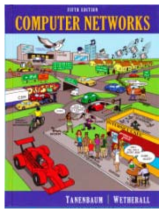
Lab Exercise – 802.11



Objective

To explore the physical layer, link layer, and management functions of 802.11. It is widely used to wireless connect mobile devices to the Internet, and covered in §4.4 of your text. Review that section first.

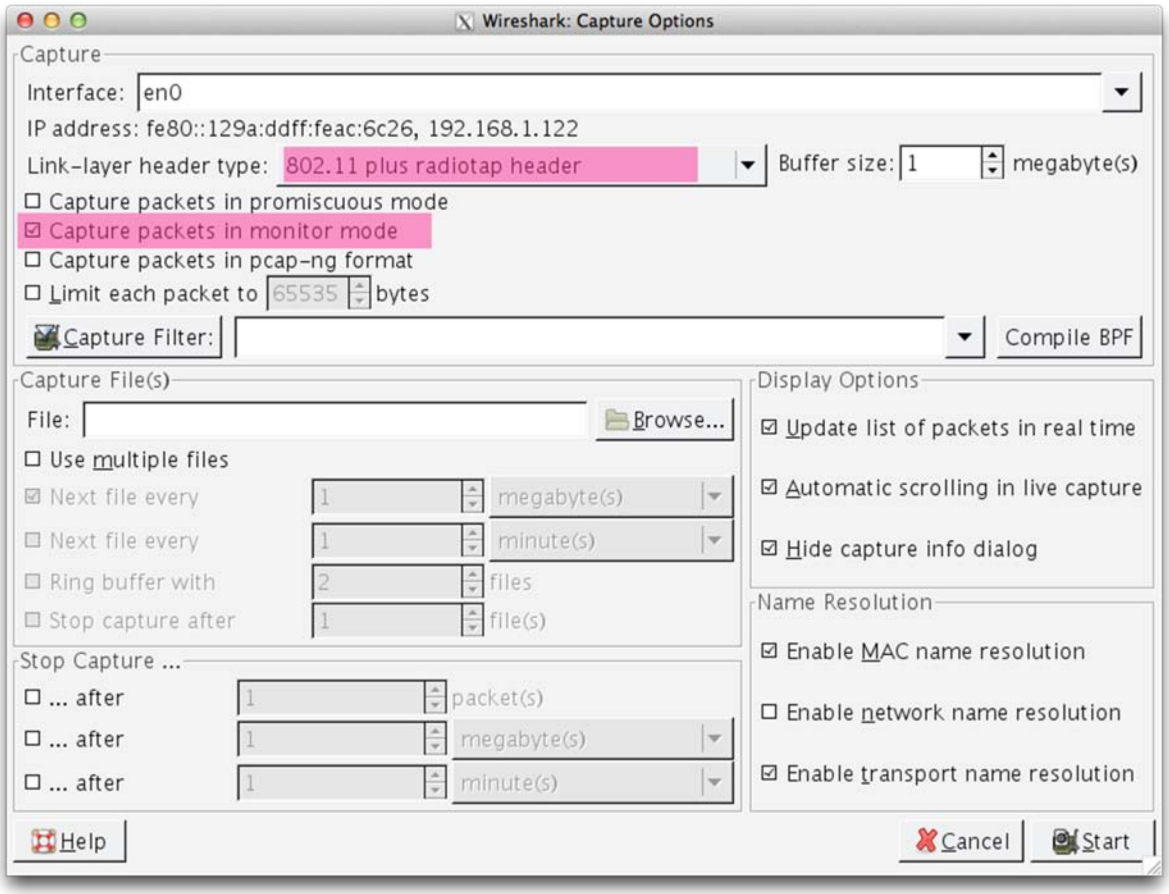
Requirements

Wireshark: This lab uses the Wireshark software tool to capture and examine a packet trace. A packet trace is a record of traffic at a location on the network, as if a snapshot was taken of all the bits that passed across a particular wire. The packet trace records a timestamp for each packet, along with the bits that make up the packet, from the lower-layer headers to the higher-layer contents. Wireshark runs on most operating systems, including Windows, Mac and Linux. It provides a graphical UI that shows the sequence of packets and the meaning of the bits when interpreted as protocol headers and data. It color-codes packets by their type, and has various ways to filter and analyze packets to let you investigate the behavior of network protocols. Wireshark is widely used to troubleshoot networks. You can download it from www.wireshark.org if it is not already installed on your computer. We highly recommend that you watch the short, 5 minute video "Introduction to Wireshark" that is on the site.

Step 1: Fetch a Trace

We provide a trace that you can use by starting Wireshark and selecting Open from the File menu. On Windows/Mac, you may locate the trace file and open it directly to launch Wireshark with the trace. You can now proceed to Step 2; the rest of this section is informational.

Unlike for the other labs, it may be difficult to gather your own trace, for several reasons. The main issue is that Windows lacks driver support to gather 802.11 frames for most wireless NICs. When we captured traffic previously, the operating system made it appear to come via a wired Ethernet (even if it actually came via a wireless network) and discarded any 802.11 frames without a higher layer data payload (such as Acknowledgements). On some systems, typically Mac and Linux, it is possible to tell the operating system to gather 802.11 frames directly, without this conversion. This is called "Monitor mode". If your system supports it, then the Wireshark capture options for your wireless interface will allow you to select Monitor mode, and to set the format of captured traffic to "802.11 plus radiotap header" rather than Ethernet. An example is shown below. If there is no way to select Monitor mode then your system likely cannot capture 802.11.



Capture
Interface: en0
IP address: fe80::129a:ddff:feac:6c26, 192.168.1.122
Link-layer header type: 802.11 plus radiotap header
Buffer size: 1 megabyte(s)
☐ Capture packets in promiscuous mode
☑ Capture packets in monitor mode
☐ Capture packets in pcap-ng format
☐ Limit each packet to 65535 bytes
Capture Filter:
Compile BPF

Capture File(s)
File:
Browse...
☐ Use multiple files
☑ Next file every 1 megabyte(s)
☐ Next file every 1 minute(s)
☐ Ring buffer with 2 files
☐ Stop capture after 1 file(s)

Stop Capture ...
☐ ... after 1 packet(s)
☐ ... after 1 megabyte(s)
☐ ... after 1 minute(s)

Display Options
☑ Update list of packets in real time
☑ Automatic scrolling in live capture
☑ Hide capture info dialog

Name Resolution
☑ Enable MAC name resolution
☐ Enable network name resolution
☑ Enable transport name resolution

Help
Cancel
Start

Figure 1: Capturing a wireless trace with Monitor mode (Mac)

A second difficulty is that when an interface captures wireless traffic in monitor mode, it is often not available for regular use. This means that you need at least two computers: one computer to send test traffic and a second monitor computer to capture a trace of wireless activity.

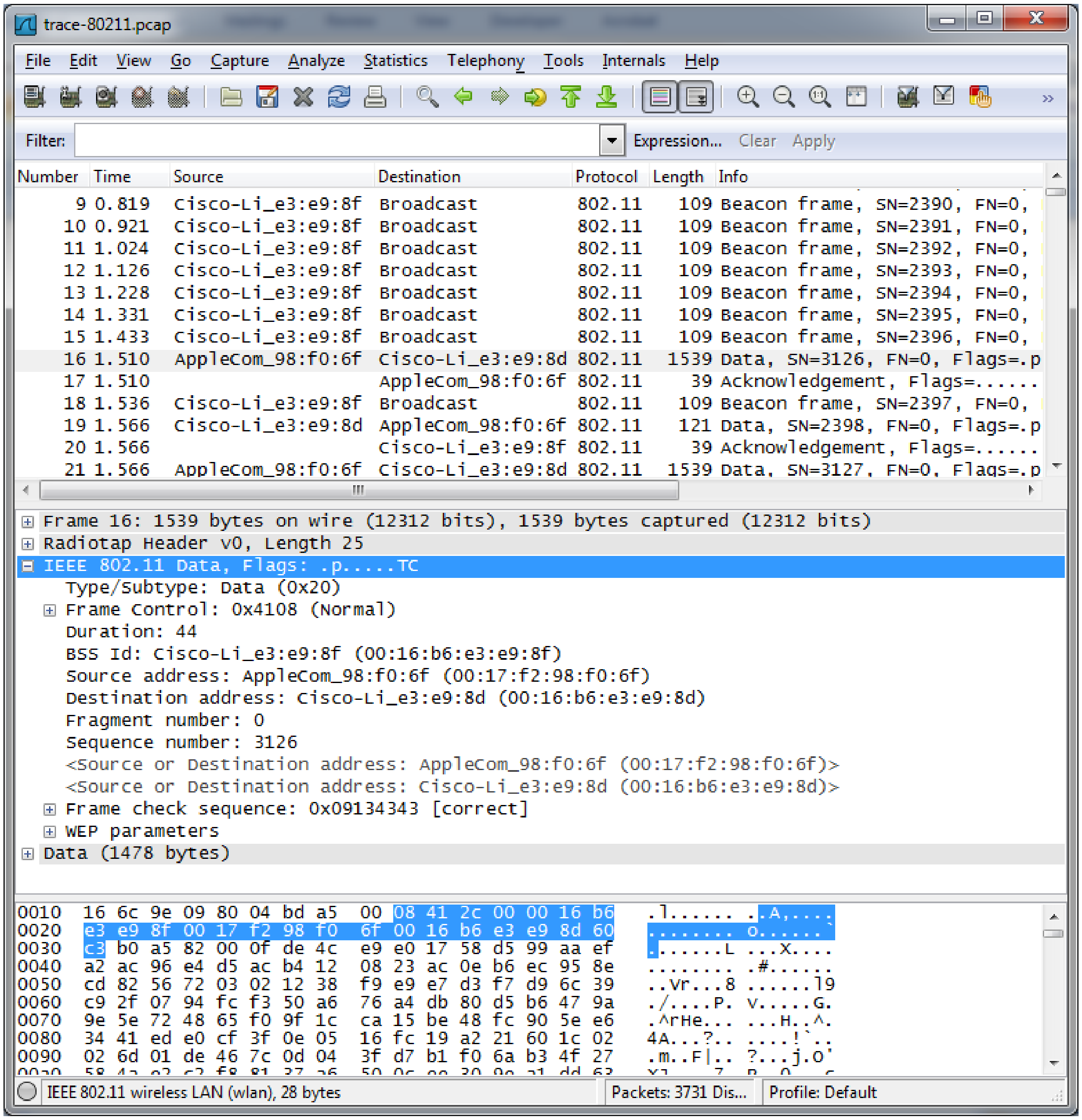
Finally, note that capturing a trace in monitor mode will record all wireless activity in the vicinity. Since 802.11 wireless devices are pervasive, it is likely that your trace will capture unwanted traffic from other nearby computers. This behavior makes it difficult to cleanly observe your own traffic.

If you can handle these difficulties, you can gather your own wireless trace to do this lab.

Step 2: Inspect the Trace

To begin, we will take a look at the format of an 802.11 frame. There are many different kinds of 802.11 frames that will be captured in a trace; the Info field describes the type, such as Beacon, Data, and Acknowledgement. We will inspect a Data frame, which carries packets across 802.11 networks.

Find a Data frame in the trace and select it. Wireshark will let us select a frame (from the top panel) and view its protocol layers, in terms of both header fields (in the middle panel) and the bytes that make up the frame (in the bottom panel). You can do this simply by scrolling down until you find one, or by clicking on the Info column to sort by that key and then scrolling to the Data portion of the trace. We have selected a Data frame in the figure below.



```
Number Time  Source              Destination         Protocol Length Info
     9 0.819 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2390, FN=0,
    10 0.921 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2391, FN=0,
    11 1.024 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2392, FN=0,
    12 1.126 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2393, FN=0,
    13 1.228 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2394, FN=0,
    14 1.331 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2395, FN=0,
    15 1.433 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2396, FN=0,
    16 1.510 AppleCom_98:f0:6f   Cisco-Li_e3:e9:8d   802.11   1539 Data, SN=3126, FN=0, Flags=.p
    17 1.510                     AppleCom_98:f0:6f   802.11     39 Acknowledgement, Flags=......
    18 1.536 Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2397, FN=0,
    19 1.566 Cisco-Li_e3:e9:8d   AppleCom_98:f0:6f   802.11    121 Data, SN=2398, FN=0, Flags=.p
    20 1.566                     Cisco-Li_e3:e9:8f   802.11     39 Acknowledgement, Flags=......
    21 1.566 AppleCom_98:f0:6f   Cisco-Li_e3:e9:8d   802.11   1539 Data, SN=3127, FN=0, Flags=.p
```

```
⊞ Frame 16: 1539 bytes on wire (12312 bits), 1539 bytes captured (12312 bits)
⊞ Radiotap Header v0, Length 25
⊟ IEEE 802.11 Data, Flags: .p.....TC
    Type/Subtype: Data (0x20)
  ⊞ Frame Control: 0x4108 (Normal)
    Duration: 44
    BSS Id: Cisco-Li_e3:e9:8f (00:16:b6:e3:e9:8f)
    Source address: AppleCom_98:f0:6f (00:17:f2:98:f0:6f)
    Destination address: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
    Fragment number: 0
    Sequence number: 3126
    <Source or Destination address: AppleCom_98:f0:6f (00:17:f2:98:f0:6f)>
    <Source or Destination address: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)>
  ⊞ Frame check sequence: 0x09134343 [correct]
  ⊞ WEP parameters
⊞ Data (1478 bytes)
```

```
0010  16 6c 9e 09 80 04 bd a5  00 08 41 2c 00 00 16 b6   .l...... ..A,....
0020  e3 e9 8f 00 17 f2 98 f0  6f 00 16 b6 e3 e9 8d 60   ........ o......`
0030  c3 b0 a5 82 00 0f de 4c  e9 e0 17 58 d5 99 aa ef   .......L ...X....
0040  a2 ac 96 e4 d5 ac b4 12  08 23 ac 0e b6 ec 95 8e   ........ .#......
0050  cd 82 56 72 03 02 12 38  f9 e9 e7 d3 f7 d9 6c 39   ..Vr...8 ......l9
0060  c9 2f 07 94 fc f3 50 a6  76 a4 db 80 d5 b6 47 9a   ./....P. v.....G.
0070  9e 5e 72 48 65 f0 9f 1c  ca 15 be 48 fc 90 5e e6   .^rHe... ...H..^.
0080  34 41 ed e0 cf 3f 0e 05  16 fc 19 a2 21 60 1c 02   4A...?.. ....!`..
0090  02 6d 01 de 46 7c 0d 04  3f d7 b1 f0 6a b3 4f 27   .m..F|.. ?...j.O'
```

IEEE 802.11 wireless LAN (wlan), 28 bytes | Packets: 3731 Dis... | Profile: Default

Figure 2: Inspecting an 802.11 Data frame

Inspect the protocol layers recorded with the frame for these protocols. Look in the middle panel.

- Frame is a record added by Wireshark with information about the time and length of the frame; it does not capture bits that were sent “over the air”.
- Radiotap is also a record created by Wireshark to capture physical layer parameters, such as the strength of the signal and the modulation. Skip this record for now; we will investigate it later.

- IEEE 802.11 is the bits of the 802.11 Data frame. This is the record we are looking for, and we will go into its details shortly. It is selected and expanded in the figure so that you can see the internal fields (in the middle panel) and the portion of the frame it occupies (highlighted in the lower panel, and identified at bottom as 28 bytes long).
- Data is a record containing the frame payload data, i.e., that has higher-layer protocols such as LLC, IP packets, etc. Alternatively you may see the higher-layer protocols themselves.

If Wireshark can understand the contents of the Data frame payload then it will create protocol records for them. However, in many wireless settings (such as the sample trace) the payload contents are encrypted and simply appear as one record. All frames are then listed as protocol 802.11, rather than higher layer protocols such as TCP. It is possible to tell Wireshark the wireless network key and have it decrypt the payloads. However, we will skip that step since our interest is the 802.11 headers.

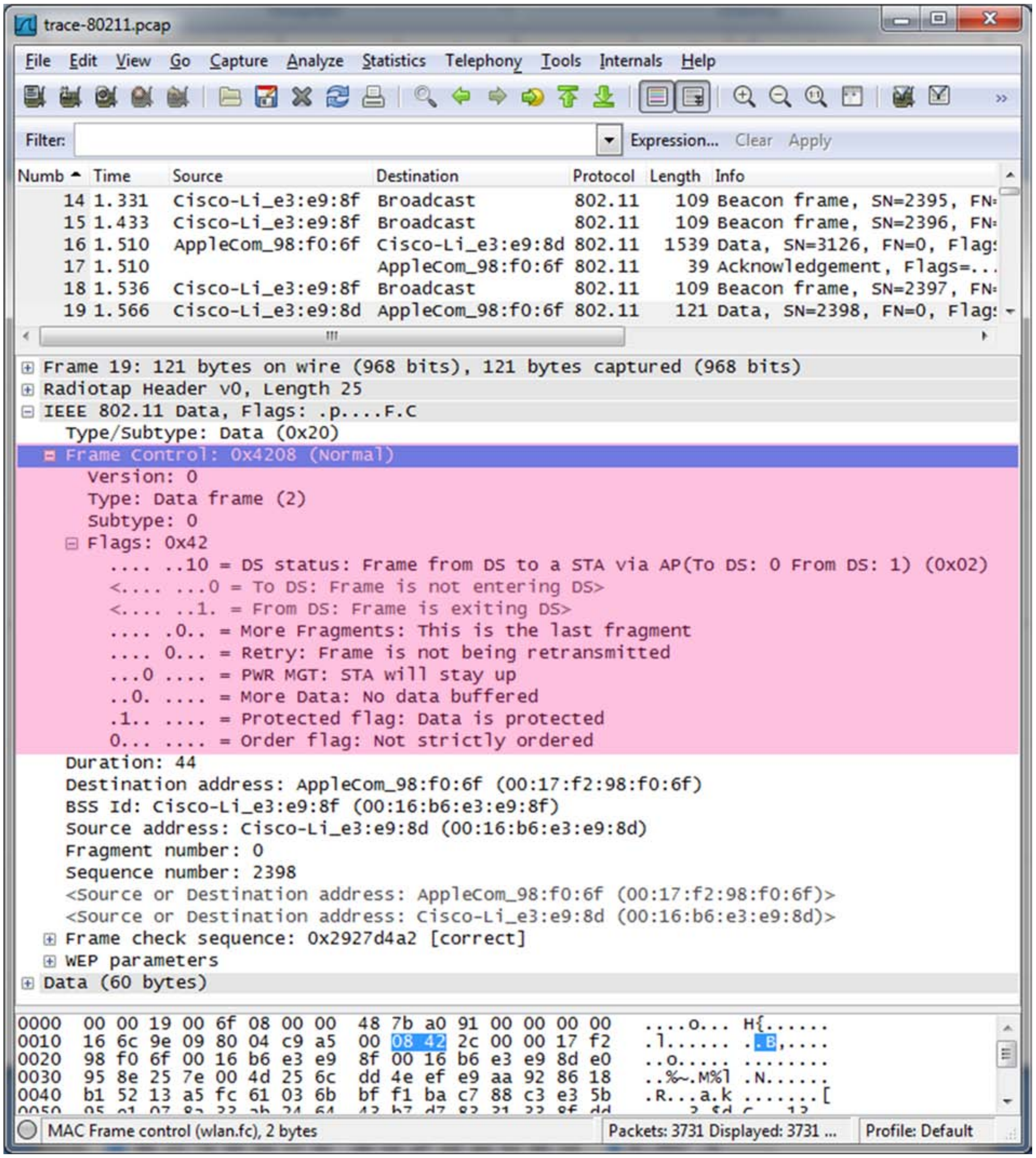
Expand the IEEE 802.11 record of the Data frame and inspect the details of the various header fields. You can expand this block using the "+" expander or icon; it is shown expanded in our figure. To inspect the fields, you may compare them with Fig. 4-29. The fields in Wireshark are:

- Frame Control . It encodes the frame Type and Subtype, e.g., Data, as well as various flags. We will look at these fields in more detail shortly.
- Duration. This field tells computers how much time is needed on the wireless medium for additional packets that are part of this exchange.
- BSS identifier, source address, and destination address, in an order that depends on the specifics of the Data frame. These address fields identify who transmitted the packet and who should receive it. The BSS identifier is the address of the wireless access point.
- Fragment and sequence number. These fields number the frame for reassembly and retransmission, if needed. The sequence number is incremented with each new transmission.
- Frame check sequence. This is a CRC over the frame. It comes at the end (click it and you will see its position in the frame) but is listed with the other 802.11 header fields for convenience.
- There may also be a WEP or WPA2 field with security parameters in the case that the frame payload is encrypted. We are not delving into wireless security here, so you can ignore that field.

