ECE435: Network Engineering – Homework 9

Link Layer and Ethernet

Due: Friday, 12 April 2024, 5:00pm

For this homework short answers will suffice.

To submit, create a document with your answers (text, pdf, libreoffice, MS Office if you must) and e-mail them to *vincent.weaver@maine.edu* by the homework deadline. Title your e-mail "ECE435 Homework 9" and be sure your name is included in the document.

1. **Ethernet Header** (4pts)

   From previous homeworks, we used tcpdump to gather that output of our client program doing a simple web request. `sudo tcpdump port 80 -xe -i eth0 -XX`

```
0x0000:  0013 3b10 667f b827 ebaf 3711 0800 4500  ..;.f..'..7...E.
0x0010:  0038 572a 4000 4006 69cc c0a8 0833 826f  .8W*@.@.i....3.o
0x0020:  2e7f bda5 0050 cdc4 6a49 3c7b 6ca5 8018  .....P..jI<{l...
0x0030:  00e5 79f4 0000 0101 080a 0104 3e58 34a8  ..y.........>X4.
0x0040:  7bc3 4745 540a                           {.GET.
```

   (a) We now have enough knowledge to decode the entire frame. Use classnotes or the IEEE 802.3 specification to debug the Ethernet fields. Note that tcpdump is not able to capture the preamble/SFD or FCS (checksum) values.

BEGIN Ethernet HEADER	Name of Field	Decoded Value
`0x0000:` 0013 3b10 667f		
`0x0006:` b827 ebaf 3711		
`0x000c:` 0800		

   (b) Who owns the OUI of the MAC addresses in the source and destination? You can use a tool such as `https://www.wireshark.org/tools/oui-lookup.html` to find this info.

   (c) In an earlier homework we decoded the IPv4 part of this frame and saw its eventual destination is 130.111.46.127. This is not on the local network. Is the destination MAC address that of 130.111.46.127? If not, what machine does the MAC address correspond to?

2. **Protocol Mystery** (2pts)

   You run tcpdump and you see packets like this.

```
16:15:57.734294 Ethernet (len 6), IPv4 (len 4),
               Request who-has macbook-air tell a10, length 46
16:15:57.734355 Ethernet (len 6), IPv4 (len 4),
                Reply macbook-air is-at 00:50:b6:47:1c:de (oui Unknown), length 28
16:16:00.408107 Ethernet (len 6), IPv4 (len 4),
                Request who-has atom tell macbook-air, length 28
16:16:00.408315 Ethernet (len 6), IPv4 (len 4),
                Reply atom is-at 00:22:4d:9f:d4:fc (oui Unknown), length 46
```

   (a) What protocol is this?

   (b) What is it used for?

3. **Answer the following questions** (2pts)

   (a) Why did Ethernet win out over TokenRing?

   (b) Why is the minimum size of an Ethernet frame 64 bytes?

   (c) Why is the maximum size of an Ethernet frame 1500 bytes?

   (d) What does your Ethernet card do to a frame if it calculates an invalid CRC?

4. **Investigating an Ethernet Interface** (2pts)

   If you have access to a Linux machine with a wired ethernet connection, we will gather some info on your Ethernet device. You should know enough now to interpret the results.

   If you don't have a wired Linux interface, look at the last two pages of this document, I've provided some sample results you can use instead.

   The traditional way of gathering info on the network interface was the command
   `/sbin/ifconfig eth0`
   where eth0 was traditionally the first ethernet interface in the system (depending on your Linux distibution a much more complicated name might be used for the interface).

   These days `ifconfig` has been deprecated and might not be installed by default. The replacement is the `ip` tool.

   You can get similar results to `ifconfig` by the commands `ip addr` and `ip -s addr`

   To find out lower-level info specific to your ethernet card you can run
   `/sbin/ethtool eth0`
   (note you might have to install ethtool, it might not be there by default).

