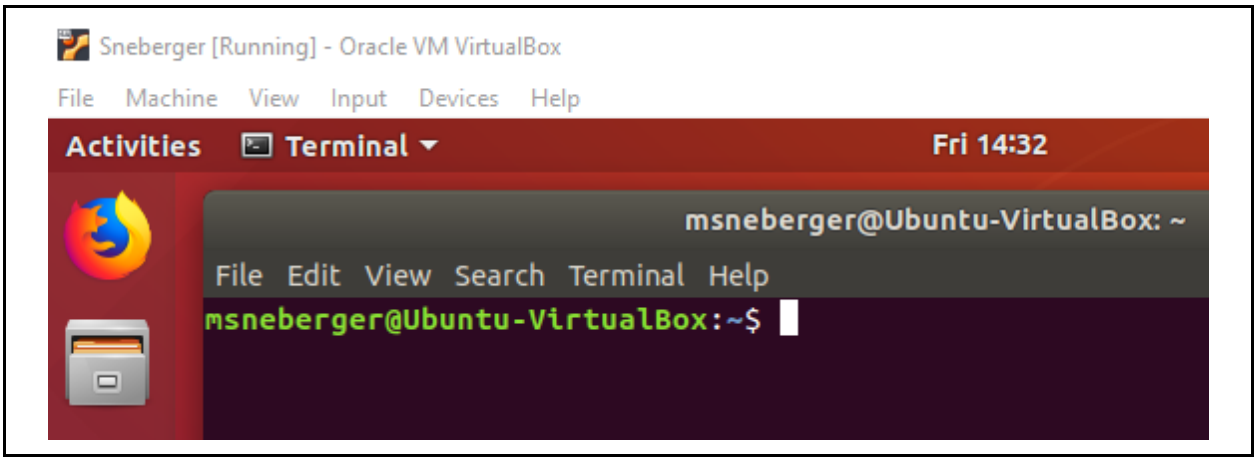
Michael Sneberger msneberg@asu.edu
ASU ID Number: 1000001544

# CSE 434 Computer Networks (Fall 2019) Assignment 1

This assignment is due on 11:59 pm September 11, 2019. It requires a certain amount of hands-on experiences, which can easily take more than 10 hours if you are new to Linux and without any knowledge of these tools. So please start early and search online from time to time to make sure you understand what you are doing. Please submit a single PDF document answering all the questions. You can just make a copy of this document and replace the blue part with your answers. Please do not forget to put your name and ASUID in the submitted document. The grading is effort-base. Note that you can leave a comment on this assignment specification document. If you have a question, just write it down as a comment so that other students with the same question can also obtain the answer. I will check this document regularly and answer your questions.

**Task 1.** Please install **VirtualBox** on your machine, create a virtual machine, and install **Ubuntu Linux** in the virtual machine. Make sure that the virtual machine is connected to the Internet. You may also install Ubuntu natively on your computer. Either way is OK. The remaining questions will be in this Linux environment, mainly under the command line instead of the GUI.



**Task 2. 0)** What does the **ifconfig** command do? If Ubuntu complains that the command cannot be found, just install the package the command requires. **1)** Please explain each line of the output of ifconfig for your first Ethernet or WiFi network interface (i.e., the interface that allows you to connect to the internet, usually it has valid IP addresses). **2)** What does the "lo" interface do? (Hint: "lo" is "loopback".) What happens if you send some data to the "lo" interface?

```
msneberger@Ubuntu-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e37e:66a7:b388:6d1a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:0c:8d:78  txqueuelen 1000  (Ethernet)
        RX packets 458  bytes 383405 (383.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 358  bytes 47798 (47.7 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 122  bytes 11107 (11.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 122  bytes 11107 (11.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

msneberger@Ubuntu-VirtualBox:~$
```

0) Ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed. If no arguments are given, ifconfig displays the status of the currently active interfaces. If a single interface argument is given, it displays the status of the given interface only; if a single -a argument is given, it displays the status of all interfaces, even those that are down. Otherwise, it configures an interface.

1) enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
   - enp0s3 is a predictable network interface name that includes hardware information which became available with v197 of Ubuntu and includes hardware information rather than a reference to the driver used like eth0
   - flags=4163<UP,BROADCAST,RUNNING,MULTICAST> describes the hardware facilitating the system's connection to the internet
   - mtu 1500 signifies that the Maximum Transmission Unit which is the maximum number of octects the interface is able to handle in one transaction is set to the default 1500
     a. inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        - inet = Internet Protocol Family is a collection of protocols layered atop the Internet Protocol (IP) transport layer and utilizing the Internet address format and providing protocol support for SOCK-STREAM, SOCK_DGRAM, and SOCK_RAW socket types with the SOCK_RAW interface providing access to the IP protocol
        - netmask = a 32-bit binary mask used to divide an IP address int subnets and specify the network's available hosts
        - broadcast = the IP address reserved for sending messages to all nodes on the network or network segment

b. inet6 fe80::e37e:66a7:b388:6d1a  prefixlen 64  scopeid 0x20<link>
   - inet6 = signifies a collection of protocols layered atop Internet Protocol version 6 (IPv6) transport layer and here signifies that what follows is the IPv6 address
   - prefixlen 64 = specifies the number of bits in the IP address that are to be used as the subnet mask. For IPv6 length must be less than or equal to 128 bits with the 64 here being the default
   - scopeid = a character string which is appended to the NetBIOS name for all NetBIOS over TCP/IP communications. It provides a method to isolate a collection of computers that only communicate with each other

c. ether 08:00:27:0c:8d:78  txqueuelen 1000  (Ethernet)
   - ether = signifies that the address of the ethernet follows
   - txqeuelen = transmit que length which is a parameter of an interface in the Linux kernel. It limits the number of packets in the transmission queue in the interface's device driver, *i.e.* instructs the kernel on the largeness of transmit queue of the network interface device

d. RX packets 458  bytes 383405 (383.4 KB) = number of packets received

e. RX errors 0  dropped 0  overruns 0  frame 0 = packet errors receive

f. TX packets 358  bytes 47798 (47.7 KB) = number of packets transmitted

g. TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0 = packet errors transmit