Finally, expand the Frame Control field and look at it in detail, including the Flags that you find within it. All 802.11 frames begin with a Frame Control field, and the details of the subfields and flags determine the format of the rest of the message; it may be like the Data frame we explored above or very different such as an Ack frame we will look at later. The subfields are:

- Version, with a value of zero for the current version.
- Type and Subtype specify the type of frame, e.g., Data or Ack.
- To DS. This flag is set if the frame is sent from a computer to the wired network via the AP.
- From DS. This flag is set if the frame is sent from the wired network to a computer via the AP.
- More fragments. Set if there are more frames in this message.
- Retry. Set if the frame is a retransmission.
- Power management. Set if the sender will go into power-save sleep after transmission.
- More data. Set if the sender has more frames to send.
- Protected. Set if the frame is encrypted with WEP/WPA2.

- Order. Set if the receiver must keep the frames in order.



trace-80211.pcap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
Numb  Time   Source              Destination         Protocol Length Info
   14 1.331  Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2395, FN=
   15 1.433  Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2396, FN=
   16 1.510  AppleCom_98:f0:6f   Cisco-Li_e3:e9:8d   802.11   1539 Data, SN=3126, FN=0, Flag:
   17 1.510                      AppleCom_98:f0:6f   802.11     39 Acknowledgement, Flags=...
   18 1.536  Cisco-Li_e3:e9:8f   Broadcast           802.11    109 Beacon frame, SN=2397, FN=
   19 1.566  Cisco-Li_e3:e9:8d   AppleCom_98:f0:6f   802.11    121 Data, SN=2398, FN=0, Flag:

⊞ Frame 19: 121 bytes on wire (968 bits), 121 bytes captured (968 bits)
⊞ Radiotap Header v0, Length 25
⊟ IEEE 802.11 Data, Flags: .p....F.C
    Type/Subtype: Data (0x20)
  ⊟ Frame Control: 0x4208 (Normal)
      Version: 0
      Type: Data frame (2)
      Subtype: 0
    ⊟ Flags: 0x42
        .... ..10 = DS status: Frame from DS to a STA via AP(To DS: 0 From DS: 1) (0x02)
        <.... ...0 = To DS: Frame is not entering DS>
        <.... ..1. = From DS: Frame is exiting DS>
        .... .0.. = More Fragments: This is the last fragment
        .... 0... = Retry: Frame is not being retransmitted
        ...0 .... = PWR MGT: STA will stay up
        ..0. .... = More Data: No data buffered
        .1.. .... = Protected flag: Data is protected
        0... .... = Order flag: Not strictly ordered
    Duration: 44
    Destination address: AppleCom_98:f0:6f (00:17:f2:98:f0:6f)
    BSS Id: Cisco-Li_e3:e9:8f (00:16:b6:e3:e9:8f)
    Source address: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
    Fragment number: 0
    Sequence number: 2398
    <Source or Destination address: AppleCom_98:f0:6f (00:17:f2:98:f0:6f)>
    <Source or Destination address: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)>
  ⊞ Frame check sequence: 0x2927d4a2 [correct]
  ⊞ WEP parameters
⊞ Data (60 bytes)

0000  00 00 19 00 6f 08 00 00  48 7b a0 91 00 00 00 00   ....o... H{......
0010  16 6c 9e 09 80 04 c9 a5  00 08 42 2c 00 00 17 f2   .l...... .B,.....
0020  98 f0 6f 00 16 b6 e3 e9  8f 00 16 b6 e3 e9 8d e0   ..o..... ........
0030  95 8e 25 7e 00 4d 25 6c  dd 4e ef e9 aa 92 86 18   ..%~.M%l .N......
0040  b1 52 13 a5 fc 61 03 6b  bf f1 ba c7 88 c3 e3 5b   .R...a.k .......[

MAC Frame control (wlan.fc), 2 bytes    Packets: 3731 Displayed: 3731 ...    Profile: Default
```

Figure3: Expanded view of the Frame Control fields and Flags

Different computers may use these flags differently depending on how they implement 802.11. For example, some computers may make heavy use of power-save or encryption features while others may not. Combined with the fact that there are dozens of different types of frames, this means that you will see all sorts of wireless traffic in most traces. Explore a bit if you are curious!

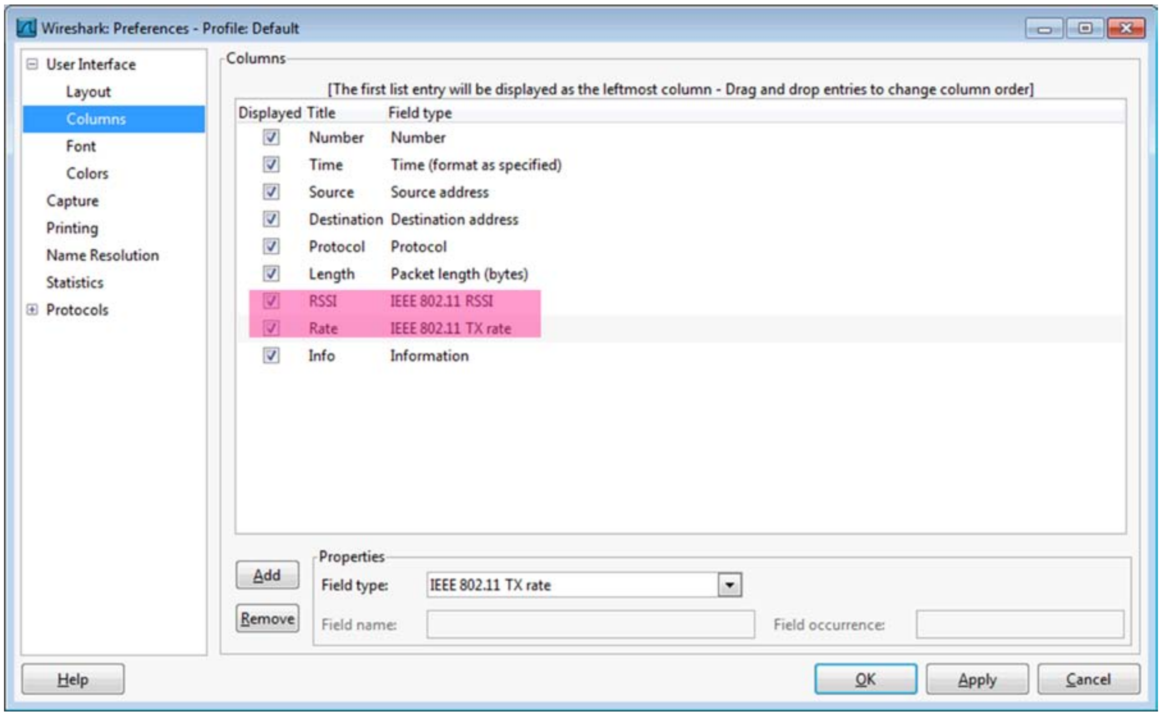
Step 3: 802.11 Physical Layer

Now that we have some familiarity with 802.11 Data frames, we will take a closer look at different parts of the wireless system, starting with the physical layer. At the lowest layer, sending and receiving messages is all about the frequency band, modulation, the signal-to-noise ratio with which the signal is received. We can look at all of these factors using information in the Radiotap header!

Answer the numbered questions in this step to explore the physical layer aspects, beginning with frequency. The frequency or channel is the same for all frames in the trace, since the wireless network interface is set to listen on a fixed frequency.

1. *What is the channel frequency?* To find the frequency, expand the Radiotap header of any frame and look for the Channel frequency.

To look at the modulation we can observe the Data Rate value, and to look at the SNR we can observe the SSI Signal value (combined with the SSI Noise value). The SSI Signal value is more commonly known as the RSSI (Received Signal Strength Indication). These fields will vary with different frames. To see them, first we must add new columns to the main display.



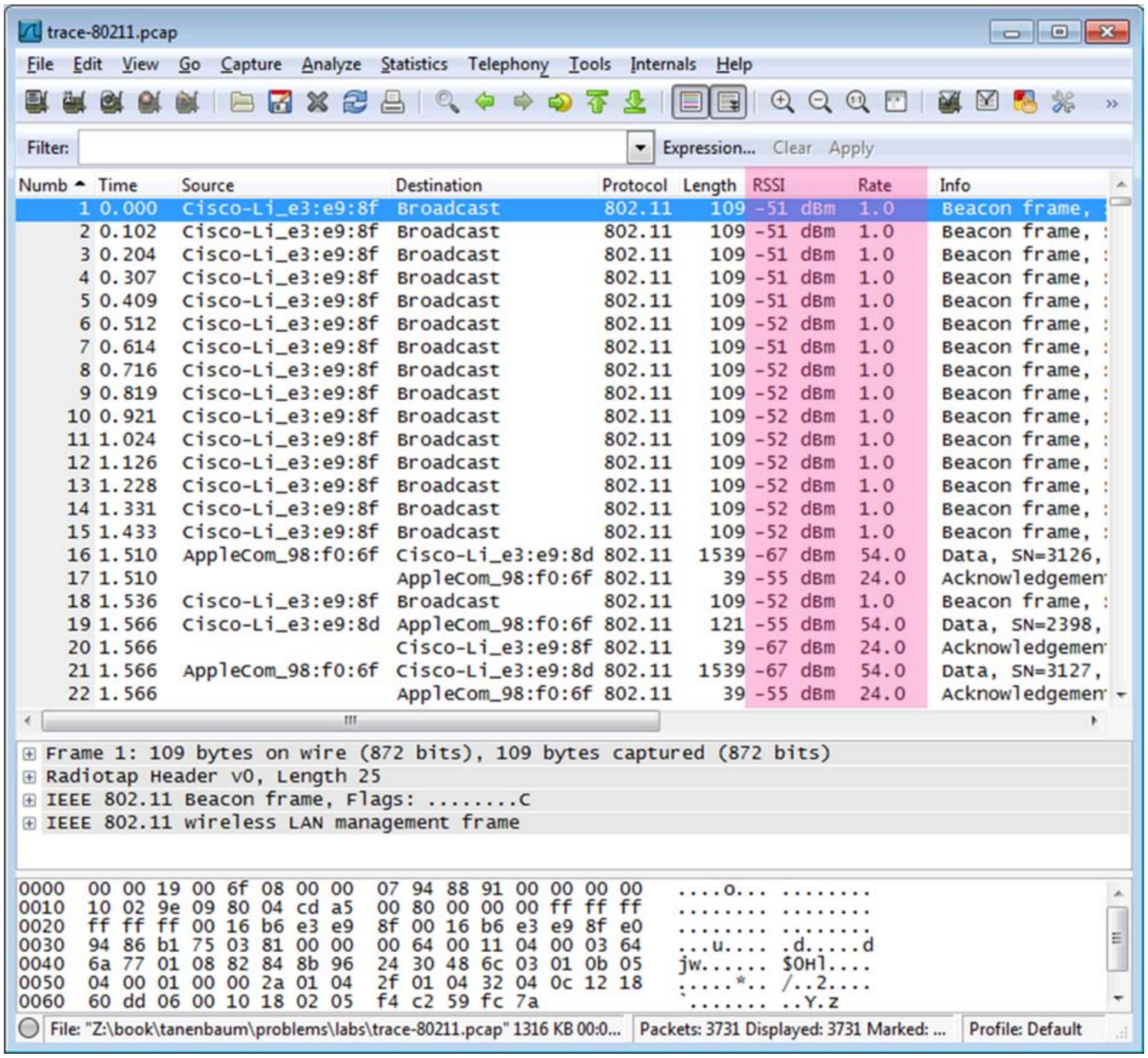
[The first list entry will be displayed as the leftmost column - Drag and drop entries to change column order]

Displayed	Title	Field type
✓	Number	Number
✓	Time	Time (format as specified)
✓	Source	Source address
✓	Destination	Destination address
✓	Protocol	Protocol
✓	Length	Packet length (bytes)
✓	RSSI	IEEE 802.11 RSSI
✓	Rate	IEEE 802.11 TX rate
✓	Info	Information

Field type: IEEE 802.11 TX rate
Field name:
Field occurrence:

Figure 2: Adding columns for RSSI and Rate

Add two new display columns for the TX Rate (or Data Rate) and RSSI (or SSI Signal value) by going to the Preferences panel (under the Edit menu) and selecting Columns (by expanding the User Interface block). The columns in our figure are called Rate, with a field of type IEEE 802.11 TX Rate, and RSSI, with a field type of IEEE 802.11 RSSI. You may reorder the columns so that these columns are to the left of Info for visibility. When you return to the main display you will have Rate and RSSI information for each frame.


trace-80211.pcap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
Numb Time  Source            Destination       Protocol Length RSSI     Rate  Info
   1 0.000 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -51 dBm  1.0   Beacon frame, :
   2 0.102 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -51 dBm  1.0   Beacon frame, :
   3 0.204 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -51 dBm  1.0   Beacon frame, :
   4 0.307 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -51 dBm  1.0   Beacon frame, :
   5 0.409 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -51 dBm  1.0   Beacon frame, :
   6 0.512 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
   7 0.614 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -51 dBm  1.0   Beacon frame, :
   8 0.716 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
   9 0.819 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  10 0.921 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  11 1.024 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  12 1.126 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  13 1.228 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  14 1.331 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  15 1.433 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  16 1.510 AppleCom_98:f0:6f Cisco-Li_e3:e9:8d 802.11   1539   -67 dBm  54.0  Data, SN=3126,
  17 1.510                   AppleCom_98:f0:6f 802.11     39   -55 dBm  24.0  Acknowledgemen
  18 1.536 Cisco-Li_e3:e9:8f Broadcast         802.11    109   -52 dBm  1.0   Beacon frame, :
  19 1.566 Cisco-Li_e3:e9:8d AppleCom_98:f0:6f 802.11    121   -55 dBm  54.0  Data, SN=2398,
  20 1.566                   Cisco-Li_e3:e9:8f 802.11     39   -67 dBm  24.0  Acknowledgemen
  21 1.566 AppleCom_98:f0:6f Cisco-Li_e3:e9:8d 802.11   1539   -67 dBm  54.0  Data, SN=3127,
  22 1.566                   AppleCom_98:f0:6f 802.11     39   -55 dBm  24.0  Acknowledgemen
```

```
⊞ Frame 1: 109 bytes on wire (872 bits), 109 bytes captured (872 bits)
⊞ Radiotap Header v0, Length 25
⊞ IEEE 802.11 Beacon frame, Flags: ........C
⊞ IEEE 802.11 wireless LAN management frame
```

```
0000  00 00 19 00 6f 08 00 00  07 94 88 91 00 00 00 00   ....o... ........
0010  10 02 9e 09 80 04 cd a5  00 80 00 00 00 ff ff ff   ........ ........
0020  ff ff ff 00 16 b6 e3 e9  8f 00 16 b6 e3 e9 8f e0   ........ ........
0030  94 86 b1 75 03 81 00 00  00 64 00 11 04 00 03 64   ...u.... .d.....d
0040  6a 77 01 08 82 84 8b 96  24 30 48 6c 03 01 0b 05   jw...... $0Hl....
0050  04 00 01 00 00 2a 01 04  2f 01 04 32 04 0c 12 18   .....*.. /..2....
0060  60 dd 06 00 10 18 02 05  f4 c2 59 fc 7a            `....... ..Y.z
```

```
File: "Z:\book\tanenbaum\problems\labs\trace-80211.pcap" 1316 KB 00:0...  Packets: 3731 Displayed: 3731 Marked: ...  Profile: Default
```

Figure 3: Wireless trace showing Rate and RSSI for each frame

You should see a variety of rates. That is, unlike wired Ethernet for which frames are sent at a fixed rate (after negotiation of the kind of Ethernet), wireless rates vary depending on the conditions and capabilities of the computers.

2. *What rates are used?* Give an ordered list of rates from lowest to highest. Hint: you can click the Rate column to sort by that value.

