212 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 43, length 64
03:49:57.115133 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:49:57.231601 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 44, length 64
03:49:57.231993 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 44, length 64
03:49:58.245730 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 45, length 64
03:49:58.246129 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 45, length 64
03:49:59.098718 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:49:59.269679 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 46, length 64
03:49:59.270118 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 46, length 64
03:49:59.429413 ARP, Request who-has 192.168.15.11 tell 192.168.15.10, length 46
03:49:59.429912 ARP, Reply 192.168.15.11 is-at 00:0c:29:98:27:7b (oui Unknown), length 46
03:50:00.293775 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 47, length 64
03:50:00.294064 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 47, length 64
03:50:01.115029 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:50:01.318644 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 48, length 64
03:50:01.319075 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 48, length 64
^C[03:50:02.320038 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 49, length 64
03:50:02.320343 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 49, length 64
^C
19 packets captured
```

In the picture above we see a ping from 192.168.15.10 being sent to 192.168.15.11. We can see this using tcpdump which monitors the traffic on an interface, which in this case is interface 2, ens33 (ethernet). In this case the program is run from the virtual switch.

We see multiple kinds of traffic, ICMP echo request/reply, STP and ARP.

The ICMP echo request/reply is the actual ping being sent and replied to.

The STP is shorthand for Spanning Tree Protocol, it helps prevent bridge loops, and the broadcast radiation that results from them (ignore these for now).

The ARP is shorthand for Address Resolution Protocol, it is used to figure out where to direct the traffic.

Commands

We used several different commands when using tcpdump, some of which we have provided explanations for in the following subsections.

tcpdump -D

```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help

strindoi@ubuntu:~$ sudo tcpdump -D
1.br0 [Up, Running]
2.ens33 [Up, Running]
3.ens38 [Up, Running]
4.lo [Up, Running, Loopback]
5.any (Pseudo-device that captures on all interfaces) [Up, Running]
6.bluetooth-monitor (Bluetooth Linux Monitor) [none]
7.nflog (Linux netfilter log (NFLOG) interface) [none]
8.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
strindoi@ubuntu:~$
```

tcpdump -d shows us a directory of interfaces.

We can use this directory of interfaces, for finding out what interface we want to monitor.

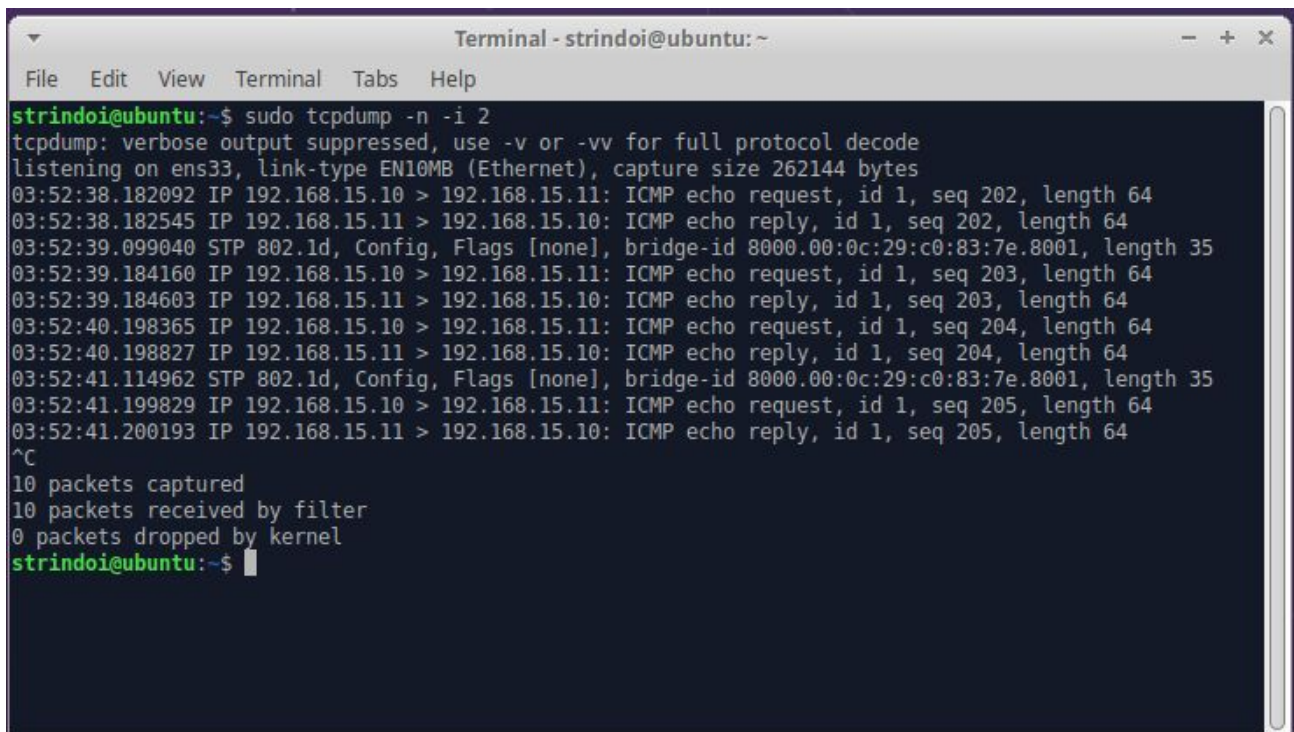
Tcpdump -i <interface name>