2) lo or the loopback interface is the special interface the system uses to communicate with itself. When a network interface is disconnected--for example, when an Ethernet port is unplugged or Wi-Fi is turned off or not associated with an access point--no communication on that interface is possible, not even communication between your computer and itself. The loopback interface does not represent any actual hardware, but exists so applications running on your computer can always connect to servers on the same machine. If you send data to the lo interface it comes back to the machine even of the connection to the LAN is disconnected

**Task 3.** What does the **ping** command do? Please type "ping -c 4 8.8.8.8" and explain the output. What is this "8.8.8.8" IP address? Please briefly explain what happens behind the curtain when the ping command runs. A simple google search can give you all the answers.

```
msneberger@Ubuntu-VirtualBox:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=251 time=63.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=251 time=21.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=251 time=23.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=251 time=24.4 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 21.623/33.332/63.962/17.713 ms
msneberger@Ubuntu-VirtualBox:~$
```

0) Explanation of ping output:
   a. The -c flag is for “count” and after sending count number (here 4) of ECHO_REQUEST packets the ping function stops which is evident in four lines of main output. Without this limit ping will run until you kill it with Ctrl+C upon which statistics are reported

b. The 8.8.8.8 IP address is for recursive name servers operated by Google Public DNS using IPV6 (for IPV4 you would use 8.8.4.4) and routing to the nearest operational server. Google Public DNS is fast as steps have been taken to reduce latency. The long version of this address is 2001:4860:4860:8888

1) What happens when "ping" runs? ping uses the ICMP protocol's ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (`"pings") have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of "pad" bytes used to fill out the packet. ping works with both IPv4 and IPv6. Using only one of them explicitly can be enforced by specifying -4 or -6. ping can also send IPv6 Node Information Queries (RFC4620). Intermediate hops may not be allowed, because IPv6 source routing was deprecated q(RFC5095).

**Task 4. 0)** What does the **netstat** command do? **1)** The netstat command has many options. Please explain the following commonly used options and briefly explain the meaning of the output. How can you know which program (i.e., process) is using a specific port?

```
duolu@duolu-desktop:~$ netstat -a
duolu@duolu-desktop:~$ netstat -l
duolu@duolu-desktop:~$ netstat -s
duolu@duolu-desktop:~$ netstat -p
duolu@duolu-desktop:~$ netstat -r
duolu@duolu-desktop:~$ netstat -ie
```

You can ignore those lines related to UNIX domain sockets.

0) netstat prints network connections, routing tables, interface statistics, masquerade connections and multicast memberships. Various flags are available to tailor netstat output of information regarding the Linux networking subsystem:

1) netstat -a stands for - -all and shows both listening and non-listening sockets. With the - - interfaces option, show interfaces that are not up

```
msneberger@Ubuntu-VirtualBox:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:netbios-ssn     0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
```

a. Proto = what protocol is used by the socket
b. Recv-Q = the count of bytes not copied by the user program connected to this socket
c. Send-Q = the count of bytes not acknowledged by the remote host
d. Local Address = Address and port number of the local end of the socket. Unless the --numeric (-n) option is specified, the socket address is resolved to its canonical host name (FQDN), and the port number is translated into the corresponding service name
e. Foreign Address = address and port number of the remote end of the socket. Analogous to "Local Address"
f. State = the state of the socket. Since there are no states in raw mode and usually no states used in UDP and UDPLite, this column may be left blank. Normally this can be one of several values:
    i. ESTABLISHED = the socket has an established connection.

ii. SYN_SENT = the socket is actively attempting to establish a connection.
iii. SYN_RECV = a connection request has been received from the network.
iv. FIN_WAIT1 = the socket is closed, and the connection is shutting down.
v. FIN_WAIT2 = connection is closed, and the socket is waiting for a shutdown from the remote end.
vi. TIME_WAIT = the socket is waiting after close to handle packets still in the net work.
vii. CLOSE = the socket is not being used.
viii. CLOSE_WAIT = The remote end has shut down, waiting for the socket to close.
ix. LAST_ACK = the remote end has shut down, and the socket is closed. Waiting for acknowledgement.
x. LISTEN = the socket is listening for incoming connections. Such sockets are not included in the output unless you specify the --listening (-l) or - -all (-a) option.
xi. CLOSING = both sockets are shut down but we still don't have all our data sent.
xii. UNKNOWN = the state of the socket is unknown.