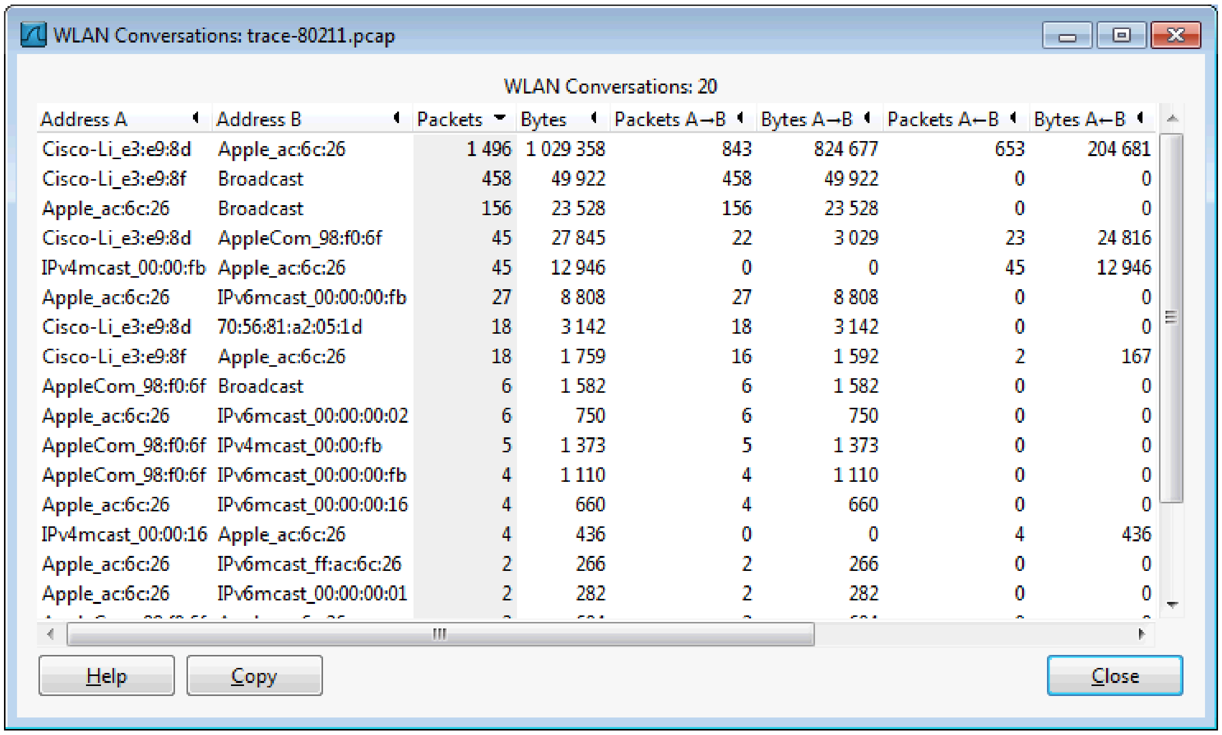
You should also see a variety of RSSI values, such as “-60 dBm”. RSSI is measured on a log scale in which 0 dBm means 1 milliWatt of power and each +10 means a factor of 10 larger and each -10 means a factor of 10 smaller. Thus -60 dBm means one million-th of 1 mW, or 10^{-9} Watts, a tiny amount of power! The SNR is the signal level relative to the noise level, a roughly fixed value given in the Radiotap header to be -90 dBm. These values add or subtract on the logarithmic scale. Thus a signal level of -60 dBm is 30 dB or a factor of 1000 larger than the noise level of -90 dBm. This means a frame with an RSSI of -60 dBm has an SNR of 30 dB. RSSIs may vary greatly, which means that some frames will have a much weaker or stronger signal than other frames. Variations of 40 dB are common, meaning that one frame may be 10,000 times weaker or stronger than another frame received by the same network interface. You should be gaining an appreciation for wireless technology!

3. *What is the range of RSSI and hence variation in SNRs in the trace?* Give this as the strongest and weakest RSSI and the dB difference between them.

Turn-in: Hand in your answers to the above questions.

Step 4: 802.11 Link Layer

Under the Statistics menu, select Conversations and WLAN (for wireless LAN, i.e., 802.11). This will pull up a window like that of the figure below which lists each pair of communicating computers. You can sort this list by size by clicking on the Packets or Bytes column headings. This view will help us further explore the trace, starting with a summary of the link layer activity.



WLAN Conversations: 20

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
Cisco-Li_e3:e9:8d	Apple_ac:6c:26	1 496	1 029 358	843	824 677	653	204 681
Cisco-Li_e3:e9:8f	Broadcast	458	49 922	458	49 922	0	0
Apple_ac:6c:26	Broadcast	156	23 528	156	23 528	0	0
Cisco-Li_e3:e9:8d	AppleCom_98:f0:6f	45	27 845	22	3 029	23	24 816
IPv4mcast_00:00:fb	Apple_ac:6c:26	45	12 946	0	0	45	12 946
Apple_ac:6c:26	IPv6mcast_00:00:00:fb	27	8 808	27	8 808	0	0
Cisco-Li_e3:e9:8d	70:56:81:a2:05:1d	18	3 142	18	3 142	0	0
Cisco-Li_e3:e9:8f	Apple_ac:6c:26	18	1 759	16	1 592	2	167
AppleCom_98:f0:6f	Broadcast	6	1 582	6	1 582	0	0
Apple_ac:6c:26	IPv6mcast_00:00:00:02	6	750	6	750	0	0
AppleCom_98:f0:6f	IPv4mcast_00:00:fb	5	1 373	5	1 373	0	0
AppleCom_98:f0:6f	IPv6mcast_00:00:00:fb	4	1 110	4	1 110	0	0
Apple_ac:6c:26	IPv6mcast_00:00:00:16	4	660	4	660	0	0
IPv4mcast_00:00:16	Apple_ac:6c:26	4	436	0	0	4	436
Apple_ac:6c:26	IPv6mcast_ff:ac:6c:26	2	266	2	266	0	0
Apple_ac:6c:26	IPv6mcast_00:00:00:01	2	282	2	282	0	0

Figure 4: 802.11 conversations ordered by size

In our trace, and likely yours, most of the activity is in a relatively small fraction of the conversations. The low activity conversations are due to background traffic from idle computers, and from a small number of packets that are occasionally captured from adjacent wireless networks.

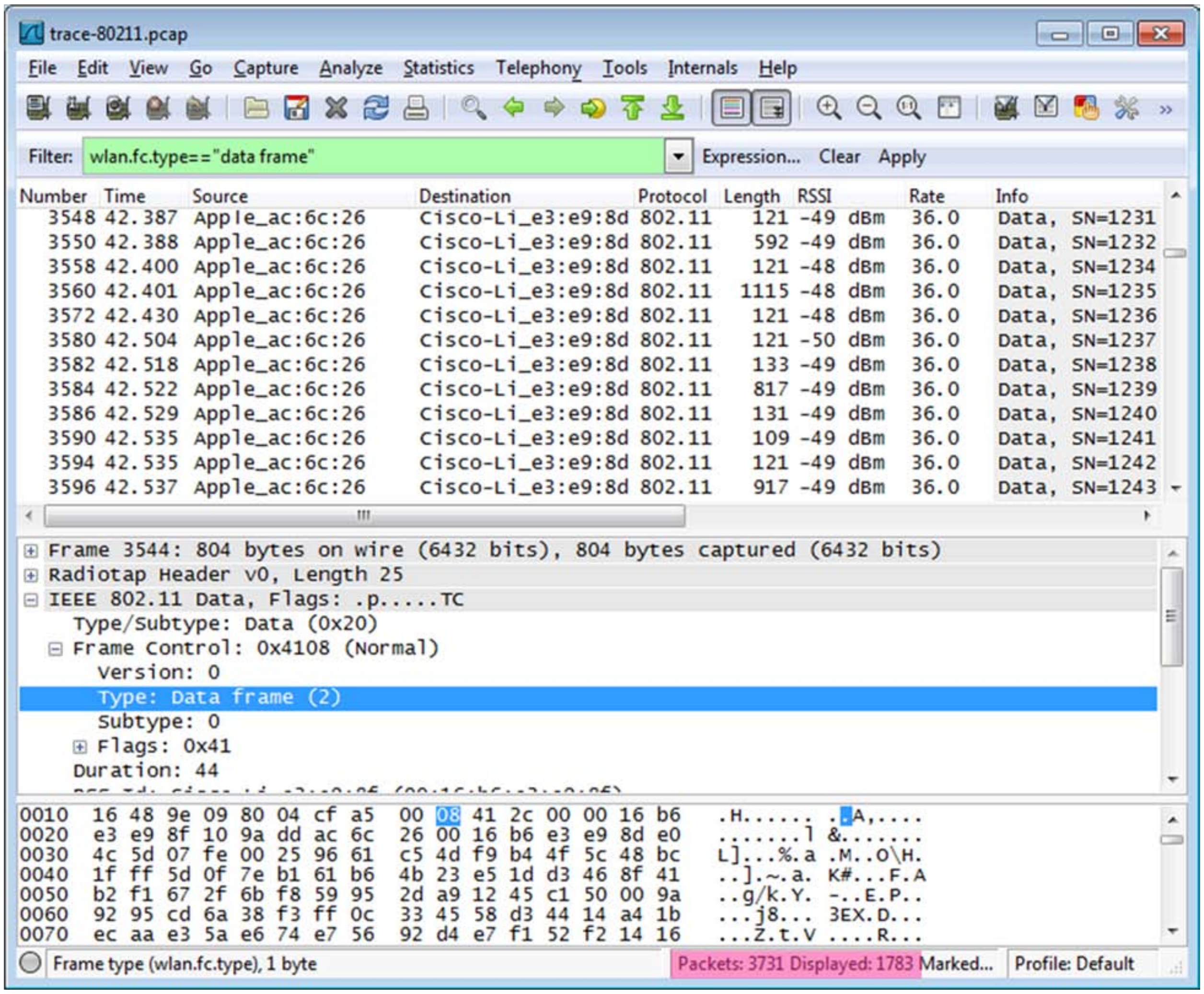
Answer the numbered questions in this step to explore the link layer aspects of 802.11:

1. *What is the BSS ID used by the most active wireless conversations?* A BSS ID value identifies an AP, so this BSS ID identifies the most active AP, presumably the AP we are monitoring. To help find it, you can sort on the source or destination address by clicking on the column heading.

We can also look to see the amounts we have of different types of traffic. 802.11 frames are either Data, Control, or Management frames. These frames are distinguished by the value in the Type subfield of the Frame Control field. You can inspect different packets to see the values for different types of frames.

Filter to see only Data frames by entering the expression “`wlan.fc.type==2`” *into the Filter box above the list of frames in the top panel.* Clicking on the Type subfield tells us in the status display at bottom that Wireshark knows this field by the name wlan.fc.type. Thus, the expression to filter for Data frames with Type value 2 is “`wlan.fc.type==”data frame”`” or “`wlan.fc.type==2`”. When you enter this expression into your Filter box the display should resemble the figure below. After you apply this filter, the status line at bottom will tell you how many of the trace packets are displayed. This tells you how many Data frames there are in the trace. There may be several different kinds of Data frames depending on the value of the Subtype sub-field, as indicated in the Info column. You can click on this column heading to sort by frame type to see what kinds are prevalent.

2. *How many Data frames are in the trace, and what is the most common subtype of Data frame?*


trace-80211.pcap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: wlan.fc.type=="data frame"
Expression... Clear Apply

```
Number Time   Source             Destination        Protocol Length RSSI     Rate Info
  3548 42.387 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    121 -49 dBm  36.0  Data, SN=1231
  3550 42.388 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    592 -49 dBm  36.0  Data, SN=1232
  3558 42.400 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    121 -48 dBm  36.0  Data, SN=1234
  3560 42.401 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11   1115 -48 dBm  36.0  Data, SN=1235
  3572 42.430 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    121 -48 dBm  36.0  Data, SN=1236
  3580 42.504 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    121 -50 dBm  36.0  Data, SN=1237
  3582 42.518 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    133 -49 dBm  36.0  Data, SN=1238
  3584 42.522 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    817 -49 dBm  36.0  Data, SN=1239
  3586 42.529 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    131 -49 dBm  36.0  Data, SN=1240
  3590 42.535 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    109 -49 dBm  36.0  Data, SN=1241
  3594 42.535 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    121 -49 dBm  36.0  Data, SN=1242
  3596 42.537 Apple_ac:6c:26     Cisco-Li_e3:e9:8d  802.11    917 -49 dBm  36.0  Data, SN=1243
```

```
⊞ Frame 3544: 804 bytes on wire (6432 bits), 804 bytes captured (6432 bits)
⊞ Radiotap Header v0, Length 25
⊟ IEEE 802.11 Data, Flags: .p.....TC
    Type/Subtype: Data (0x20)
  ⊟ Frame Control: 0x4108 (Normal)
      Version: 0
      Type: Data frame (2)
      Subtype: 0
    ⊞ Flags: 0x41
    Duration: 44
```

```
0010  16 48 9e 09 80 04 cf a5  00 08 41 2c 00 00 16 b6   .H...... ..A,....
0020  e3 e9 8f 10 9a dd ac 6c  26 00 16 b6 e3 e9 8d e0   .......l &.......
0030  4c 5d 07 fe 00 25 96 61  c5 4d f9 b4 4f 5c 48 bc   L]...%.a .M..O\H.
0040  1f ff 5d 0f 7e b1 61 b6  4b 23 e5 1d d3 46 8f 41   ..].~.a. K#...F.A
0050  b2 f1 67 2f 6b f8 59 95  2d a9 12 45 c1 50 00 9a   ..g/k.Y. -..E.P..
0060  92 95 cd 6a 38 f3 ff 0c  33 45 58 d3 44 14 a4 1b   ...j8... 3EX.D...
0070  ec aa e3 5a e6 74 e7 56  92 d4 e7 f1 52 f2 14 16   ...Z.t.V ....R...
```

```
Frame type (wlan.fc.type), 1 byte    Packets: 3731 Displayed: 1783 Marked...    Profile: Default
```

Figure 5: Filtering the wireless trace for Data frames

Perform the same exercise for Control (Type 1) and Management (Type 0) frames by changing the filter expression to search for a different Type value. This will let you find out how many of these frames are in the trace, and their most prevalent kinds.

3. *How many Control frames are in the trace, and what is the most common subtype?*
4. *How many Management frames are in the trace, and what is the most common subtype?*

As you look at these different types of frames, note their lengths. Data frames may be long, up to 1500 bytes, while Management frames are typically much shorter, and Control frames are very short. You should conclude that most of the bytes in the trace are taken up in Data frames, even though there are many other frames. This is reassuring, since the whole goal of 802.11 is to transfer data.

Inspect the IEEE 802.11 record of an Acknowledgement frame. We have looked at the format of Data frames, so let us turn to Acknowledgement frames. You should see that it has few fields compared to a Data frame, e.g., only one address, and that it is very short.

5. *List in the order they are sent the IEEE 802.11 fields in an Acknowledgement frame and their lengths in bytes.* Do not break down the Frame Control field into subfields, as we have already looked at these details.

We will investigate Management frames in the next step. To conclude our look at the link layer, let us consider reliability and features such as power management. We expect that wireless transmissions are not highly reliable, like well-engineered wired transmissions, but the wireless error rate should not be very large or much of the medium would be wasted – let's see. We can estimate the retransmission rate or by checking to see how many frames have their Retry bit set in the Frame Control field. This bit indicates that a frame is a retransmission of an original.

Use filter expressions to find the number of data frames that are originals and retransmissions. For example, "`wlan.fc.type==2 && wlan.fc.retry==0`" will find original Data frames.

6. *Give an estimate of the retransmission rate as the number of retransmissions over the number of original transmissions. Show your calculation.*

Finally, we will look at power management. Increasingly, 802.11 client devices use power management functionality to go to a low-power sleep mode when they are finished sending or receiving traffic. Clients that are going to sleep set the Power Management flag in the Frame Control field.

Use a filter expression to search for frames that indicate a client is going to sleep. You can find all frames indicating sleep with the expression "`wlan.fc.pwrmgt==1`". You only want to consider power saving behavior in frames going from a client to the AP, as frames coming from the AP will not indicate that a client is going to sleep. These frames will have the "to DS" flag set ("`wlan.fc.tods==1`"). To search for both conditions, you can combine filter expressions with "&&" or "`and`".

7. *What fraction of the frames sent to the AP signal that the client is powering down?*

As you browse the frames that use power management, you are likely to see some unusual types. For instance, the "Null function" frame carries no data. Instead, it is sent by a client to signal sleep.

Turn-in: Hand in your answers to the above questions.

Step 5: 802.11 Management

As well as the Data and Acknowledgment frames, we will look at several types of Management frames that are used to connect a computer to an AP so that it may send and receive messages.

Beacon Frames

Select a Beacon frame in your trace whose BSS ID is that of the main AP from Step 4. Beacon frames are sent out periodically by an AP to advertise its existence and capabilities to nearby computers. The IEEE 802.11 record for this frame will be similar to the record for a Data frame that we reviewed above, with different type and subtype codes to indicate that it is a Beacon frame. However, the payload of this frame will differ: it is an IEEE 802.11 wireless LAN management frame record. You will see that after some fixed parameters it has a series of tagged parameters that list the capabilities of the AP. These include the SSID name of the AP (a text string to go with the BSS ID), the data rates it supports, and the channel on which it is operating.

Expand the payload of the Beacon frames to view its parameters and answer these questions:

1. *What is the SSID of the main AP?* This is one of the tagged parameters in the Beacon frame.
2. *How often are Beacon frames sent for the main AP?* You may find the Beacon interval given in the Beacon frame itself, or change the Time display to be show the interval since the last frame. (Under View, select Time Display Format, and "Seconds Since Previous Displayed Packet".)
3. *What data rates does the main AP support?* The rates are listed under tagged parameters.
4. *What rate is the Beacon frame transmission?* The answer to this question will be found on the Radiotap header, or more conveniently displayed in the column you added in an earlier step.

Association

Once a computer has learned of an AP via a Beacon or otherwise, it must associate with the AP and possibly authenticate itself before it can use the wireless network. You will see the computer send the Association Request to the AP until it is acknowledged. If association is successful then the AP will return an Association Response, which the computer will acknowledge. After the usual IEEE 802.11 header fields, the Association Request and Response carry information that describes the capabilities of the AP and computer, such as what rates it supports. In this way, both endpoints can know the other's abilities.

Find and examine an Association Request and Association Response frame to answer this question:

5. *What are the Type and Subtype values of Association Request / Association Response frames?*

You may also see Authentication Request and Authentication Response frames before the association. This is legacy behavior; the type of authentication is usually "Open", meaning that it provides no security. Instead, the computer and AP share a pre-configured key with WEP, and for WPA2 (the modern scheme) an 802.1X authentication dialogue happens after association using higher layer protocols.

Probe Request/Response

Finally, we will look briefly at Probe frames. Instead of a computer waiting to learn about an AP from Beacons, a computer may probe for specific APs. A Probe Request is sent by a computer to test whether an AP with a specific SSID is nearby. If the sought after AP is nearby then it will reply with a Probe Response. Like Beacon and Association frames, each of these frames has the usual header and carries a list of parameters describing the capabilities of the computer and AP. It is common for computers to send Probe Requests for wireless networks that they have previously used to speed up connection to a known network, e.g., when a laptop has returned home for the day. Thus you may see a sequence of probes for many different SSIDs. Only the SSIDs that are present will reply.

Find and examine a Probe Request and Probe Response frame to answer this question:

6. *What are the Type and Subtype values for the Probe Request / Probe Response frames?*

Turn-in: Hand in the answers to the above questions.

Congratulations, you now know a great deal about 802.11 in practice!

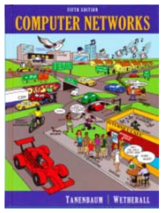
Explore on your own

We encourage you to explore 802.11 on your own once you have completed this lab. We have covered the basics of many topics, each of which you can delve into more deeply. Some ideas:

- Look to see how a given client uses different rates over time. This is called rate adaptation.
- See which clients are using power management and if you can understand their sleep behavior.
- See whether you can find clients that use the RTS/CTS mechanism.
- Look for Probe Request sequences to see when computers send them and what you can learn.
- Configure Wireshark with WEP/WPA2 keys to see witihin the 802.11 payload, starting with LLC.
- Try all of the above with a trace taken from your own network.