```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help

strindoi@ubuntu:~$ sudo tcpdump -i 2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
03:49:56.229662 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 43, length 64
03:49:56.230212 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 43, length 64
03:49:57.115133 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:49:57.231601 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 44, length 64
03:49:57.231993 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 44, length 64
03:49:58.245730 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 45, length 64
03:49:58.246129 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 45, length 64
03:49:59.098718 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:49:59.269679 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 46, length 64
03:49:59.270118 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 46, length 64
03:49:59.429413 ARP, Request who-has 192.168.15.11 tell 192.168.15.10, length 46
03:49:59.429912 ARP, Reply 192.168.15.11 is-at 00:0c:29:98:27:7b (oui Unknown), length 46
03:50:00.293775 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 47, length 64
03:50:00.294064 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 47, length 64
03:50:01.115029 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:50:01.318644 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 48, length 64
03:50:01.319075 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 48, length 64
^C[03:50:02.320038 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 49, length 64
03:50:02.320343 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 49, length 64
^C
19 packets captured
```

Tcpdump -i <interface name>, in this case, 2. Is used to tell tcpdump which interface we want to monitor. The number 2 specifies the interface number.

Tcpdump -n -i <interface name>

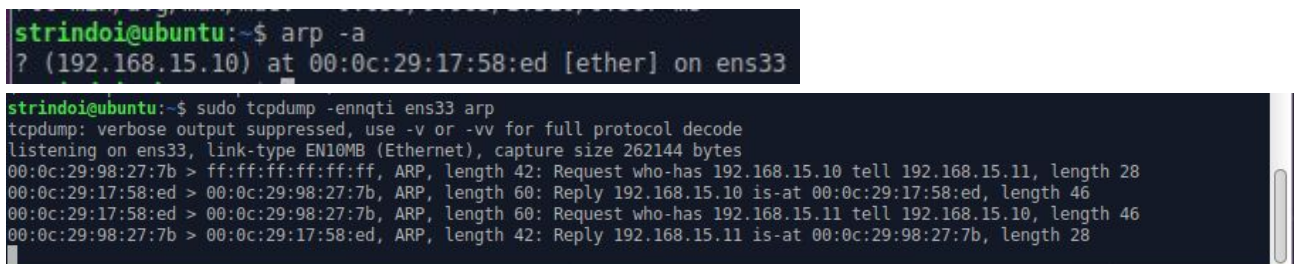