2) netstat -l stands for - -listening and shows only listening sockets which are ordinarily omitted by default

```
msneberger@Ubuntu-VirtualBox:~$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:netbios-ssn     0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
```

a. See netstat -a above at 1) for explanation of various fields

3) netstat -s stands for - -statistics and displays summary statistics for each protocol (IP/ICMPTCP/UDP)

```
msneberger@Ubuntu-VirtualBox:~$ netstat -s
Ip:
    Forwarding: 2
    4126 total packets received
    1 with invalid addresses
    0 forwarded
```

a. Ip stands for Internet Protocol (IP) and statistics are provided

4) netstat -p stands for - -program and shows the PID and name of program to which each socket belongs

```
msneberger@Ubuntu-VirtualBox:~$ netstat -p
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Progr
am name
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ]         DGRAM                    46909    10948/systemd        /run/user
/1000/systemd/notify
unix  2      [ ]         DGRAM                    23470    -                    /run/user
/121/systemd/notify
unix  2      [ ]         DGRAM                    21455    -                    /var/lib/
samba/private/msg.sock/959
unix  2      [ ]         DGRAM                    21296    -                    /var/lib/
```

* * *

```
unix  8      [ ]         DGRAM                    12536    -                    /run/syst
emd/journal/socket
unix  3      [ ]         STREAM     CONNECTED     49993    11290/gsd-wacom
unix  3      [ ]         STREAM     CONNECTED     24376    -
unix  3      [ ]         STREAM     CONNECTED     24299    -
unix  3      [ ]         STREAM     CONNECTED     51084    10974/dbus-daemon    /run/user
/1000/bus
```

a. See netstat -a above at 1) for explanation of Proto, Recv-Q, Send-Q
b. RefCnt = the reference count, *i.e*. attached processes via this socket
c. Flags could be SO_ACCEPTON (displayed as **ACC**), SO_WAITDATA (**W**) or SO_NOSPACE (**N**). SO_ACCECPTON is used on unconnected sockets if their corresponding processes are waiting for a connect request. The other flags are not of normal interest
d. DGRAM = datagram socket which uses UDP protocol
e. STREAM = stream socket which uses TCP protocol
f. State this field will contain one of the following Keywords:
    i. FREE = the socket is not allocated
    ii. LISTENING = the socket is listening for a connection request. Such sockets are only included in the output if you specify the **--listening** (**-l**) or **--all** (**-a**) option
    iii. CONNECTING = the socket is about to establish a connection
    iv. CONNECTED = the socket is connected
    v. DISCONNECTING = he socket is disconnecting
    vi. (empty) = he socket is not connected to another one
    vii. UNKNOWN - this state should never happen
g. I-Node = an entry in an inode table which contains information or metadata about a regular file or directory. An inode does not sotre the name of the file but its contents only. An inode contains a list of all the blocks in which a file is stored, the owner information for that file, permissions and all other attributes that are set for the file. In a sense, you could say that a file really is the inode, and names are attached to these inodes to make it easier for humans to work with them

5) netstat -r stand for - -route and displays the kernel routing tables with route -e producing the same output

```
msneberger@Ubuntu-VirtualBox:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
msneberger@Ubuntu-VirtualBox:~$
```

a. Destination identifies the destination network
b. Gateway identifies the defined gateway for the specified network
    i. An asterisk appears in the Gateway column if no forwarding gateway is needed for the network
c. Genmask shows the *netmask* for the network; in this case, it is 255.255.255.0
d. Flags
    i. A = Receive all multicast at this interface
    ii. B = OK broadcast
    iii. D = Debugging ON
    iv. M = Promiscuous Mode
    v. O = No ARP at this interface
    vi. P = P2P connection at this interface
    vii. R = Interface is running
    viii. U = Interface is up

ix. G = Not a direct entry

e. MSS indicates the default Maximum Segment Size for TCP connections over this route
f. Window indicates the default window size for TCP connections over this route
g. Irtt indicates the Initial Round Trip Time for this route. The kernel uses this to select values for certain TCP parameters without having to wait for potentially slow answers from remote hosts
h. Iface shows the network interface. If you had more than one interface, you would see *lo* (for loopback)

6) netstat -ie

```
msneberger@Ubuntu-VirtualBox:~$ netstat -ie
Kernel Interface table
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e37e:66a7:b388:6d1a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:0c:8d:78  txqueuelen 1000  (Ethernet)
        RX packets 8043  bytes 6617388 (6.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4002  bytes 295083 (295.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

a. -i stands for - -interfaces and displays a table of all network interfaces
b. -e stands for - -extend and displays additional information and -ee displays maximum detail
c. See Task 2 section 1) for description of output

**Task 5.** **0)** What is the public IP address of your machine? **a-e)** How can you know your public IP address of the machine? (Hint: ask Google.) **1)** How does a website know your public IP address? **2)** Is it different from the IP address shown on the network interface obtained by ifconfig? **3)** Why are they different? (Hint: NAT.) What does this NAT thing do (briefly explain it in one sentence)? **4)** Who did it? (Hint: the VirtualBox software may do it again in addition to your ISP's firewall or router.)

0) Finding public IP address of my VirtualBox VM hosted on Windows 10 host:
   a. Using dig = 70.171.228.4

```
msneberger@Ubuntu-VirtualBox:~$ dig +short myip.opendns.com @resolver1.opendns.c
om
70.171.228.4
msneberger@Ubuntu-VirtualBox:~$ dig TXT +short o-o.myaddr.l.google.com @ns1.goog
le.com
"70.171.228.4"
msneberger@Ubuntu-VirtualBox:~$
```

   b. Using host = 70.171.228.4