[END]

Lab Exercise – ARP



Objective

To see how ARP (Address Resolution Protocol) works. ARP is an essential glue protocol that is used to join Ethernet and IP. It is covered in §5.6.4 of your text. Review the text section before doing this lab.

Requirements

Wireshark: This lab uses the Wireshark software tool to capture and examine a packet trace. A packet trace is a record of traffic at a location on the network, as if a snapshot was taken of all the bits that passed across a particular wire. The packet trace records a timestamp for each packet, along with the bits that make up the packet, from the lower-layer headers to the higher-layer contents. Wireshark runs on most operating systems, including Windows, Mac and Linux. It provides a graphical UI that shows the sequence of packets and the meaning of the bits when interpreted as protocol headers and data. It color-codes packets by their type, and has various ways to filter and analyze packets to let you investigate the behavior of network protocols. Wireshark is widely used to troubleshoot networks. You can download it from www.wireshark.org if it is not already installed on your computer. We highly recommend that you watch the short, 5 minute video "Introduction to Wireshark" that is on the site.

arp: This lab uses the "`arp`" command-line utility to inspect and clear the cache used by the ARP protocol on your computer. `arp` is installed as part of the operating system on Windows, Linux, and Mac computers, but uses different arguments. It requires administrator privileges to clear the cache.

ifconfig / ipconfig: This lab uses the "`ipconfig`" (Windows) or "`ifconfig`" (Mac/Linux) command-line utility to inspect the state of your computer's network interface. `ifconfig/ipconfig` is installed as part of the operating system on Windows, Linux, and Mac computers.

route / netstat: This lab uses the "`route`" or "`netstat`" command-line utility to inspect the routes used by your computer. A key route is the default route (or route to prefix 0.0.0.0) that uses the default gateway to reach remote parts of the Internet. Both "`route`" and "`netstat`" are installed as part of the operating system across Windows and Mac/Linux, but there are many variations on the command-line parameters that must be used.

Browser: This lab uses a web browser to find or fetch pages as a workload. Any web browser will do.

Network Setup

We want to observe the ARP protocol in action. Recall that ARP is used to find the Ethernet address that corresponds to a local IP address to which your computer wants to send a packet. A typical example of a local IP address is that of the local router or default gateway that connects your computer to the rest of the Internet. Your computer caches these translations in an ARP cache so that the ARP protocol need only be used occasionally to do the translation. The setup from the viewpoint of your computer is as shown in the example below.

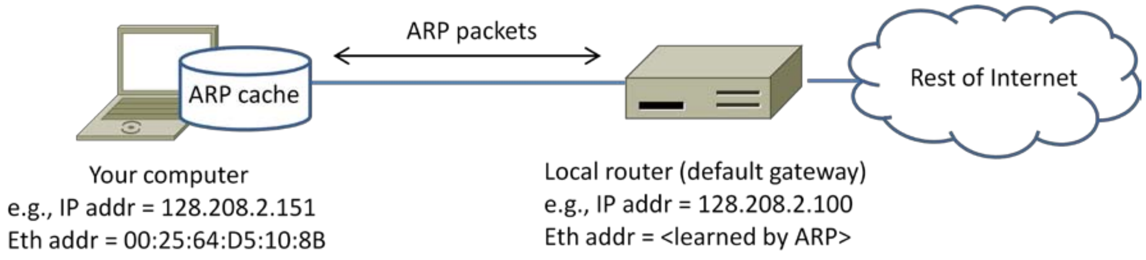
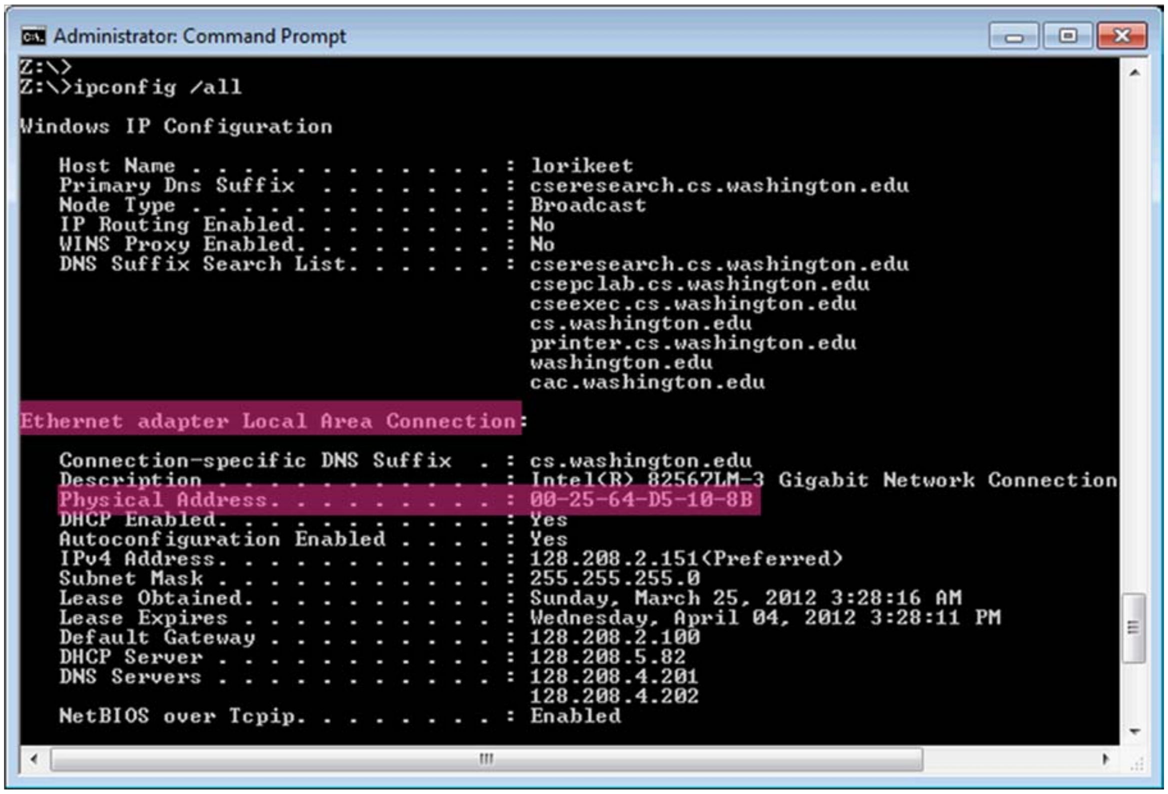

ARP packets
ARP cache
Rest of Internet
Your computer
e.g., IP addr = 128.208.2.151
Eth addr = 00:25:64:D5:10:8B
Local router (default gateway)
e.g., IP addr = 128.208.2.100
Eth addr = <learned by ARP>

Figure 1: Network setup under which we will study ARP

Step 1: Capture a Trace

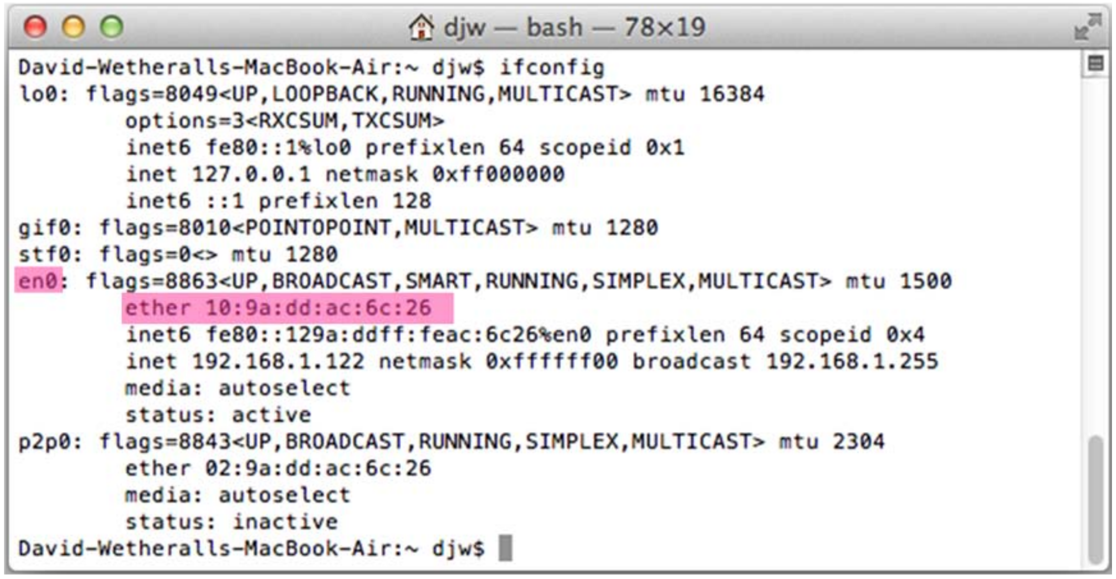
Proceed as follows to capture a trace of ARP traffic; alternatively, you may use a supplied trace. To gather ARP packets, we will cause your computer to send traffic to the local router when it does not know the router's Ethernet address – your computer will then use ARP to discover the Ethernet address.

1. *Find the Ethernet address of the main network interface of your computer with the* `ifconfig` / `ipconfig` *command*. You will want to know this address for later analysis. On Windows, bring up a command-line shell and type "`ipconfig /all`". On Mac/Linux, bring up a command-line shell and type "`ifconfig`". Among the output will be a section for the main interface of the computer (likely an Ethernet interface) and its Ethernet address. Common names for the interface are "eth0", "en0", or "Ethernet adapter". Two examples are shown below, with our added highlighting.



```
Z:\>
Z:\>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : lorikeet
   Primary Dns Suffix  . . . . . . . : cseresearch.cs.washington.edu
   Node Type . . . . . . . . . . . . : Broadcast
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : cseresearch.cs.washington.edu
                                       csepclab.cs.washington.edu
                                       cseexec.cs.washington.edu
                                       cs.washington.edu
                                       printer.cs.washington.edu
                                       washington.edu
                                       cac.washington.edu

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : cs.washington.edu
   Description . . . . . . . . . . . : Intel(R) 82567LM-3 Gigabit Network Connection
   Physical Address. . . . . . . . . : 00-25-64-D5-10-8B
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv4 Address. . . . . . . . . . . : 128.208.2.151(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Sunday, March 25, 2012 3:28:16 AM
   Lease Expires . . . . . . . . . . : Wednesday, April 04, 2012 3:28:11 PM
   Default Gateway . . . . . . . . . : 128.208.2.100
   DHCP Server . . . . . . . . . . . : 128.208.5.82
   DNS Servers . . . . . . . . . . . : 128.208.4.201
                                       128.208.4.202
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

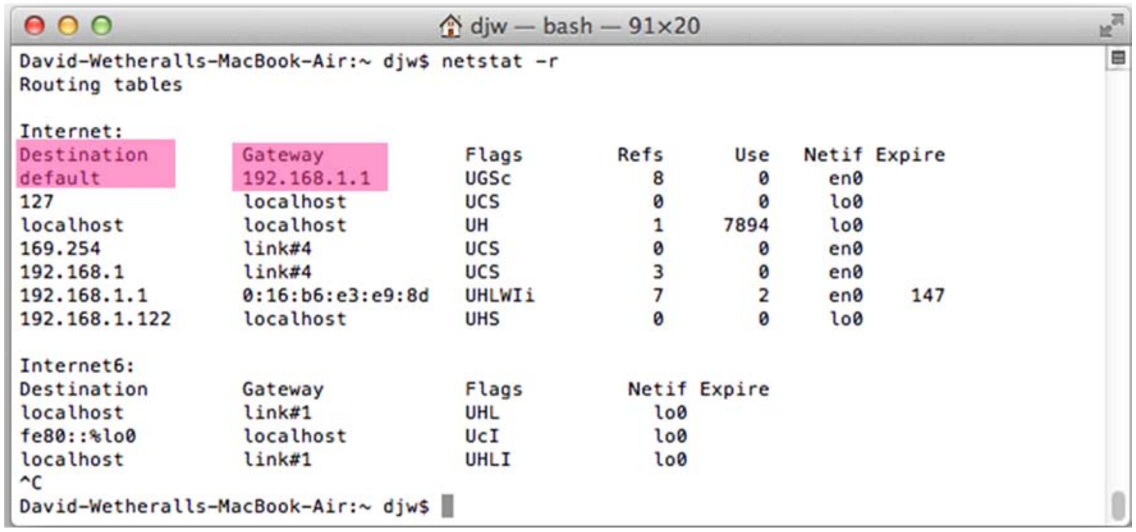
Figure 2: Finding the computer's Ethernet address with `ipconfig` (Windows)



```
David-Wetheralls-MacBook-Air:~ djw$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=3<RXCSUM,TXCSUM>
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 10:9a:dd:ac:6c:26
        inet6 fe80::129a:ddff:feac:6c26%en0 prefixlen 64 scopeid 0x4
        inet 192.168.1.122 netmask 0xffffff00 broadcast 192.168.1.255
        media: autoselect
        status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
        ether 02:9a:dd:ac:6c:26
        media: autoselect
        status: inactive
David-Wetheralls-MacBook-Air:~ djw$
```

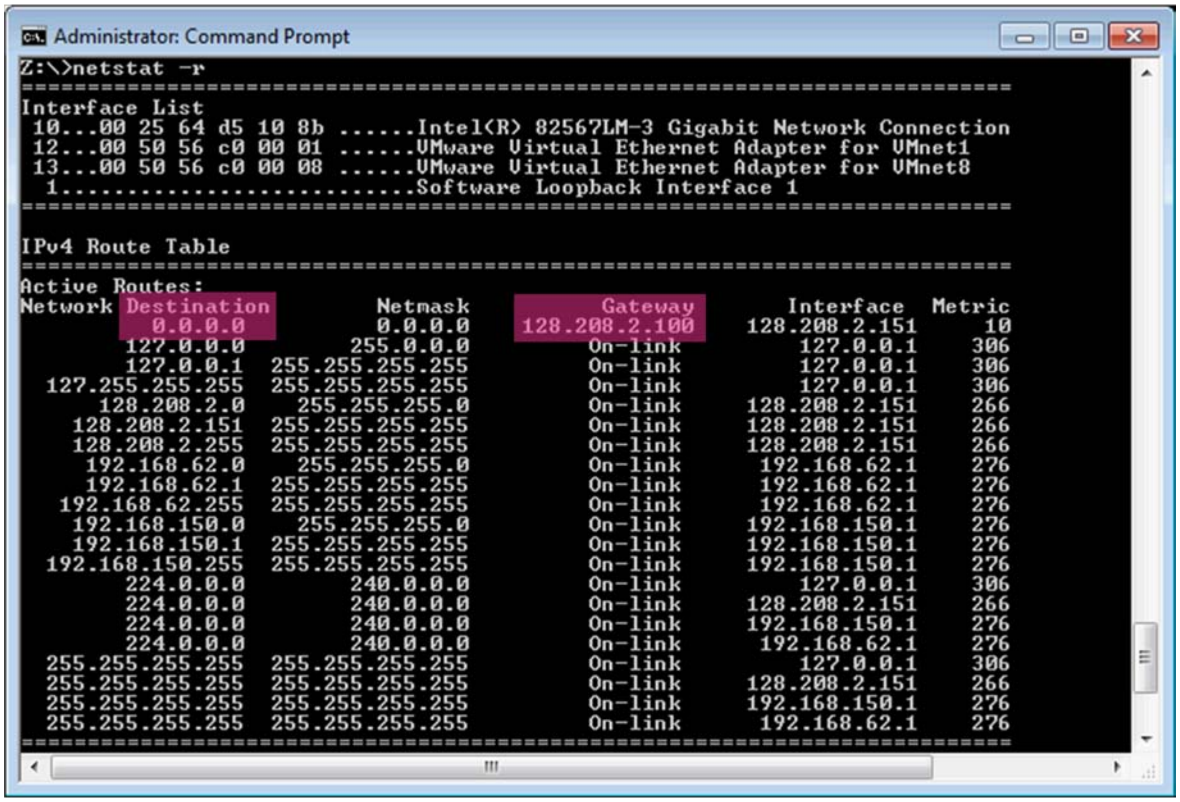
Figure 3: Finding the computer's Ethernet address with `ifconfig` (Mac)

2. *Find the IP address of the local router or default gateway that your computer uses to reach the rest of the Internet using the* `netstat` */* `route` *command.* You should be able to use the `netstat` command ("`netstat -r`" on Windows, Mac and Linux, may require ctrl-C to stop). Alternatively, you can use the route command ("`route print`" on Windows, "`route`" on Linux, "`route -n get default`" on Mac). In either case you are looking for the gateway IP address that corresponds to the destination of default or 0.0.0.0. Two examples are shown below for `netstat`, with our added highlighting.