   You can also find out about your card by looking at the boot messages
   `sudo dmesg | grep eth`

   Answer the following questions, either using info you gathered yourself or else from the results provided on the following pages:

   (a) What is the MAC address?

   (b) Look up the MAC owner using an OUI lookup tool. If it's your own device, does the OUI match what you'd expect?

   (c) What is the default frame size (MTU)?

   (d) How many bytes have been received (RX)?

   (e) How many bytes have been transmitted (TX)?

   (f) Has your device seen any collisions?

   (g) Has your device dropped any packets?

   (h) If the collision count is low, can you explain why that is?

   (i) What speed is your device running at?

   (j) Are you connected to a switch rather than a hub? How can you tell?

```
/sbin/ifconfig eth0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 130.111.218.24  netmask 255.255.254.0  broadcast 130.111.219.255
        inet6 fe80::225:90ff:fee3:5734  prefixlen 64  scopeid 0x20<link>
        ether 00:25:90:e3:57:34  txqueuelen 1000  (Ethernet)
        RX packets 359075559  bytes 76137884844 (70.9 GiB)
        RX errors 0  dropped 0  overruns 13216  frame 0
        TX packets 74498857  bytes 18280085535 (17.0 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device memory 0xfe120000-fe13ffff
```

```
ip addr

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group def
    link/ether 00:25:90:e3:57:34 brd ff:ff:ff:ff:ff:ff
    inet 130.111.218.24/23 brd 130.111.219.255 scope global dynamic eth0
       valid_lft 34690sec preferred_lft 34690sec
    inet6 fe80::225:90ff:fee3:5734/64 scope link
       valid_lft forever preferred_lft forever
```

```
ip -s addr

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group def
    link/ether 00:25:90:e3:57:34 brd ff:ff:ff:ff:ff:ff
    inet 130.111.218.24/23 brd 130.111.219.255 scope global dynamic eth0
       valid_lft 34657sec preferred_lft 34657sec
    inet6 fe80::225:90ff:fee3:5734/64 scope link
       valid_lft forever preferred_lft forever
    RX: bytes  packets  errors  dropped overrun mcast
    76139606874 359098268 0       0       0       98966998
    TX: bytes  packets  errors  dropped carrier collsns
    18280800017 74503021 0       0       0       0
```

```
/sbin/ethtool eth0

Settings for eth0:
 Supported ports: [ TP ]
 Supported link modes:   10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
 Supported pause frame use: Symmetric
 Supports auto-negotiation: Yes
 Supported FEC modes: Not reported
 Advertised link modes:  10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
 Advertised pause frame use: Symmetric
 Advertised auto-negotiation: Yes
 Advertised FEC modes: Not reported
 Speed: 1000Mb/s
 Duplex: Full
 Auto-negotiation: on
 Port: Twisted Pair
 PHYAD: 1
 Transceiver: internal
 MDI-X: off (auto)
 Supports Wake-on: pumbg
 Wake-on: g
        Current message level: 0x00000007 (7)
                               drv probe link
 Link detected: yes
```

```
sudo dmesg | grep eth

[    5.289165] igb 0000:05:00.0: added PHC on eth0
[    5.289206] igb 0000:05:00.0: eth0: (PCIe:5.0Gb/s:Width x2)
       00:25:90:e3:57:34
[    5.289294] igb 0000:05:00.0: eth0: PBA No: 106100-000
[    5.348648] igb 0000:05:00.1: added PHC on eth1
[    5.348689] igb 0000:05:00.1: eth1: (PCIe:5.0Gb/s:Width x2)
       00:25:90:e3:57:35
[    5.348780] igb 0000:05:00.1: eth1: PBA No: 106100-000
[   31.353051] igb 0000:05:00.1 eth1: igb: eth1 NIC Link is Up
       1000 Mbps Full Duplex, Flow Control: RX/TX
[   31.353294] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
[   36.757030] igb 0000:05:00.0 eth0: igb: eth0 NIC Link is Up
       1000 Mbps Full Duplex, Flow Control: RX
[   36.757240] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```