```
Terminal - strindoi@ubuntu: ~
File Edit View Terminal Tabs Help

strindoi@ubuntu:~$ sudo tcpdump -n -i 2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
03:52:38.182092 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 202, length 64
03:52:38.182545 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 202, length 64
03:52:39.099040 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:52:39.184160 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 203, length 64
03:52:39.184603 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 203, length 64
03:52:40.198365 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 204, length 64
03:52:40.198827 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 204, length 64
03:52:41.114962 STP 802.1d, Config, Flags [none], bridge-id 8000.00:0c:29:c0:83:7e.8001, length 35
03:52:41.199829 IP 192.168.15.10 > 192.168.15.11: ICMP echo request, id 1, seq 205, length 64
03:52:41.200193 IP 192.168.15.11 > 192.168.15.10: ICMP echo reply, id 1, seq 205, length 64
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
strindoi@ubuntu:~$
```

tcpdump -n is used to tell tcpdump not to convert addresses (i.e., host addresses, port numbers, etc.) to names. Example: Let's say we were pinging google.com instead of an ip address, then the option -n would convert google.com into their ip address.

8 Show ARP tables



```
strindoi@ubuntu:~$ arp -a
? (192.168.15.10) at 00:0c:29:17:58:ed [ether] on ens33

strindoi@ubuntu:~$ sudo tcpdump -ennqti ens33 arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
00:0c:29:98:27:7b > ff:ff:ff:ff:ff:ff, ARP, length 42: Request who-has 192.168.15.10 tell 192.168.15.11, length 28
00:0c:29:17:58:ed > 00:0c:29:98:27:7b, ARP, length 60: Reply 192.168.15.10 is-at 00:0c:29:17:58:ed, length 46
00:0c:29:17:58:ed > 00:0c:29:98:27:7b, ARP, length 60: Request who-has 192.168.15.11 tell 192.168.15.10, length 46
00:0c:29:98:27:7b > 00:0c:29:17:58:ed, ARP, length 42: Reply 192.168.15.11 is-at 00:0c:29:98:27:7b, length 28
```

On the first picture, the arp table from one of the virtual machines is shown after it has learned the MAC address of the other virtual machine. It will also display on which interface the machine is connected through.

On the second picture, a tcpdump of an ARP exchange is shown. First, VM1 will send out a broadcast message, looking for 192.168.15.10 (VM2), and to send the reply to 192.168.15.10 (in this case, itself). A response is received, telling VM1 that VM2 can be found using the mac address: 00:0c:29:17:58:ed.

9 MAC table of switch

```
strindoi@ubuntu:~$ sudo brctl showmacs br0
port no mac addr          is local?    ageing timer
 1    00:0c:29:17:58:ed     no           228.56
 2    00:0c:29:98:27:7b     no           251.66
 1    00:0c:29:c0:83:7e     yes          0.00
 1    00:0c:29:c0:83:7e     yes          0.00
 2    00:0c:29:c0:83:88     yes          0.00
 2    00:0c:29:c0:83:88     yes          0.00
```

The table is currently populated by the learned MAC addresses from the virtual machines. The table will show which port the MAC address was learned from, whether it is a local address or not, and how long until the address will be cleared from the table if nothing has been passed from it.

```
strindoi@ubuntu:~$ sudo ifdown br0
run-parts: failed to stat component /etc/network/if-post-down.d/avahi-daemon: No such file or directory
strindoi@ubuntu:~$ sudo brctl showmacs br0
read of forward table failed: No such device
strindoi@ubuntu:~$ sudo ifup br0

Waiting for br0 to get ready (MAXWAIT is 32 seconds).
strindoi@ubuntu:~$ sudo brctl showmacs br0
port no mac addr          is local?    ageing timer
 1    00:0c:29:c0:83:7e     yes          0.00
 1    00:0c:29:c0:83:7e     yes          0.00
 2    00:0c:29:c0:83:88     yes          0.00
 2    00:0c:29:c0:83:88     yes          0.00
```

The ifdown command will turn off the interface or bridge and the ifup command will turn on the interface or bridge.

10 Conclusion

The students have now shown their capability in regards to setting up a virtual switch based on linux using brctl. A demonstration on setting up static IP addresses and showing connectivity between machines was also completed.

A description of how the layer 2 switching works was written, how to access and show a hosts MAC table, while quickly explaining the specific mechanics and commands used. The theory and application of ARP was also displayed by the students.

TCPdump was used to display relevant information and demonstrate how network devices communicate with each other.