```
msneberger@Ubuntu-VirtualBox:~$ host myip.opendns.com resolver1.opendns.com
Using domain server:
Name: resolver1.opendns.com
Address: 208.67.222.222#53
Aliases:

myip.opendns.com has address 70.171.228.4
Host myip.opendns.com not found: 3(NXDOMAIN)
Host myip.opendns.com not found: 3(NXDOMAIN)
msneberger@Ubuntu-VirtualBox:~$
```

c. Using wget = 70.171.228.4

```
msneberger@Ubuntu-VirtualBox:~$ wget -qO- http://ipecho.net/plain | xargs echo
70.171.228.4
msneberger@Ubuntu-VirtualBox:~$ wget -qO - icanhazip.com
70.171.228.4
msneberger@Ubuntu-VirtualBox:~$
```

d. Using curl = 70.171.228.4

```
msneberger@Ubuntu-VirtualBox:~$ curl ifconfig.me
70.171.228.4msneberger@Ubuntu-VirtualBox:~$ curl icanhazip.com
70.171.228.4
msneberger@Ubuntu-VirtualBox:~$ curl ipecho.net/plain
70.171.228.4msneberger@Ubuntu-VirtualBox:~$ curl ifconfig.co
70.171.228.4
msneberger@Ubuntu-VirtualBox:~$ curl ipinfo.io/ip
70.171.228.4
msneberger@Ubuntu-VirtualBox:~$
```

e. Using whatismyip.com/what-is-my-public-ip-address = IPv4 70.171.228.4 Local IP 192.168.0.21 and IPv6 2600:8800:b80:86a;1cc2:23b2:9ca2:ec8

Your Public IPv6 Address Is: 2600:8800:b80:86a:1cc2:23b2:9ca2:5ec8

Your Public IPv4 is: 70.171.228.4

Your Local IP is: 192.168.0.21

f. Store my IP address in shell variable: I added this in as an extra bonus

```
msneberger@Ubuntu-VirtualBox:~$ myip="$(dig +short myip.opendns.com @resolver1.o
pendns.com)"
msneberger@Ubuntu-VirtualBox:~$ echo "My WAN/Public IP address: ${myip}"
My WAN/Public IP address: 70.171.228.4
msneberger@Ubuntu-VirtualBox:~$
```

1) Websites know your public IP address because Your public IP address is the IP address that is logged by various servers/devices when you connect to them through your internet connection
2) The public IP address obtained above is different from the IP address found through other avenues including ifconfig:
   a. Using ifconfig = 10.0.2.15 or 127.0.0.1 (this is what is used for Putty SSH and Filezilla via Port 2222)

```
msneberger@Ubuntu-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::e37e:66a7:b388:6d1a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:0c:8d:78  txqueuelen 1000  (Ethernet)
        RX packets 17980  bytes 16270804 (16.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8159  bytes 766590 (766.5 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 950  bytes 94934 (94.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 950  bytes 94934 (94.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

b. Using ip a = 10.0.2.15/24 or 127.0.0.1/8

```
msneberger@Ubuntu-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:0c:8d:78 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 80831sec preferred_lft 80831sec
    inet6 fe80::e37e:66a7:b388:6d1a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

c. Using hostname -I = 10.0.2.15

```
msneberger@Ubuntu-VirtualBox:~$ hostname -I
10.0.2.15
```

d. Using cat /etc/hosts = 127.0.0.1 with 127.0.1.1 for VirtualBox

```
msneberger@Ubuntu-VirtualBox:~$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       Ubuntu-VirtualBox

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
msneberger@Ubuntu-VirtualBox:~$
```

e. Running ipconfig from the cmd line of my Windows 10 host laptop = 192.168.56.1

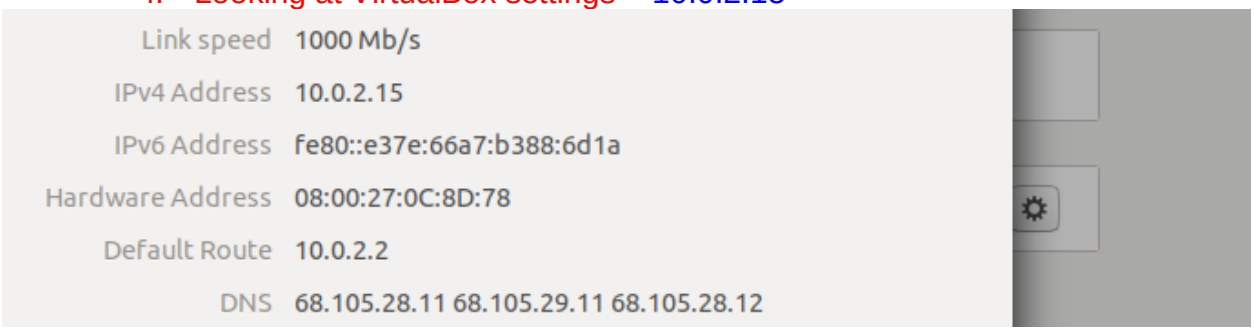
```
C:\Users\Michael>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::8873:837d:b1fe:f10e%19
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
```

f. Looking at VirtualBox settings = 10.0.2.15



3) The difference in the IP address we find via the various methods to find our public IP address and ifconfig is because each device on your network has a private IP address only seen by other devices on the local network. But your ISP assigns you a public IP address that other devices on the Internet can see
4) NAT stands for Network Address Translation which is the technique of rewriting addresses on a packet as it passes through a routing device.

**Task 6.** **0)** Can you make your Linux machine a router if you have multiple network interface cards? **1)** Does the routing table only exists on the router instead of on your host machine? Why? **2)** What does the **route** command do? Please run the command and explain the output.

0) Yes
1) On Linux systems, information on how packets are to be forwarded is stored in a kernel structure called a routing table, so no, the routing table does not only exist on the router. You can examine the routing table by running the command netstat -r.