```
David-Wetheralls-MacBook-Air:~ djw$ netstat -r
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            192.168.1.1        UGSc            8        0     en0
127                localhost          UCS             0        0     lo0
localhost          localhost          UH              1     7894     lo0
169.254            link#4             UCS             0        0     en0
192.168.1          link#4             UCS             3        0     en0
192.168.1.1        0:16:b6:e3:e9:8d   UHLWIi          7        2     en0    147
192.168.1.122      localhost          UHS             0        0     lo0

Internet6:
Destination        Gateway            Flags         Netif Expire
localhost          link#1             UHL             lo0
fe80::%lo0         localhost          UcI             lo0
localhost          link#1             UHLI            lo0
^C
David-Wetheralls-MacBook-Air:~ djw$
```

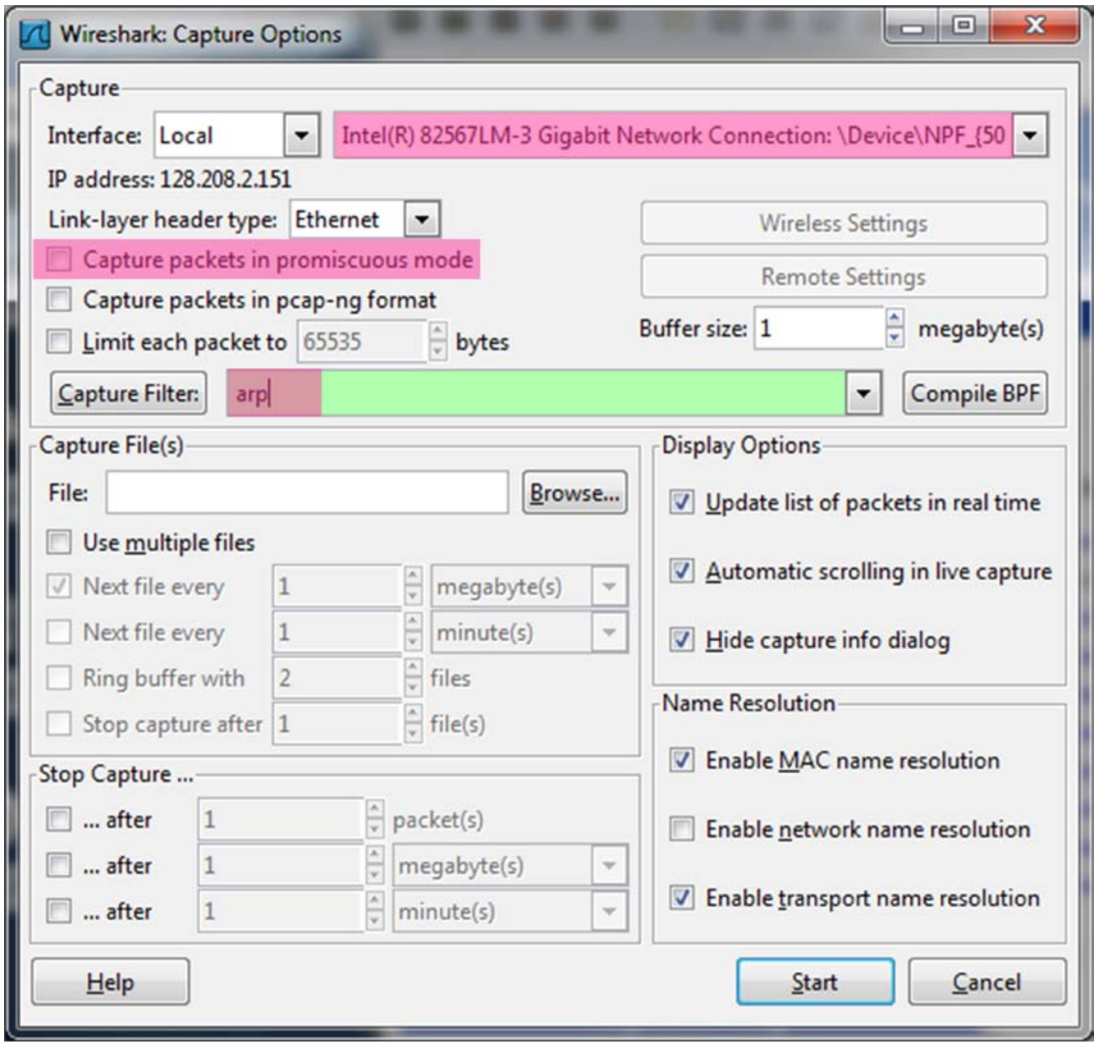
Figure 4: Finding the default gateway IP address with `netstat` (Mac)



```
Z:\>netstat -r
===========================================================================
Interface List
 10...00 25 64 d5 10 8b ......Intel(R) 82567LM-3 Gigabit Network Connection
 12...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter for VMnet1
 13...00 50 56 c0 00 08 ......VMware Virtual Ethernet Adapter for VMnet8
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    128.208.2.100    128.208.2.151     10
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
      128.208.2.0    255.255.255.0         On-link     128.208.2.151    266
    128.208.2.151  255.255.255.255         On-link     128.208.2.151    266
    128.208.2.255  255.255.255.255         On-link     128.208.2.151    266
     192.168.62.0    255.255.255.0         On-link      192.168.62.1    276
     192.168.62.1  255.255.255.255         On-link      192.168.62.1    276
   192.168.62.255  255.255.255.255         On-link      192.168.62.1    276
    192.168.150.0    255.255.255.0         On-link     192.168.150.1    276
    192.168.150.1  255.255.255.255         On-link     192.168.150.1    276
  192.168.150.255  255.255.255.255         On-link     192.168.150.1    276
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
        224.0.0.0        240.0.0.0         On-link     128.208.2.151    266
        224.0.0.0        240.0.0.0         On-link     192.168.150.1    276
        224.0.0.0        240.0.0.0         On-link      192.168.62.1    276
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link     128.208.2.151    266
  255.255.255.255  255.255.255.255         On-link     192.168.150.1    276
  255.255.255.255  255.255.255.255         On-link      192.168.62.1    276
===========================================================================
```

Figure 5: Finding the default gateway IP address with `netstat` (Windows)

3. *Launch Wireshark and start a capture with a filter of* "`arp`". Your capture window should be similar to the one pictured below, other than our highlighting. Select the interface from which to capture as the main wired or wireless interface used by your computer to connect to the Internet. If unsure, guess and revisit this step later if your capture is not successful. Uncheck "capture packets in promiscuous mode". This mode is useful to overhear packets sent to/from other computers on broadcast networks. We only want to record packets sent to/from your computer. Leave other options at their default values. The capture filter, if present, is used to prevent the capture of other traffic your computer may send or receive. On Wireshark 1.8, the capture filter box is present directly on the options screen, but on Wireshark 1.9, you set a capture filter by double-clicking on the interface.


Wireshark: Capture Options

Capture

Interface: Local — Intel(R) 82567LM-3 Gigabit Network Connection: \Device\NPF_{50

IP address: 128.208.2.151

Link-layer header type: Ethernet

Capture packets in promiscuous mode

Capture packets in pcap-ng format

Limit each packet to 65535 bytes

Buffer size: 1 megabyte(s)

Capture Filter: arp

Capture File(s)

File:

Use multiple files

[✓ Next file every] 1 megabyte(s)

Next file every 1 minute(s)

Ring buffer with 2 files

Stop capture after 1 file(s)

Stop Capture ...

... after 1 packet(s)

... after 1 megabyte(s)

... after 1 minute(s)

Display Options

[✓ Update list of packets in real time]

[✓ Automatic scrolling in live capture]

[✓ Hide capture info dialog]

Name Resolution

[✓ Enable MAC name resolution]

Enable network name resolution

[✓ Enable transport name resolution]

Figure 4: Setting up the capture options

4. *When the capture is started, use the "*`arp`*" command to clear the default gateway from the ARP cache.* Using the command "`arp –a`" will show you the contents of the ARP cache as a check that you can run "`arp`". You should see an entry for the IP address of the default gateway. To clear this entry, use the arp command with different arguments ("`arp –d`" on Windows, "`arp -d -a`" on Mac, "`arp -d xx.xx.xx.xx`" where xx.xx.xx.xx is the IP address of the default gateway on Linux). This usage of `arp` will need administrator privileges to run, so you may run as a privileged user on Windows or issue "`sudo arp –d xx.xx.xx.xx`" on Linux/Mac. Note that the command should run without error but the ARP entry may not appear to be cleared if you check with "`arp –a`". This is because your computer will send ARP packets to repopulate this entry as soon as you need to send a packet to a remote IP address, and that can happen very quickly due to background activity on the computer.
5. *Now that you have cleared your ARP cache, fetch a remote page with your Web browser*. This will cause ARP to find the Ethernet address of the default gateway so that the packets can be sent. These ARP packets will be captured by Wireshark. You might clear the ARP cache and fetch a document a couple of times. Hopefully there will also be other ARP packets sent by other

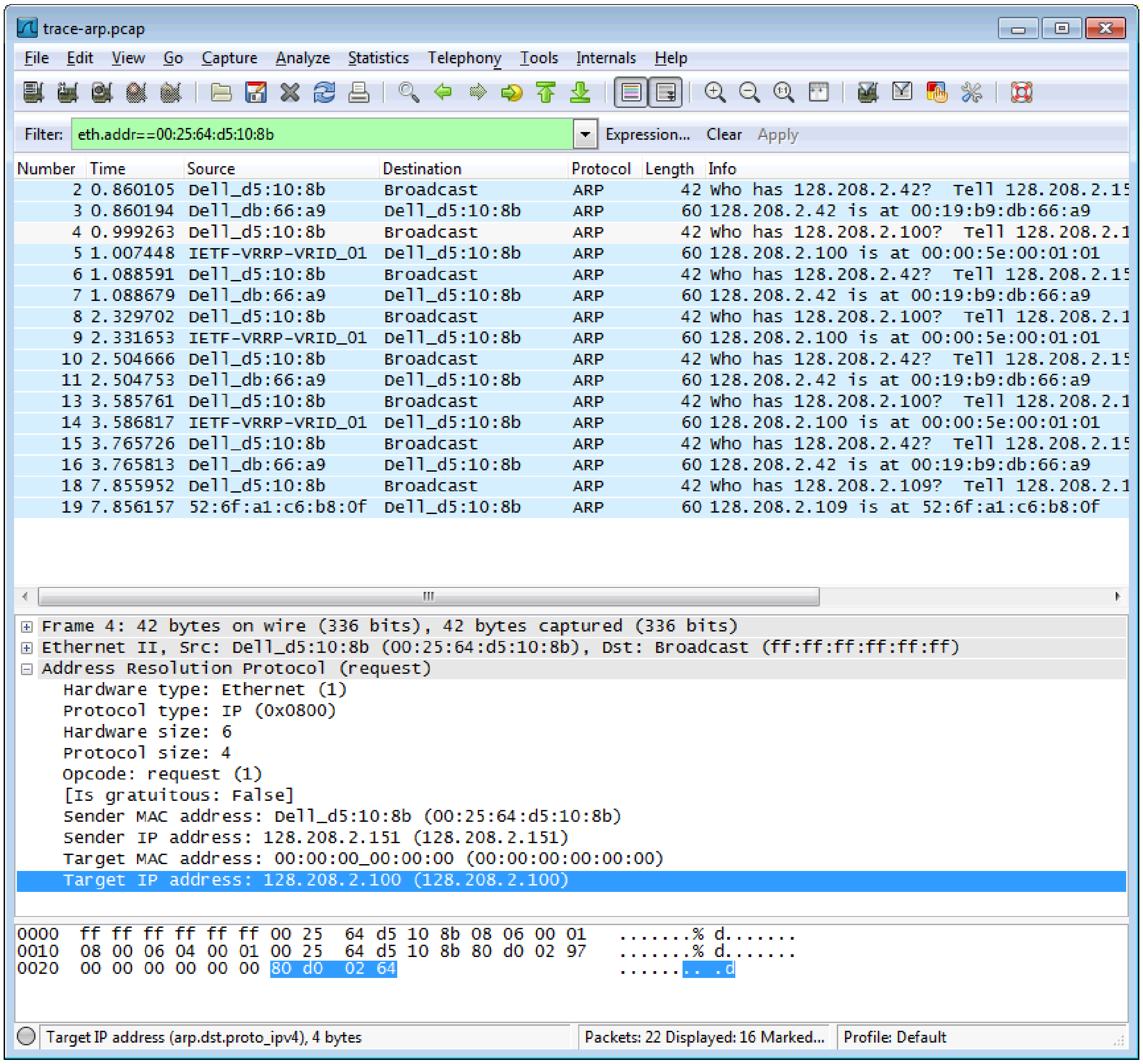
computers on the local network that will be captured. These packets are likely to be present if there are other computers on your local network. In fact, if you have a busy computer and extensive local network then you may capture many ARP packets. The ARP traffic of other computers will be captured when the ARP packets are sent to the broadcast address, since in this case they are destined for all computers including the one on which you are running Wireshark. Because ARP activity happens slowly, you may need to wait up to 30 seconds to observe some of this background ARP traffic.

6. *Once you have captured some ARP traffic, stop the capture.* You will need the trace, plus the Ethernet address of your computer and the IP address of the default gateway for the next steps.

Step 2: Inspect the Trace

Now we can look at an ARP exchange! Since there may be many ARP packets in your trace, we'll first narrow our view to only the ARP packets that are sent directly from or to your computer.

Set a display filter for packets with the Ethernet address of your computer. You can do this by entering an expression in the blank "Filter:" box near the top of the Wireshark window and clicking "Apply". The filter to enter depends on your Ethernet address. For example, if your Ethernet address is 01:02:03:04:05:06 then enter a filter expression of "`eth.addr==01:02:03:04:05:06`". Note the double equal sign. If you are using the supplied trace, it comes with an additional text file giving the Ethernet address and default gateway IP address. After applying this filter your capture should look something like the figure below, in which we have expanded the ARP protocol details.


trace-arp.pcap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: eth.addr==00:25:64:d5:10:8b
Expression... Clear Apply

```
Number Time      Source              Destination       Protocol Length Info
     2 0.860105  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.42?  Tell 128.208.2.15
     3 0.860194  Dell_db:66:a9       Dell_d5:10:8b     ARP      60 128.208.2.42 is at 00:19:b9:db:66:a9
     4 0.999263  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.100?  Tell 128.208.2.1
     5 1.007448  IETF-VRRP-VRID_01   Dell_d5:10:8b     ARP      60 128.208.2.100 is at 00:00:5e:00:01:01
     6 1.088591  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.42?  Tell 128.208.2.15
     7 1.088679  Dell_db:66:a9       Dell_d5:10:8b     ARP      60 128.208.2.42 is at 00:19:b9:db:66:a9
     8 2.329702  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.100?  Tell 128.208.2.1
     9 2.331653  IETF-VRRP-VRID_01   Dell_d5:10:8b     ARP      60 128.208.2.100 is at 00:00:5e:00:01:01
    10 2.504666  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.42?  Tell 128.208.2.15
    11 2.504753  Dell_db:66:a9       Dell_d5:10:8b     ARP      60 128.208.2.42 is at 00:19:b9:db:66:a9
    13 3.585761  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.100?  Tell 128.208.2.1
    14 3.586817  IETF-VRRP-VRID_01   Dell_d5:10:8b     ARP      60 128.208.2.100 is at 00:00:5e:00:01:01
    15 3.765726  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.42?  Tell 128.208.2.15
    16 3.765813  Dell_db:66:a9       Dell_d5:10:8b     ARP      60 128.208.2.42 is at 00:19:b9:db:66:a9
    18 7.855952  Dell_d5:10:8b       Broadcast         ARP      42 Who has 128.208.2.109?  Tell 128.208.2.1
    19 7.856157  52:6f:a1:c6:b8:0f   Dell_d5:10:8b     ARP      60 128.208.2.109 is at 52:6f:a1:c6:b8:0f
```

```
⊞ Frame 4: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
⊞ Ethernet II, Src: Dell_d5:10:8b (00:25:64:d5:10:8b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    [Is gratuitous: False]
    Sender MAC address: Dell_d5:10:8b (00:25:64:d5:10:8b)
    Sender IP address: 128.208.2.151 (128.208.2.151)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 128.208.2.100 (128.208.2.100)
```

```
0000  ff ff ff ff ff ff 00 25  64 d5 10 8b 08 06 00 01   .......% d.......
0010  08 00 06 04 00 01 00 25  64 d5 10 8b 80 d0 02 97   .......% d.......
0020  00 00 00 00 00 00 80 d0  02 64                     ...... .. .d
```