```
msneberger@Ubuntu-VirtualBox:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
msneberger@Ubuntu-VirtualBox:~$ route
```

2) The route command manipulates the kernel's IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the ifconfig(8) program

```
msneberger@Ubuntu-VirtualBox:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
msneberger@Ubuntu-VirtualBox:~$
```

a. Destination is the destination network of destination host
b. Gateway is the gateway address or '*' if none set
c. Genmask is the netmask for the destination net; '255.255.255.255' for a host destination and '0.0.0.0' for a default route
d. Flag U means route is up
e. Flag G means use gateway
f. Metric is the 'distance' to the target (usually in hops)
g. Ref is the number of references to this route (not used in Linux Kernel)
h. Use is the count or lookups for the route. Depending on the use of -F and -C this will be either route cache misses (-F) or hits (-C)
    i. Flag -F operates on the kernel's Forwarding Information Base (FIB) routing table. This is the default
    ii. Flag -C operates on the kernel's routing cache
i. Iface stands for interface to which packets will be sent

**Task 7.** We mentioned that IP packets are forwarded hop-by-hop. **0)** How do you know how many hops are there if you ping 8.8.8.8? **1)** What does the **traceroute** command do? How does this command work? **2)** Can you explain how it works using the output of the traceroute command generated on your machine?

0) The number of hops are numbered in the output of the traceroute command – here 2
    a. The default is that the output will show a maximum of 64 hops

```
msneberger@Ubuntu-VirtualBox:~$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max
  1   10.0.2.2  0.629ms  0.005ms  0.242ms
  2   10.0.2.2  3.837ms  2.167ms  2.143ms
msneberger@Ubuntu-VirtualBox:~$
```

*You can see from the above output that the fact that I am running Ubuntu from a VirtualBox virtual machine is keeping traceroute from seeing the full path. However, when I run tracert 8.8.8.8 from my host computer's Windows command line I get the following full path:*

```
C:\Users\Michael>tracert 8.8.8.8

Tracing route to dns.google [8.8.8.8]
over a maximum of 30 hops:

  1     4 ms    <1 ms    <1 ms  192.168.0.1
  2     8 ms     7 ms    12 ms  10.113.124.1
  3    18 ms     9 ms     8 ms  100.127.73.32
  4    19 ms    10 ms     9 ms  100.120.100.0
  5    21 ms    22 ms    21 ms  langbprj02-ae1.0.rd.la.cox.net [68.1.1.14]
  6    21 ms    20 ms     *     72.14.196.240
  7    22 ms    22 ms    23 ms  108.170.247.225
  8     *       25 ms    18 ms  209.85.242.59
  9    22 ms    27 ms    21 ms  dns.google [8.8.8.8]

Trace complete.

C:\Users\Michael>
```

*A fellow student suggested that if I added a -I flag to the traceroute command it would overcome the issue of only seeing the link between the VM and my host laptop. He was correct. The man page for traceroute says the the -I flag stands for - -icmp and uses ICMP ECHO as probe.*