```
Target IP address (arp.dst.proto_ipv4), 4 bytes    Packets: 22 Displayed: 16 Marked...    Profile: Default
```

Figure 5: Capture of ARP packets, showing details of a request

Find and select an ARP request for the default gateway and examine its fields. There are two kinds of ARP packets, a request and a reply, and we will look at each one in turn. The Info line for the request will start with “Who has …”. You want to look for one of these packets that asks for the MAC address of the default gateway, e.g., “Who has xx.xx.xx.xx …” where xx.xx.xx.xx is your default gateway. You can click on the + expander or icon for the Address Resolution Protocol block to view the fields:

- Hardware and Protocol type are set to constants that tell us the hardware is Ethernet and the protocol is IP. This matches the ARP translation from IP to Ethernet address.
- Hardware and Protocol size are set to 6 and 4, respectively. These are the sizes of Ethernet and IP addresses in bytes.
- The opcode field tells us that this is a request.

- Next come the four key fields, the sender MAC (Ethernet) and IP and the target MAC (Ethernet) and IP. These fields are filled in as much as possible. For a request, the sender knows their MAC and IP address and fills them in. The sender also knows the target IP address – it is the IP address for which an Ethernet address is wanted. But the sender does not know the target MAC address, so it does not fill it in.

Next, select an ARP reply and examine its fields. The reply will answer a request and have an Info line of the form "xx.xx.xx.xx is at yy:yy:yy:yy:yy:yy":

- The Hardware and Protocol type and sizes are as set as before.
- The opcode field has a different value that tells us that this is a reply.
- Next come the four key fields, the sender MAC (Ethernet) and IP and the target MAC (Ethernet) and IP just as before. These fields are reversed from the corresponding request, since the old target is the new sender (and vice versa). The fields should now be all filled in since both computers have supplied their addresses.

Step 3: ARP request and reply

To show your understanding of an ARP exchange, draw a figure that shows the ARP request and reply packets sent between your computer and the default gateway. Make it for the case we examined of your computer doing an ARP for the default gateway. Label one packet the request and the other the reply. Give the sender and target MAC and IP addresses for each packet; you can use Wireshark to inspect the packets to get these values. Finally, circle the sought after Ethernet address on your drawing to show where it comes from in the exchange.

Turn-in: Hand in your drawing of the ARP exchange.

Step 4: Details of ARP over Ethernet

To look at further details of ARP, examine an ARP request and ARP reply to answer these questions:

1. *What opcode is used to indicate a request? What about a reply?*
2. *How large is the ARP header for a request? What about for a reply?*
3. *What value is carried on a request for the unknown target MAC address?*

ARP packets are carried in Ethernet frames, and the values of the Ethernet header fields are chosen to support ARP. For instance, you may wonder how an ARP request packet is delivered to the target computer so that it can reply and tell the requestor its MAC address. The answer is that the ARP request is (normally) broadcast at the Ethernet layer so that it is received by all computers on the local network including the target. Look specifically at the destination Ethernet address of a request: it is set to `ff:ff:ff:ff:ff:ff`, the broadcast address. So the target receives the request and recognizes that it is the intended recipient of the message; other computers that receive the request know that it is not meant for them. Only the target responds with a reply. However, anyone who receives an ARP packet can learn a mapping from it: the sender MAC and sender IP pair.

Examine an ARP request and reply to answer these questions:

4. *What Ethernet Type value which indicates that ARP is the higher layer protocol?*
5. *Is the ARP reply broadcast (like the ARP request) or not?*

Turn-in: Hand in your answers to the above questions.

Explore on your own

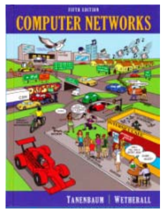
We encourage you to explore ARP on your own once you have completed this lab. One suggestion is to look at other ARP packets that may have been recorded in your trace; we only examined an ARP request by your computer and the ARP reply from the default gateway.

To see if there is other ARP activity, make sure to clear any Ethernet address filter that is set. Other ARP packets may exhibit any of the following kinds of behavior for you to explore:

- ARP requests broadcast by other computers. The other computers on the local network are also using ARP. Since requests are broadcast, your computer will receive their requests.
- ARP replies sent by your computer. If another computer happens to ARP for the IP address of your computer, then your computer will send an ARP reply to tell it the answer.
- Gratuitous ARPs in which your computer sends a request or reply about itself. This is helpful when a computer or link comes up to make sure that no-one else is using the same IP address. Gratuitous ARPs have the same sender and target IP address, and they have an Info field in Wireshark that identified them as gratuitous.
- Other ARP requests sent by your computer and the corresponding ARP reply. Your computer may need to ARP for other hosts besides the default gateway after you flush its ARP cache.

[END]

Lab Exercise – Ethernet



Objective

To explore the details of Ethernet frames. Ethernet is a popular link layer protocol that is covered in §4.3 of your text; modern computers connect to Ethernet switches (§4.3.4) rather than use classic Ethernet (§4.3.2). Review section §4.3 before doing this lab.

Requirements

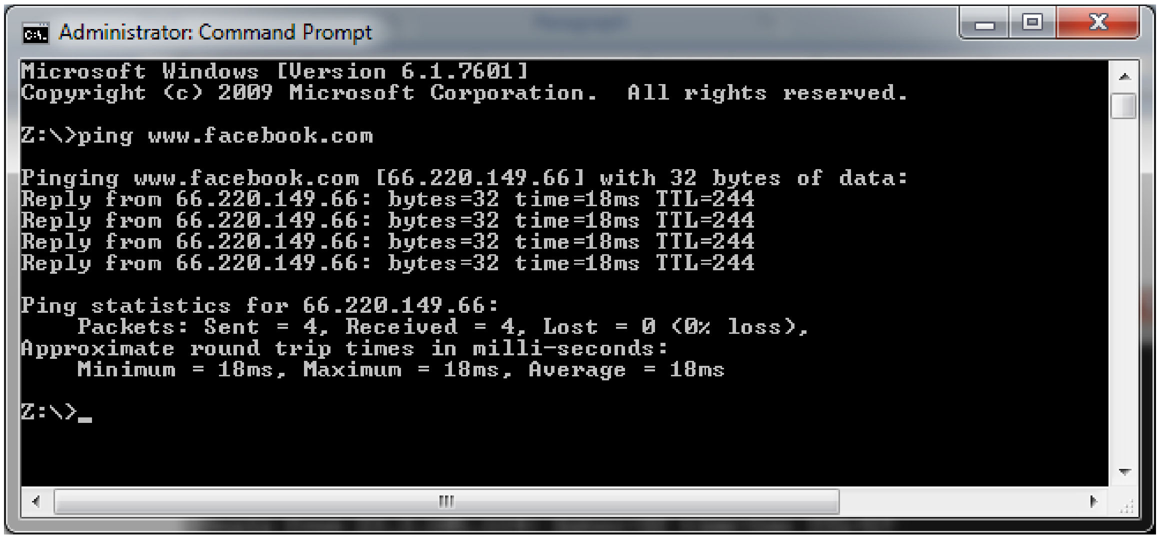
Wireshark: This lab uses the Wireshark software tool to capture and examine a packet trace. A packet trace is a record of traffic at a location on the network, as if a snapshot was taken of all the bits that passed across a particular wire. The packet trace records a timestamp for each packet, along with the bits that make up the packet, from the lower-layer headers to the higher-layer contents. Wireshark runs on most operating systems, including Windows, Mac and Linux. It provides a graphical UI that shows the sequence of packets and the meaning of the bits when interpreted as protocol headers and data. It color-codes packets by their type, and has various ways to filter and analyze packets to let you investigate the behavior of network protocols. Wireshark is widely used to troubleshoot networks. You can download it from www.wireshark.org if it is not already installed on your computer. We highly recommend that you watch the short, 5 minute video "Introduction to Wireshark" that is on the site.

ping: This lab uses "`ping`" to send and receive messages. `ping` is a standard command-line utility for checking that another computer is responsive. It is widely used for network troubleshooting and comes pre-installed on Window, Linux, and Mac. While `ping` has various options, simply issuing the command "`ping www.bing.com`" will cause your computer to send a small number of ICMP ping requests to the remote computer (here www.bing.com), each of which should elicit an ICMP ping response.

Step 1: Capture a Trace

Proceed as follows to capture a trace of ping packets; alternatively you may use a supplied trace. We will use ping simply as an easy way to collect a small trace. Perhaps surprisingly, you can capture a trace for this lab from a computer connected to the Internet using either wired Ethernet or wireless 802.11.

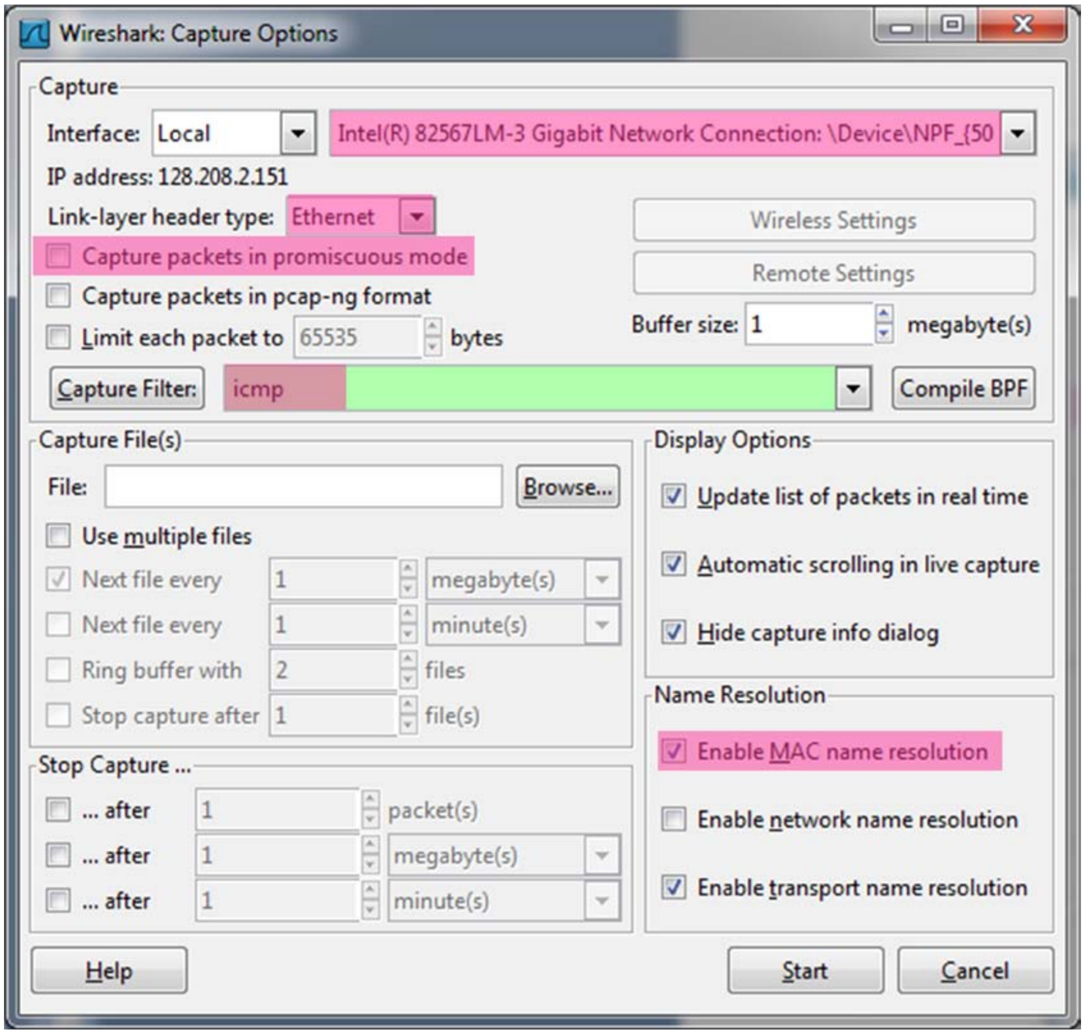
1. *Pick a remote web server or other publicly reachable Internet host and use* `ping` *to send some ping messages and check that it sends replies*. For example, "`ping www.bing.com`". You should see several replies indicating that the pings reached the remote host and were returned. The figure below shows a successful example. Note that some versions of `ping` will continue to bounce messages off of a remote server until you tell the program to stop by signaling it with ^C. If your ping test does not succeed then try another server.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

Z:\>ping www.facebook.com

Pinging www.facebook.com [66.220.149.66] with 32 bytes of data:
Reply from 66.220.149.66: bytes=32 time=18ms TTL=244
Reply from 66.220.149.66: bytes=32 time=18ms TTL=244
Reply from 66.220.149.66: bytes=32 time=18ms TTL=244
Reply from 66.220.149.66: bytes=32 time=18ms TTL=244

Ping statistics for 66.220.149.66:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 18ms, Maximum = 18ms, Average = 18ms

Z:\>_
```

Figure 1: Using `ping` to bounce messages off a remote host

2. *Launch Wireshark and start a capture of Ethernet frames with a filter of* "`icmp`"*, making sure that "enable MAC name resolution" is checked*. The latter will translate Ethernet (MAC) addresses to provide vendor information. Also check that the Link-layer header type pulldown says "Ethernet". Your capture window should be similar to the one pictured below, other than our highlighting. Select the interface from which to capture as the main wired or wireless interface used by your computer to connect to the Internet. If unsure, guess and revisit this step later if your capture is not successful. Uncheck "capture packets in promiscuous mode". This mode is useful to overhear packets sent to/from other computers on broadcast networks. We only want to record packets sent to/from your computer. Leave other options at their default values. The capture filter, if present, is used to prevent the capture of other traffic your computer may send or receive. On Wireshark 1.8, the capture filter box is present directly on the options screen, but on Wireshark 1.9, you set a capture filter by double-clicking on the interface.


Wireshark: Capture Options

Capture
Interface: Local | Intel(R) 82567LM-3 Gigabit Network Connection: \Device\NPF_{50
IP address: 128.208.2.151
Link-layer header type: Ethernet
☐ Capture packets in promiscuous mode
☐ Capture packets in pcap-ng format
☐ Limit each packet to 65535 bytes
Buffer size: 1 megabyte(s)
Capture Filter: icmp

Capture File(s)
File:
☐ Use multiple files
✓ Next file every 1 megabyte(s)
☐ Next file every 1 minute(s)
☐ Ring buffer with 2 files
☐ Stop capture after 1 file(s)

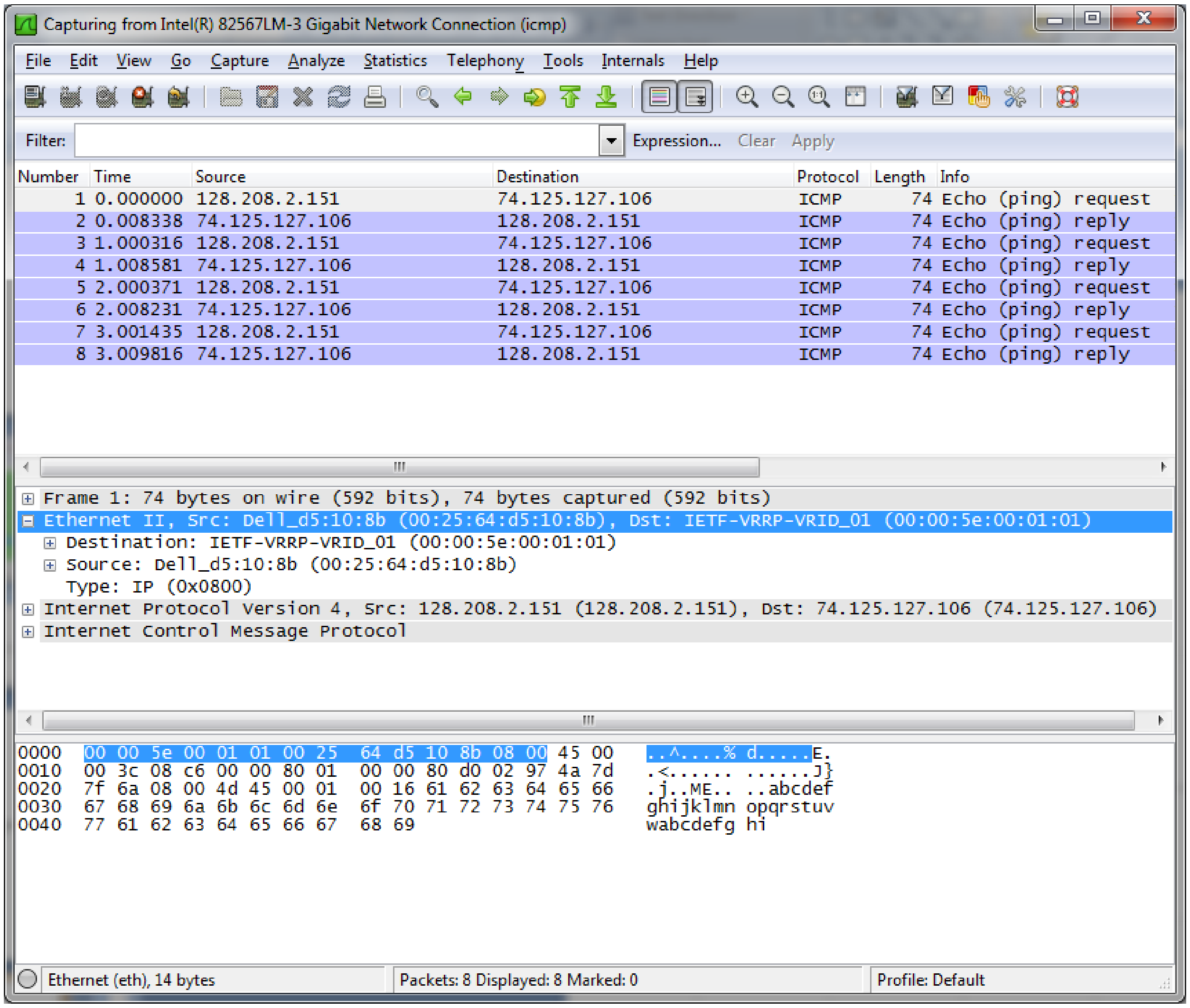
Stop Capture ...
☐ ... after 1 packet(s)
☐ ... after 1 megabyte(s)
☐ ... after 1 minute(s)

Display Options
✓ Update list of packets in real time
✓ Automatic scrolling in live capture
✓ Hide capture info dialog

Name Resolution
✓ Enable MAC name resolution
☐ Enable network name resolution
✓ Enable transport name resolution

Figure 2: Setting the capture options for `ping` traffic

3. *When the capture is started, repeat the* `ping` *command above.* This time, the packets will also be recorded by Wireshark.
4. *After the* `ping` *command is complete, return to Wireshark and use the menus or buttons to stop the trace.* You should now have a short trace similar to that shown in the figure below. If you do not succeed in capturing a trace then use the supplied one. Note that the trace we supply begins with ping messages, and then has other kinds of Ethernet frames.


Capturing from Intel(R) 82567LM-3 Gigabit Network Connection (icmp)
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
Number Time     Source          Destination     Protocol Length Info
     1 0.000000 128.208.2.151   74.125.127.106  ICMP         74 Echo (ping) request
     2 0.008338 74.125.127.106  128.208.2.151   ICMP         74 Echo (ping) reply
     3 1.000316 128.208.2.151   74.125.127.106  ICMP         74 Echo (ping) request
     4 1.008581 74.125.127.106  128.208.2.151   ICMP         74 Echo (ping) reply
     5 2.000371 128.208.2.151   74.125.127.106  ICMP         74 Echo (ping) request
     6 2.008231 74.125.127.106  128.208.2.151   ICMP         74 Echo (ping) reply
     7 3.001435 128.208.2.151   74.125.127.106  ICMP         74 Echo (ping) request
     8 3.009816 74.125.127.106  128.208.2.151   ICMP         74 Echo (ping) reply
```

```
⊞ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
⊟ Ethernet II, Src: Dell_d5:10:8b (00:25:64:d5:10:8b), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
  ⊞ Destination: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
  ⊞ Source: Dell_d5:10:8b (00:25:64:d5:10:8b)
    Type: IP (0x0800)
⊞ Internet Protocol Version 4, Src: 128.208.2.151 (128.208.2.151), Dst: 74.125.127.106 (74.125.127.106)
⊞ Internet Control Message Protocol
```

```
0000  00 00 5e 00 01 01 00 25  64 d5 10 8b 08 00 45 00   ..^....% d.....E.
0010  00 3c 08 c6 00 00 80 01  00 00 80 d0 02 97 4a 7d   .<...... ......J}
0020  7f 6a 08 00 4d 45 00 01  00 16 61 62 63 64 65 66   .j..ME.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

```
Ethernet (eth), 14 bytes    Packets: 8 Displayed: 8 Marked: 0    Profile: Default
```

Figure 3: Trace of `ping` traffic, showing Ethernet details of the first packet

Step 2: Inspect the Trace

Select any packet in the trace (in the top panel) to see details of its structure (in the middle panel) and the bytes that make up the packet (in the bottom panel). Now we can inspect the details of the packets. In the figure, we have selected the first packet in the trace. Note that we are using the term "packet" in a loose way. Each record captured by Wireshark more correctly corresponds to a single frame in Ethernet format that carries a packet as its payload; Wireshark interprets as much structure as it can.

In the middle panel, expand the Ethernet header fields (using the "+" expander or icon) to see their details. Our interest is the Ethernet header, and you may ignore the higher layer protocols (which are IP and ICMP in this case). You can click on the Ethernet header to see the bytes that correspond to it in the packet highlighted in the bottom panel. We have performed both steps in the figure.

If you are capturing traffic over an 802.11 interface, you may wonder why you have an Ethernet header at all, instead of an 802.11 header. This happens because we asked Wireshark to capture packets in

Ethernet format on the capture options (in Figure 2). In this case, the OS software converted the real 802.11 header into a pseudo-Ethernet header. We are seeing the pseudo-Ethernet header.

Compare the fields you see with the picture of an Ethernet frame in Fig. 4-14 of your text. You will see both similarities and differences:

- There are two kinds of Ethernet shown in your book, IEEE 802.3 and DIX Ethernet. IEEE 802.3 is rare and you are not likely to see it. The frames in the figure and likely your capture are DIX Ethernet, called "Ethernet II" in Wireshark.
- There is no preamble in the fields shown in Wireshark. The preamble is a physical layer mechanism to help the NIC identify the start of a frame. It carries no useful data and is not received like other fields.
- There is a destination address and a source address. Wireshark is decoding some of these bits in the OUI (Organizationally Unique Identifier) portion of the address to tell us the vendor of the NIC, e.g., Dell for the source address.
- There is a Type field. For the ping messages, the Ethernet type is IP, meaning the Ethernet payload carries an IP packet. (There is no Length field as in the IEEE 802.3 format. Instead, the length of a DIX Ethernet frame is determined by the hardware of a receiving computer, which looks for valid frames that start with a preamble and end with a correct checksum, and passed up to higher layers along with the packet.)
- There is no Data field per se – the data starts with the IP header right after the Ethernet header.
- There is no pad. A pad will be present at the end if the frame would otherwise be less than 64 bytes, the minimum Ethernet frame size.
- There is no checksum in most traces, even though it really does exist. Typically, Ethernet hardware that is sending or receiving frames computes or checks this field and adds or strips it. Thus it is simply not visible to the OS or Wireshark in most capture setups.
- There are also no VLAN fields such as the Tag shown in Fig. 4-49. If VLANs are in use, the VLAN tags are normally added and removed by switch ports so they will not be visible at host computers using the network.