```
msneberger@Ubuntu-VirtualBox:~$ traceroute -I 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max
  1   10.0.2.2  0.260ms  0.424ms  0.223ms
  2   192.168.0.1  37.859ms  1.766ms  1.631ms
  3   10.113.124.1  22.721ms  7.820ms  8.201ms
  4   100.127.73.32  8.372ms  9.731ms  8.563ms
  5   100.120.100.0  10.186ms  9.247ms  10.204ms
  6   68.1.1.14  22.659ms  20.881ms  23.047ms
  7   72.14.196.240  21.964ms  22.495ms  22.587ms
  8   108.170.247.225  23.207ms  22.346ms  23.340ms
  9   209.85.242.59  22.662ms  21.012ms  21.947ms
 10   8.8.8.8  22.446ms  20.890ms  22.686ms
msneberger@Ubuntu-VirtualBox:~$
```

1) traceroute traces the route to a host by making a series of UDP packets containing the following:
    a. Source address = IP address
    b. Destination address = in this case 8.8.8.8
    c. A destination UDP port number which is invalid (in the range of 33434-33534)
    d. The first packet will have a time to live (TTL) value of 1 which will fall to 0 on the first hop and return a "TTL Time Exceeded" message which informs traceroute of the ID of the first node.

i. Then the subsequent packet will have a TTL value of 2 which will fall to 0 on the second hop and return the ID of the second node
ii. Each successive packet has it TTL value increased by one until traceroute gathers ID data on each node on the way to the destination

e. Ultimately, when the last packet makes it to dns.google server that is 8.8.8.8 in our example, that server will return a “ICMP Destination/PORT Unreachable” message because traceroute sent to an unused port and this message will inform traceroute that the destination is the last node on the route

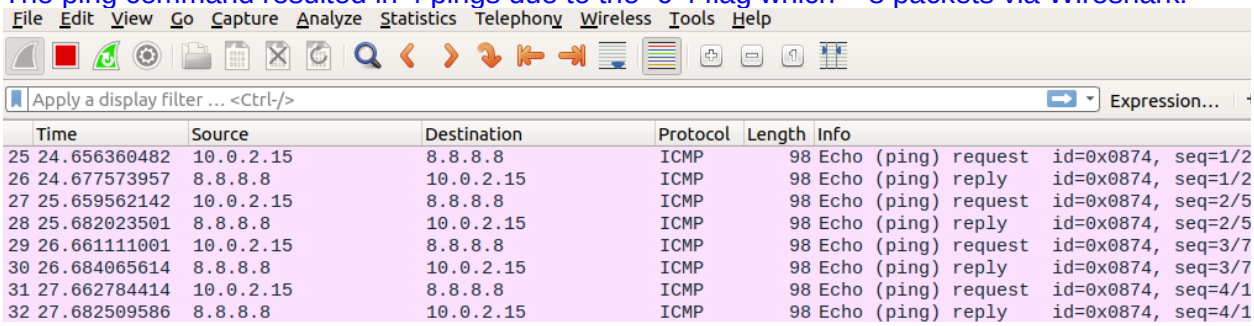
2) Looking at the tracert command I ran in windows above:

a. My machine send a UDP packet with a TTL value equal to the number of the node as listed on the output to each of the following nodes at an unused port, and until the destination the TTL value fell to 0 at the target node which returned a TTL Time Exceeded message containing its identity:
i. **Node 1**, or 10.0.2.2 is the default route of my VirtualBox virtual machine
ii. **Node 2**, or 192.168.0.1 is the Cox Cable Gateway that facilitates my Internet service, as such when the first packet went out it sent back the TTL Time Exceeded message in a mere millisecond
iii. **Nodes 3-5** are private networks you cannot identify
iv. **Node 6**, or 68.1.1.14 is in Fairfax, Virginia and is owned my by ISP Cox Cable
v. **Nodes 7-9** are Google close to Wichita, Kansas

b. Ultimately, my machine sent a UDP packet with a TTL value equal to the number of the destination node as listed on the output and the destination node returned an ICMP Destination/PORT Unreachable message containing its identity:
i. **Node 10** is the actual dns.google destination and is any anycast address so it goes to the server closest to me in hops but you cannot tell what particular server