Step 3: Ethernet Frame Structure

To show your understanding of the Ethernet frame format, draw a figure of the ping message that shows the position and size in bytes of the Ethernet header fields. Your figure can simply show the frame as a long, thin rectangle. The leftmost fields come first in the packet and are sent on the wire first. On this drawing, show the range of the Ethernet header and the Ethernet payload. Add a dashed box at the end to represent the 4-byte checksum; we know it is there even if Wireshark does not show us this field.

To work out sizes, observe that when you click on a protocol block in the middle panel (the block itself, not the "+" expander) then Wireshark will highlight the bytes it corresponds to in the packet in the lower panel and display the length at the bottom of the window. You may also use the overall packet size shown in the Length column or Frame detail block.

Turn-in: Hand in your drawing of an Ethernet frame.

Step 4: Scope of Ethernet Addresses

Each Ethernet frame carries a source and destination address. One of these addresses is that of your computer. It is the source for frames that are sent, and the destination for frames that are received. But what is the other address? Assuming you pinged a remote Internet server, it cannot be the Ethernet address of the remote server because an Ethernet frame is only addressed to go within one LAN. Instead, it will be the Ethernet address of the router or default gateway, such as your AP in the case of 802.11. This is the device that connects your LAN to the rest of the Internet. In contrast, the IP addresses in the IP block of each packet do indicate the overall source and destination endpoints. They are your computer and the remote server.

Draw a figure that shows the relative positions of your computer, the router, and the remote server. Label your computer and the router with their Ethernet addresses. Label your computer and the remote server with their IP addresses. Show where the Ethernet and the rest of the Internet fit on the drawing.

Turn-in: Hand in your drawing.

Step 5: Broadcast Frames

The trace that you gathered above captured unicast Ethernet traffic sent between a specific source and destination, e.g., your computer to the router. It is also possible to send multicast or broadcast Ethernet traffic, destined for a group of computers or all computers on the Ethernet, respectively. We can tell from the address whether it is unicast, multicast, or broadcast. Broadcast traffic is sent to a reserved Ethernet address that has all bits set to "1". Multicast traffic is sent to addresses that have a "1" in the first bit sent on the wire; broadcast is a special case of multicast. Broadcast and multicast traffic is widely used for discovery protocols, e.g., a packet sent to everyone in an effort to find the local printer.

Start a capture for broadcast and multicast Ethernet frames with a filter of "`ether multicast`", *wait up to 30 seconds to record background traffic, and then stop the capture. If you do not capture any packets with this filter then use the trace that we supplied.* On most Ethernets, there is a steady chatter of background traffic as computers exchange messages to maintain network state, which is why we try to capture traffic without running any other programs. The capture filter of "`ether multicast`" will capture both multicast and broadcast Ethernet frames, but not regular unicast frames. You may have to wait a little while for these packets to be captured, but on most LANs with multiple computers you will see at least a packet every few seconds.

Examine the multicast and broadcast packets that you captured, looking at the details of the source and destination addresses. Most likely one has the broadcast Ethernet address, as broadcast frames tend to be more common than multicast frames. Look at a broadcast frame to see what address is used for broadcast by Ethernet. Expand the Ethernet address fields of either broadcast or multicast frames to see which bit is set to distinguish broadcast/multicast or group traffic from unicast traffic.

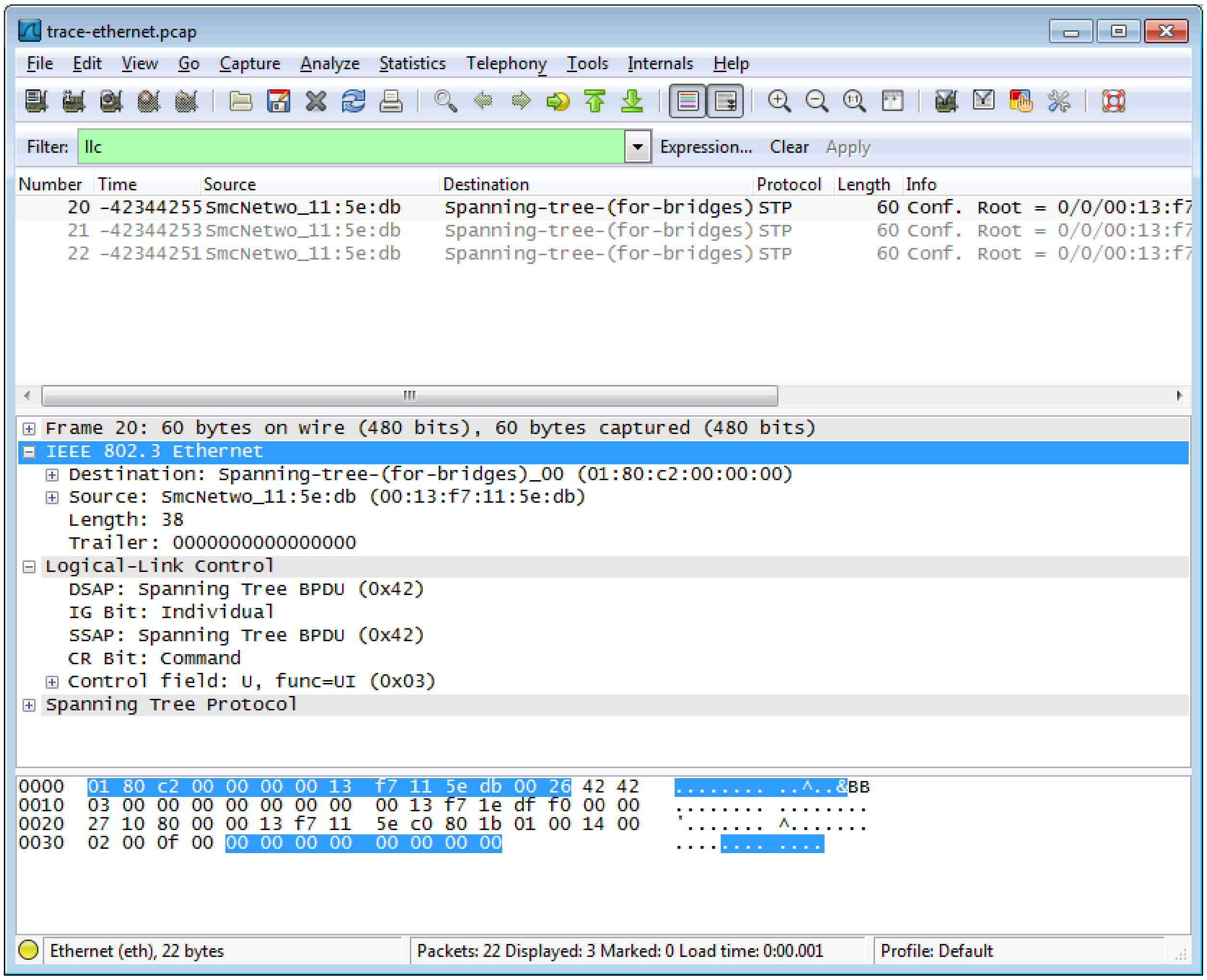
Answer the following questions:

1. *What is the broadcast Ethernet address, written in standard form as Wireshark displays it?*
2. *Which bit of the Ethernet address is used to determine whether it is unicast or multicast/broadcast?*

Turn-in: Hand in your answers to the above questions.

Explore on your own (IEEE 802.3)

We encourage you to explore Ethernet on your own once you have completed this lab. As one possibility, recall that there are two types of Ethernet frame, IEEE 802.3 and DIX Ethernet. DIX is common and what we considered above, while IEEE 802.3 is rare. If you are rather lucky, you may see some IEEE 802.3 frames in the trace you have captured. If not, then there are some of these packets in the trace that we supplied. To search for IEEE 802.3 packets, enter a display filter (above the top panel of the Wireshark window) of “llc” (that was lowercase “LLC”) because the IEEE 802.3 format has the LLC protocol on top of it. LLC is also present on top of IEEE 802.11 wireless, but it is not present on DIX Ethernet.


trace-ethernet.pcap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: llc
Expression... Clear Apply

```
Number  Time       Source              Destination                  Protocol  Length  Info
    20 -42344255 SmcNetwo_11:5e:db    Spanning-tree-(for-bridges) STP           60 Conf. Root = 0/0/00:13:f7
    21 -42344253 SmcNetwo_11:5e:db    Spanning-tree-(for-bridges) STP           60 Conf. Root = 0/0/00:13:f7
    22 -42344251 SmcNetwo_11:5e:db    Spanning-tree-(for-bridges) STP           60 Conf. Root = 0/0/00:13:f7
```

```
⊞ Frame 20: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
⊟ IEEE 802.3 Ethernet
  ⊞ Destination: Spanning-tree-(for-bridges)_00 (01:80:c2:00:00:00)
  ⊞ Source: SmcNetwo_11:5e:db (00:13:f7:11:5e:db)
    Length: 38
    Trailer: 0000000000000000
⊟ Logical-Link Control
    DSAP: Spanning Tree BPDU (0x42)
    IG Bit: Individual
    SSAP: Spanning Tree BPDU (0x42)
    CR Bit: Command
  ⊞ Control field: U, func=UI (0x03)
⊞ Spanning Tree Protocol
```

```
0000  01 80 c2 00 00 00 00 13  f7 11 5e db 00 26 42 42   ........ ..^..&BB
0010  03 00 00 00 00 00 00 00  00 13 f7 1e df f0 00 00   ........ ........
0020  27 10 80 00 00 13 f7 11  5e c0 80 1b 01 00 14 00   '....... ^.......
0030  02 00 0f 00 00 00 00 00  00 00 00 00               ........ ....
```

Ethernet (eth), 22 bytes | Packets: 22 Displayed: 3 Marked: 0 Load time: 0:00.001 | Profile: Default

Figure 4: IEEE 802.3 frames with Ethernet and LLC header detail

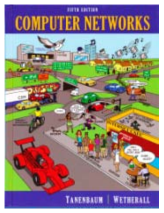
Have a look at the details of an IEEE 802.3 frame, including the LLC header. The figure shows the details for our trace. Observe that the Type field is now a Length field. In our example, the frame is short enough that there is also padding of zeros identified as a Trailer or Padding. The changes lead to a few questions for you to ponder:

1. How long are the combined IEEE 802.3 and LLC headers compared to the DIX Ethernet headers? You can use Wireshark to work this out. Note that the Trailer/Padding and Checksum may be shown as part of the header, but they come at the end of the frame.
2. How does the receiving computer know whether the frame is DIX Ethernet or IEEE 802.3? Hint: you may need to both use Wireshark to look at packet examples and read your text near where the Ethernet formats are described.
3. If IEEE 802.3 has no Type field, then how is the next higher layer determined? Use Wireshark to look for the demultiplexing key.

Turn-in: Your answers to the above questions.

[END]

Lab Exercise – Protocol Layers



Objective

To learn how protocols and layering are represented in packets. They are key concepts for structuring networks that are covered in §1.3 and §1.4 of your text. Review those sections before doing the lab.

Requirements

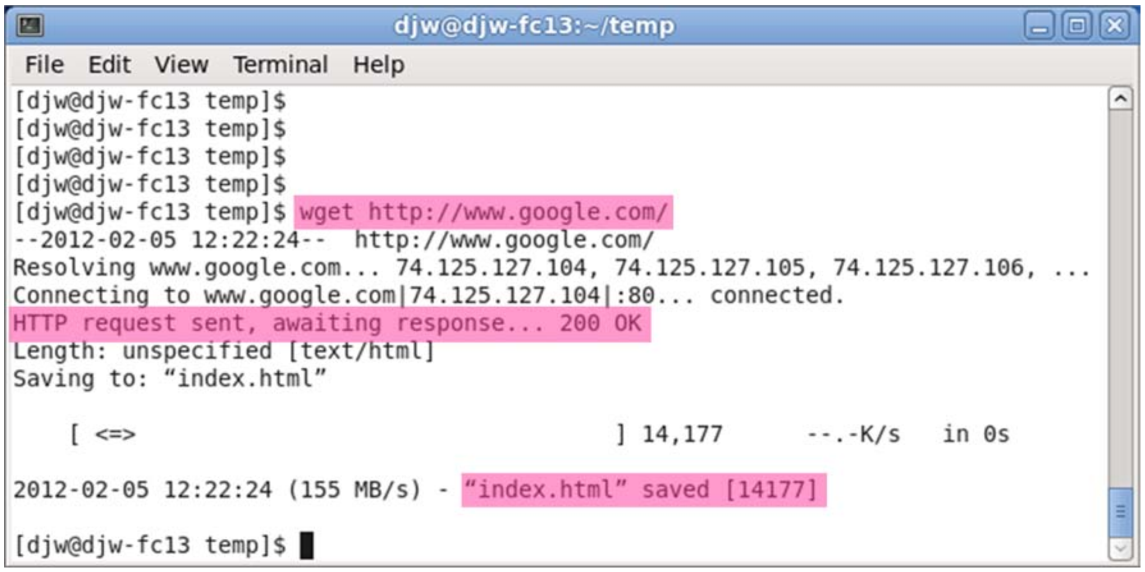
Wireshark: This lab uses the Wireshark software tool to capture and examine a packet trace. A packet trace is a record of traffic at a location on the network, as if a snapshot was taken of all the bits that passed across a particular wire. The packet trace records a timestamp for each packet, along with the bits that make up the packet, from the lower-layer headers to the higher-layer contents. Wireshark runs on most operating systems, including Windows, Mac and Linux. It provides a graphical UI that shows the sequence of packets and the meaning of the bits when interpreted as protocol headers and data. It color-codes packets by their type, and has various ways to filter and analyze packets to let you investigate the behavior of network protocols. Wireshark is widely used to troubleshoot networks. You can download it from www.wireshark.org if it is not already installed on your computer. We highly recommend that you watch the short, 5 minute video "Introduction to Wireshark" that is on the site.

wget / curl: This lab uses `wget` (Linux and Windows) and `curl` (Mac) to fetch web resources. `wget` and `curl` are command-line programs that let you fetch a URL. Unlike a web browser, which fetches and executes entire pages, `wget` and `curl` give you control over exactly which URLs you fetch and when you fetch them. Under Linux, `wget` can be installed via your package manager. Under Windows, `wget` is available as a binary; look for download information on http://www.gnu.org/software/wget/. Under Mac, `curl` comes installed with the OS. Both have many options (try "`wget --help`" or "`curl --help`" to see) but a URL can be fetched simply with "`wget` *URL*" or "`curl` *URL* ".

Step 1: Capture a Trace

Proceed as follows to capture a trace of network traffic; alternatively, you may use a supplied trace. We want this trace to look at the protocol structure of packets. A simple Web fetch of a URL from a server of your choice to your computer, which is the client, will serve as traffic.

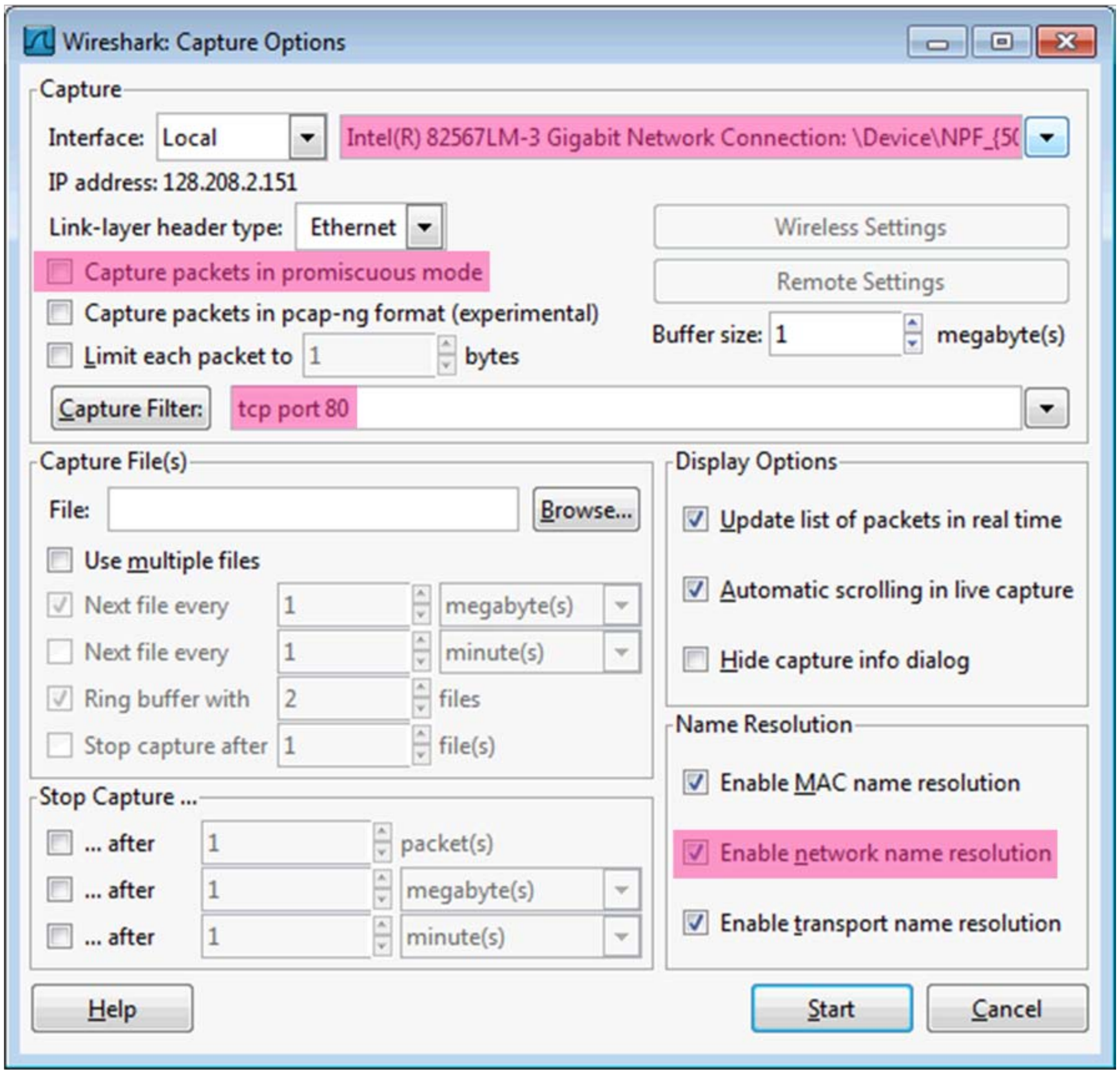
1. *Pick a URL and fetch it with* `wget` *or* `curl`. For example, "`wget http://www.google.com`" or "`curl http://www.google.com`". This will fetch the resource and either write it to a file (`wget`) or to the screen (`curl`). You are checking to see that the fetch works and retrieves some content. A successful example is shown below (with added highlighting) for `wget`. You want a single response with status code "200 OK". If the fetch does not work then try a different URL; if no URLs seem to work then debug your use of `wget/curl` or your Internet connectivity.



```
[djw@djw-fc13 temp]$
[djw@djw-fc13 temp]$
[djw@djw-fc13 temp]$
[djw@djw-fc13 temp]$
[djw@djw-fc13 temp]$ wget http://www.google.com/
--2012-02-05 12:22:24--  http://www.google.com/
Resolving www.google.com... 74.125.127.104, 74.125.127.105, 74.125.127.106, ...
Connecting to www.google.com|74.125.127.104|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: "index.html"

    [ <=>                                  ] 14,177      --.-K/s   in 0s

2012-02-05 12:22:24 (155 MB/s) - "index.html" saved [14177]

[djw@djw-fc13 temp]$ 
```

Figure 1: Using `wget` to fetch a URL

2. *Close unnecessary browser tabs and windows*. By minimizing browser activity you will stop your computer from fetching unnecessary web content, and avoid incidental traffic in the trace.
3. *Launch Wireshark and start a capture with a filter of* "`tcp port 80`" and check "enable network name resolution". This filter will record only standard web traffic and not other kinds of packets that your computer may send. The checking will translate the addresses of the computers sending and receiving packets into names, which should help you to recognize whether the packets are going to or from your computer. Your capture window should be similar to the one pictured below, other than our highlighting. Select the interface from which to capture as the main wired or wireless interface used by your computer to connect to the Internet. If unsure, guess and revisit this step later if your capture is not successful. Uncheck "capture packets in promiscuous mode". This mode is useful to overhear packets sent to/from other computers on broadcast networks. We only want to record packets sent to/from your computer. Leave other options at their default values. The capture filter, if present, is used to prevent the capture of other traffic your computer may send or receive. On Wireshark 1.8, the capture filter box is present directly on the options screen, but on Wireshark 1.9, you set a capture filter by double-clicking on the interface.