**Task 8.** Please install **wireshark**. It is a packet sniffing tool that allows you to capture and analyze packets in and out of your machine. Now please open wireshark, listen to the network interface that actually sends out and receives your packets, then run the "ping -c 4 8.8.8.8" command. Try to locate those packets related to this ping command. How can you locate them?

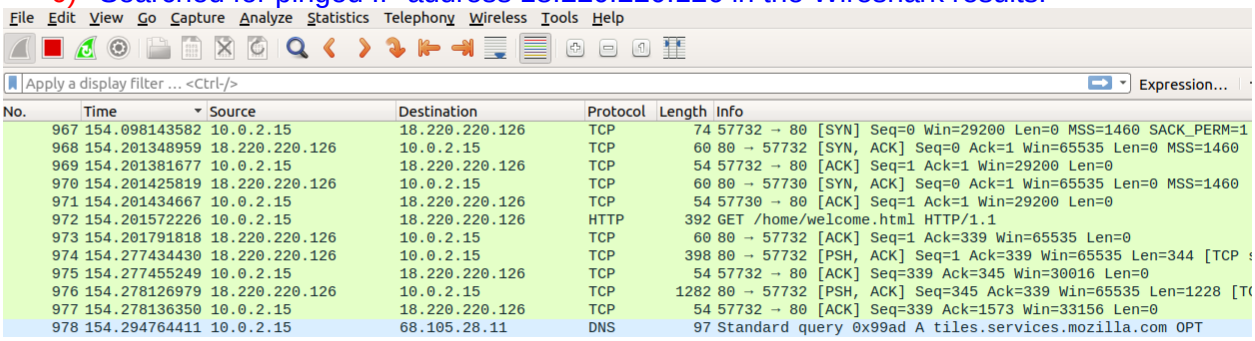
The ping command resulted in 4 pings due to the -c 4 flag which = 8 packets via Wireshark:



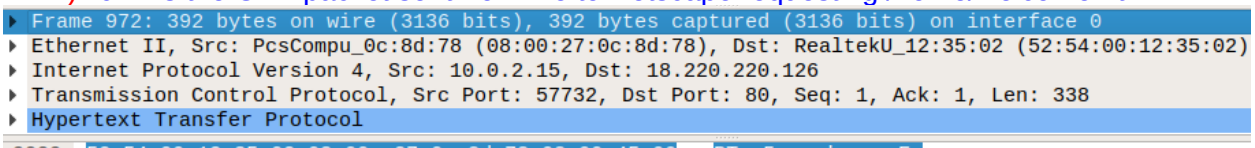

**Task 9.** Still on **wireshark**. This time, open wireshark, listen to the network interface, and open browser, visit "http://home.mcom.com/home/welcome.html". 0) Now please figure out all the packets related to open this web page. Follow a TCP flow (i.e., a "flow" is a sequence of packets with the same source IP, destination IP, source port, destination port, and in the same transport layer protocol such as TCP). **1)** Pick those packets of the three-way handshake. (Hint: a SYN packet, a SYN/ACK packet, and an ACK immediately after these two packets.) **2)** Then

find the packet with the HTTP GET request. Show the bytes of this packet, and figuring out **a)** which part is the link layer header, **b)** which part is the IP header, **c)** which part is the TCP header, and **d)** which part is in HTTP. Show a screenshot.

0) Searched for pinged IP address 18.220.220.126 in the Wireshark results:

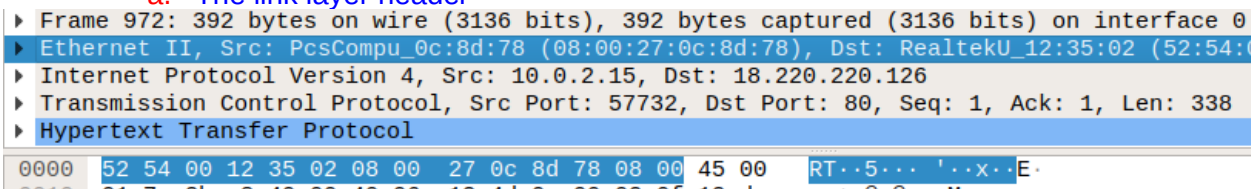



1) The packets of the three-way handshake:
   a. 967 is the SYN packet from me to Netscape
   b. 968 is the SYN/ACK packet from Netscape back to me
   c. 969 is the ACK packet that from me to Netscape completing the handshake
2) 972 is the GET packet sent from me to Netscape requesting /home/welcome.html

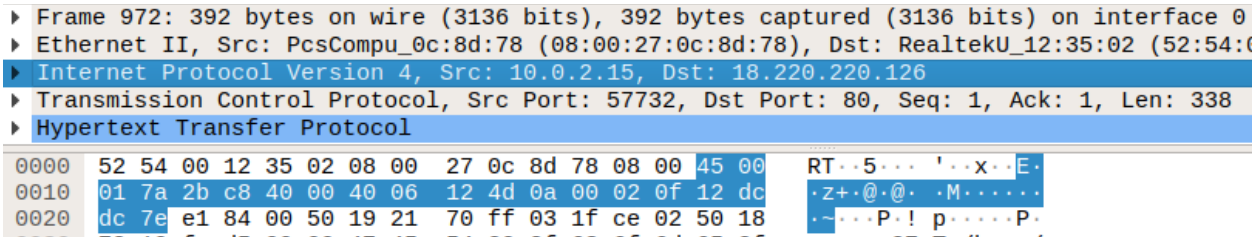




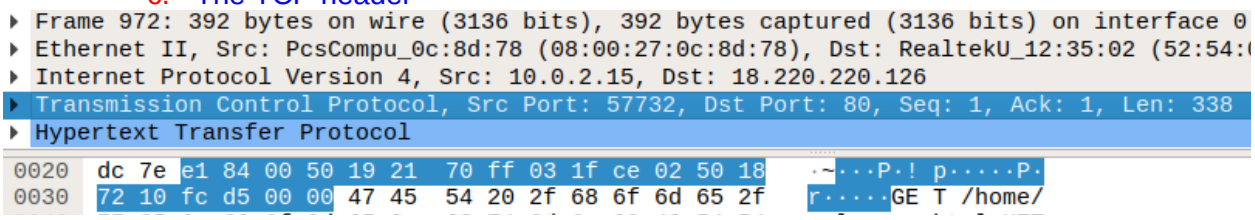
a. The link layer header



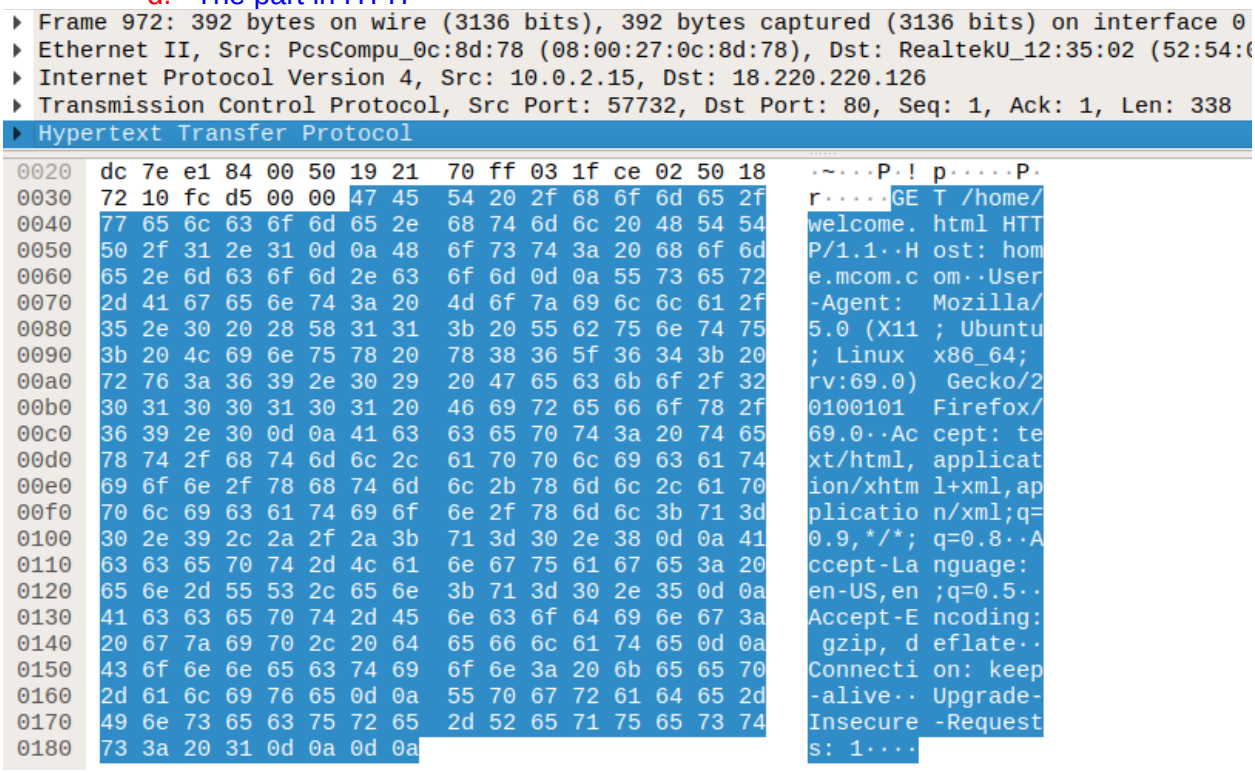
b. The IP header




c. The TCP header




d. The part in HTTP




**Task 10.** Compile and run the four pieces of code in the first lecture about **0)** UDP send, **1)** UDP receive, **2)** TCP send, **3)** TCP receive. A more detailed version is attached in the appendix of this document. Please scroll down to find it. Also the .c files are uploaded on Canvas so that you do not need to copy and paste them from the lecture slides or from this document. Note that you can run the sender and the receiver on the same machine, but you need to open two different terminals, one for the sender and the other for the receiver.

Be careful with the IP address and the port number in the code, make sure that the sender is sending to "127.0.0.1" (which is localhost) and the receiver is receiving from the same address or ANY address. You should also run the receiver first so that it waits for the sender to connect. Take

a screenshot and briefly explain what happens. **4)** Can you catch the packets send by these programs with wireshark? Why? (Hint: Think about the loopback interface.)

0) UDP client/sender:

```
msneberger@Ubuntu-VirtualBox:~$ cd CSE434/
msneberger@Ubuntu-VirtualBox:~/CSE434$ ./udp_client
UDP Test From Second Terminal.
Due to the code, these messages may only be 1024 characters long.
```

1) UDP server/receiver:

```
msneberger@Ubuntu-VirtualBox:~/CSE434$ ./udp_server
[0.0.0.0] UDP Test From Second Terminal.
[127.0.0.1] Due to the code, these messages may only be 1024 characters long.
```

2) TCP client/sender:

```
msneberger@Ubuntu-VirtualBox:~$ cd CSE434/
msneberger@Ubuntu-VirtualBox:~/CSE434$ ./tcp_client
TCP Test Message From Second Terminal
Due to the code these messages may only be 1024 characters long.
```

3) TCP server/receiver:

```
msneberger@Ubuntu-VirtualBox:~/CSE434$ ./tcp_server
waiting for connection...
connection accept from 127.0.0.1.
waiting for connection...
[connfd-4] worker thread started.
[connfd-4] TCP Test Message From Second Terminal
[connfd-4] Due to the code these messages may only be 1024 characters long.
[connfd-4] recv() error: Connection reset by peer.
```

4) You cannot look at these packets with Wireshark as they are using the lo or loopback interface which means the packets are not going from one machine to another.

**Task 11.** Answer the following questions (no hands-on required):

(a) A noiseless 4 kHz channel is sampled every 1 msec and each sample is either a 0 or 1. **0)** What is the maximum data rate? **1)** How does the maximum data rate change if the channel is noisy, with a signal-to-noise ratio of 30 dB?

**0)** base max data rate = $2B \log_2 V$ bits/second where B = samples per second, V = 2
base max data rate = $2 * 1000\text{Hz} * (\log_2 2)$ = **2,000 bits/sec = 2kbs**

*Note: this is low because we were given a low value for V. If we were not given a value for V the potential max data rate would be infinite.*
*Also note: since the sample rate is less than the potential band width, the sample rate effectively throttles the bandwidth and the bandwidth becomes superfluous. If we had not been given a sample rate the potential rate would have been 8,000 bits.sec.*

**1)** max data rate = $B \log_2(1 + \text{SNR}/10)$ where SNR = $(10 * \log_{10}\text{SNR})/10$
Max data rate = $2000\text{Hz} * \log_2(1 + (10 \log_2(1 + 1000))$ = **19934 bits/sec ≈ 19.9kbs**

*Note: if we would not have been given the 1 msec sample rate in section 0) above the noiseless max data rate would have been 8000 bits/sec, but since that sample rate overrode the 4kHz bandwidth that was available with no noise, I have used 2000 bits/sec noiseless rate to plug into Shannon's formula for the 30dB of noise rate.*

(b) A CDMA receiver gets the following chips: (-1 +1 -3 +1 -1 -3 +1 +1). The chip sequences of each station is shown as follows.

| | | | |
|---|---|---|---|
| A = | ( -1 -1 -1 +1 +1 -1 +1 +1 ) | A = | ( -1 -1 -1 +1 +1 -1 +1 +1 ) = sent a 1 |
| B = | ( -1 -1 +1 -1 +1 +1 +1 -1 ) | ¬ B = | (+1 +1 -1 +1 -1 -1 -1 +1 ) = sent a 0 |
| C = | ( -1 +1 -1 +1 +1 +1 -1 -1 ) | none C = | ( -1 +1 -1 +1 +1 +1 -1 -1 ) = silent |
| D = | ( -1 +1 -1 -1 -1 -1 +1 -1 ) | D = | ( -1 +1 -1 -1 -1 -1 +1 -1 ) = sent a 1 |
| | | A¬BD = | ( -1 +1 -3 +1 -1 -3 +1 +1) = received |

Which stations transmitted, and which bits did each one send?

Station A transmitted a 1
Station B transmitted a 0
Station C did not transmit
Station D transmitted a 1