Wireshark: Capture Options

Capture

Interface: Local Intel(R) 82567LM-3 Gigabit Network Connection: \Device\NPF_{5(

IP address: 128.208.2.151

Link-layer header type: Ethernet

☐ Capture packets in promiscuous mode

☐ Capture packets in pcap-ng format (experimental)

☐ Limit each packet to 1 bytes

Buffer size: 1 megabyte(s)

Capture Filter: tcp port 80

Capture File(s)

File:

☐ Use multiple files

✓ Next file every 1 megabyte(s)

☐ Next file every 1 minute(s)

✓ Ring buffer with 2 files

☐ Stop capture after 1 file(s)

Stop Capture ...

☐ ... after 1 packet(s)

☐ ... after 1 megabyte(s)

☐ ... after 1 minute(s)

Display Options

✓ Update list of packets in real time

✓ Automatic scrolling in live capture

☐ Hide capture info dialog

Name Resolution

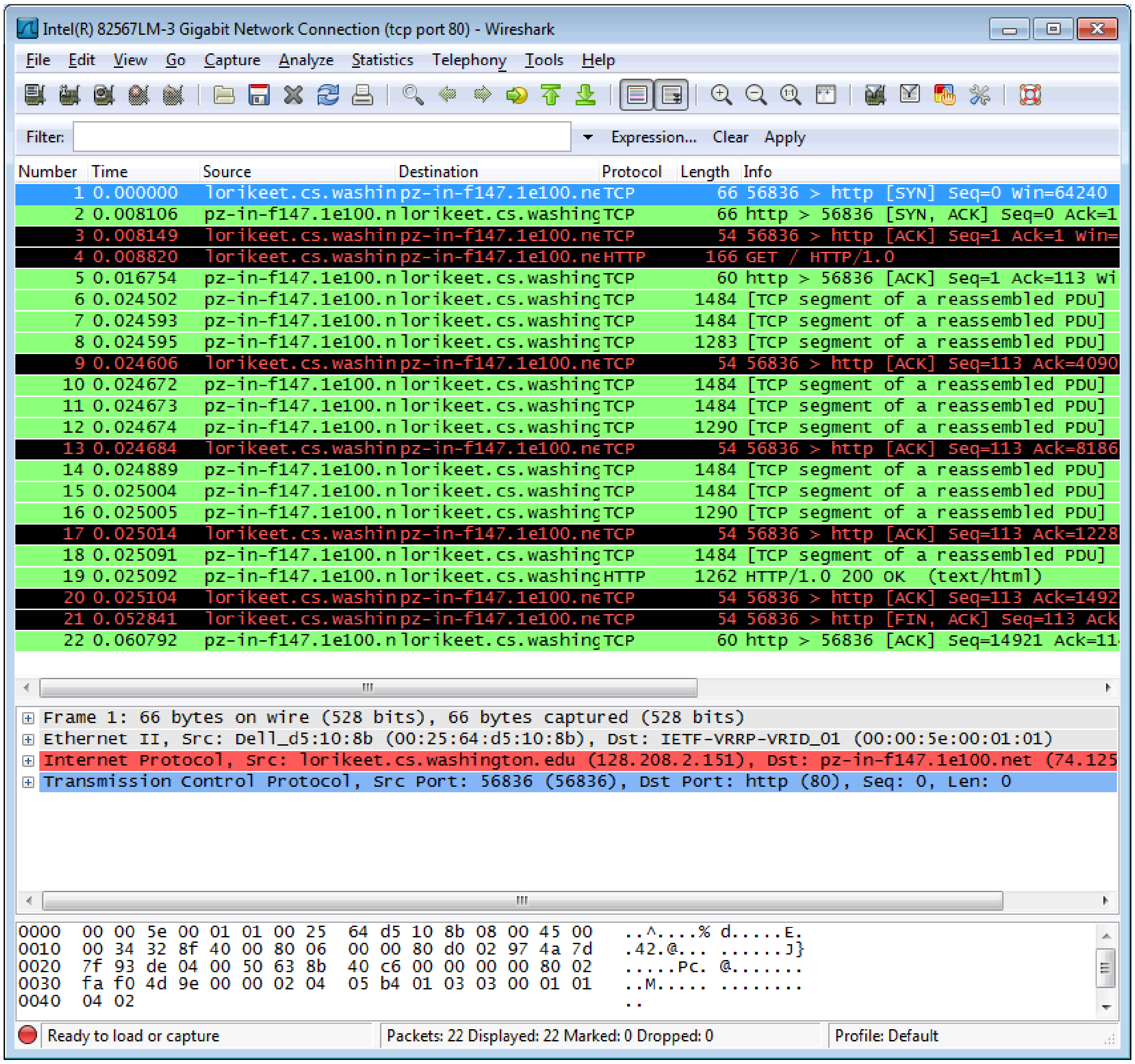
✓ Enable MAC name resolution

✓ Enable network name resolution

✓ Enable transport name resolution

Figure 2: Setting up the capture options

4. *When the capture is started, repeat the web fetch using* `wget/curl` *above.* This time, the packets will be recorded by Wireshark as the content is transferred.
5. *After the fetch is successful, return to Wireshark and use the menus or buttons to stop the trace.* If you have succeeded, the upper Wireshark window will show multiple packets, and most likely it will be full. How many packets are captured will depend on the size of the web page, but there should be at least 8 packets in the trace, and typically 20-100, and many of these packets will be colored green. An example is shown below. Congratulations, you have captured a trace!


Intel(R) 82567LM-3 Gigabit Network Connection (tcp port 80) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
Number Time     Source           Destination      Protocol Length Info
     1 0.000000 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    66 56836 > http [SYN] Seq=0 Win=64240
     2 0.008106 pz-in-f147.1e100.n lorikeet.cs.washing TCP    66 http > 56836 [SYN, ACK] Seq=0 Ack=1
     3 0.008149 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    54 56836 > http [ACK] Seq=1 Ack=1 Win=
     4 0.008820 lorikeet.cs.washin pz-in-f147.1e100.ne HTTP  166 GET / HTTP/1.0
     5 0.016754 pz-in-f147.1e100.n lorikeet.cs.washing TCP    60 http > 56836 [ACK] Seq=1 Ack=113 Wi
     6 0.024502 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
     7 0.024593 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
     8 0.024595 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1283 [TCP segment of a reassembled PDU]
     9 0.024606 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    54 56836 > http [ACK] Seq=113 Ack=4090
    10 0.024672 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
    11 0.024673 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
    12 0.024674 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1290 [TCP segment of a reassembled PDU]
    13 0.024684 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    54 56836 > http [ACK] Seq=113 Ack=8186
    14 0.024889 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
    15 0.025004 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
    16 0.025005 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1290 [TCP segment of a reassembled PDU]
    17 0.025014 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    54 56836 > http [ACK] Seq=113 Ack=1228
    18 0.025091 pz-in-f147.1e100.n lorikeet.cs.washing TCP  1484 [TCP segment of a reassembled PDU]
    19 0.025092 pz-in-f147.1e100.n lorikeet.cs.washing HTTP 1262 HTTP/1.0 200 OK  (text/html)
    20 0.025104 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    54 56836 > http [ACK] Seq=113 Ack=1492
    21 0.052841 lorikeet.cs.washin pz-in-f147.1e100.ne TCP    54 56836 > http [FIN, ACK] Seq=113 Ack
    22 0.060792 pz-in-f147.1e100.n lorikeet.cs.washing TCP    60 http > 56836 [ACK] Seq=14921 Ack=11
```

```
⊞ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
⊞ Ethernet II, Src: Dell_d5:10:8b (00:25:64:d5:10:8b), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
⊞ Internet Protocol, Src: lorikeet.cs.washington.edu (128.208.2.151), Dst: pz-in-f147.1e100.net (74.125
⊞ Transmission Control Protocol, Src Port: 56836 (56836), Dst Port: http (80), Seq: 0, Len: 0
```

```
0000  00 00 5e 00 01 01 00 25  64 d5 10 8b 08 00 45 00   ..^....% d.....E.
0010  00 34 32 8f 40 00 80 06  00 00 80 d0 02 97 4a 7d   .42.@... ......J}
0020  7f 93 de 04 00 50 63 8b  40 c6 00 00 00 00 80 02   .....Pc. @.......
0030  fa f0 4d 9e 00 00 02 04  05 b4 01 03 03 00 01 01   ..M..... ........
0040  04 02                                              ..
```

Figure 3: Packet trace of `wget` traffic

Step 2: Inspect the Trace

Wireshark will let us select a packet (from the top panel) and view its protocol layers, in terms of both header fields (in the middle panel) and the bytes that make up the packet (in the bottom panel). In the figure above, the first packet is selected (shown in blue). Note that we are using "packet" as a general term here. Strictly speaking, a unit of information at the link layer is called a frame. At the network layer it is called a packet, at the transport layer a segment, and at the application layer a message. Wireshark is gathering frames and presenting us with the higher-layer packet, segment, and message structures it can recognize that are carried within the frames. We will often use "packet" for convenience, as each frame contains one packet and it is often the packet or higher-layer details that are of interest.

Select a packet for which the Protocol column is "HTTP" and the Info column says it is a GET. It is the packet that carries the web (HTTP) request sent from your computer to the server. (You can click the column headings to sort by that value, though it should not be difficult to find an HTTP packet by inspection.) Let's have a closer look to see how the packet structure reflects the protocols that are in use.

Since we are fetching a web page, we know that the protocol layers being used are as shown below. That is, HTTP is the application layer web protocol used to fetch URLs. Like many Internet applications, it runs on top of the TCP/IP transport and network layer protocols. The link and physical layer protocols depend on your network, but are typically combined in the form of Ethernet (shown) if your computer is wired, or 802.11 (not shown) if your computer is wireless.

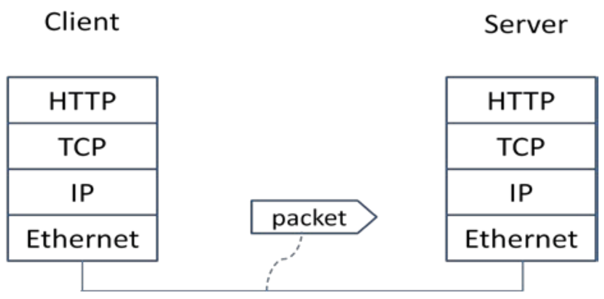


Figure 4: Protocol stack for a web fetch

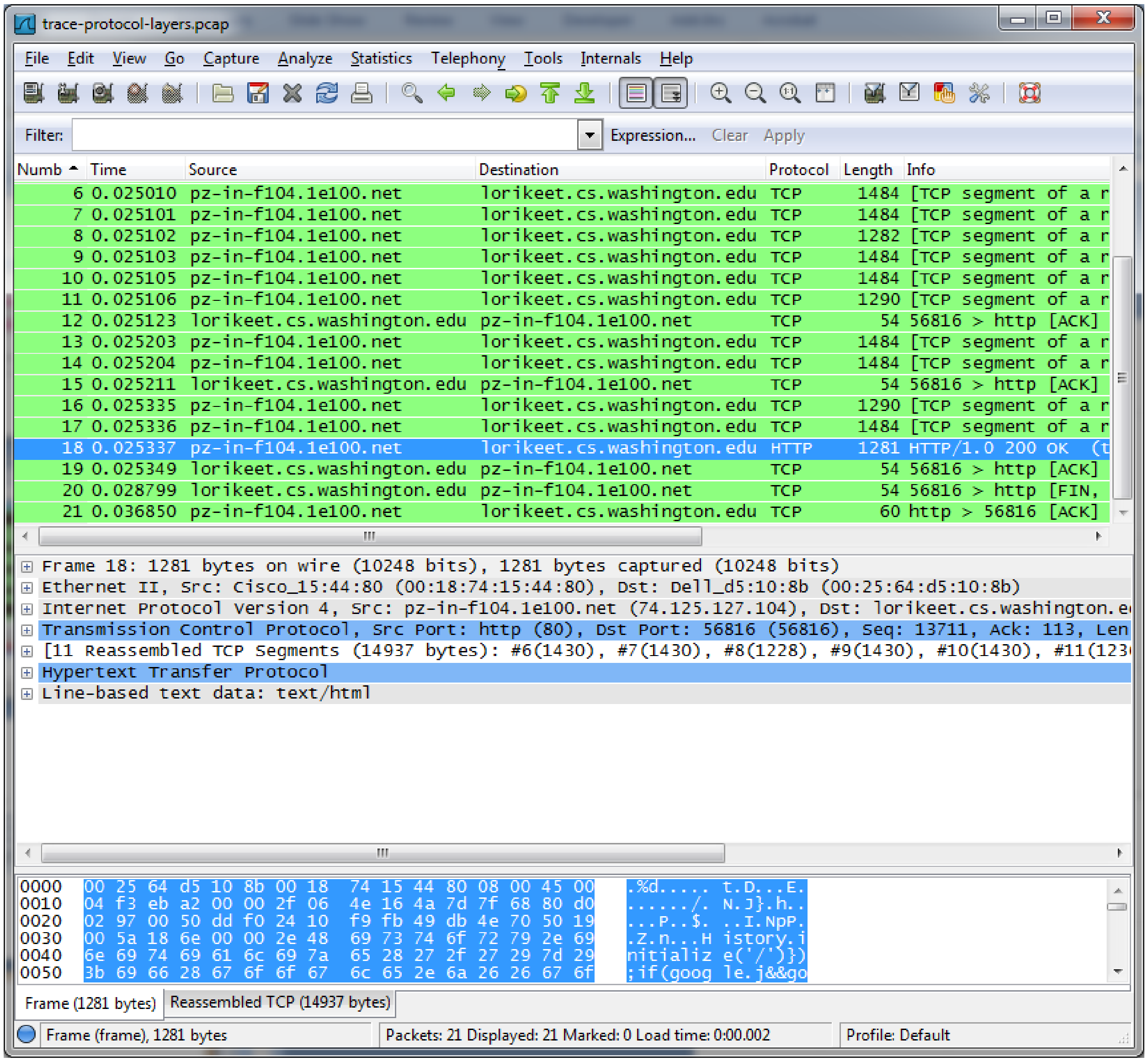
With the HTTP GET packet selected, look closely to see the similarities and differences between it and our protocol stack as described next. The protocol blocks are listed in the middle panel. You can expand each block (by clicking on the "+" expander or icon) to see its details.

- The first Wireshark block is "Frame". This is not a protocol, it is a record that describes overall information about the packet, including when it was captured and how many bits long it is.
- The second block is "Ethernet". This matches our diagram! Note that you may have taken a trace on a computer using 802.11 yet still see an Ethernet block instead of an 802.11 block. Why? It happens because we asked Wireshark to capture traffic in Ethernet format on the capture options, so it converted the real 802.11 header into a pseudo-Ethernet header.
- Then come IP, TCP, and HTTP, which are just as we wanted. Note that the order is from the bottom of the protocol stack upwards. This is because as packets are passed down the stack, the header information of the lower layer protocol is added to the front of the information from the higher layer protocol, as in Fig. 1-15 of your text. That is, the lower layer protocols come first in the packet "on the wire".

Now find another HTTP packet, the response from the server to your computer, and look at the structure of this packet for the differences compared to the HTTP GET packet. This packet should have "200 OK" in the Info field, denoting a successful fetch. In our trace, there are two extra blocks in the detail panel as seen in the next figure.

- The first extra block says "[11 reassembled TCP segments …]". Details in your capture will vary, but this block is describing more than the packet itself. Most likely, the web response was sent across the network as a series of packets that were put together after they arrived at the computer. The packet labeled HTTP is the last packet in the web response, and the block lists packets that are joined together to obtain the complete web response. Each of these packets is shown as having protocol TCP even though the packets carry part of an HTTP response. Only the final packet is shown as having protocol HTTP when the complete HTTP message may be understood, and it lists the packets that are joined together to make the HTTP response.

- The second extra block says "Line-based text data …". Details in your capture will vary, but this block is describing the contents of the web page that was fetched. In our case it is of type text/html, though it could easily have been text/xml, image/jpeg, or many other types. As with the Frame record, this is not a true protocol. Instead, it is a description of packet contents that Wireshark is producing to help us understand the network traffic.


trace-protocol-layers.pcap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply

```
Numb  Time      Source                      Destination                 Protocol Length Info
   6 0.025010  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
   7 0.025101  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
   8 0.025102  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1282 [TCP segment of a r
   9 0.025103  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
  10 0.025105  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
  11 0.025106  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1290 [TCP segment of a r
  12 0.025123  lorikeet.cs.washington.edu  pz-in-f104.1e100.net        TCP        54 56816 > http [ACK]
  13 0.025203  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
  14 0.025204  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
  15 0.025211  lorikeet.cs.washington.edu  pz-in-f104.1e100.net        TCP        54 56816 > http [ACK]
  16 0.025335  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1290 [TCP segment of a r
  17 0.025336  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP      1484 [TCP segment of a r
  18 0.025337  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  HTTP     1281 HTTP/1.0 200 OK  (t
  19 0.025349  lorikeet.cs.washington.edu  pz-in-f104.1e100.net        TCP        54 56816 > http [ACK]
  20 0.028799  lorikeet.cs.washington.edu  pz-in-f104.1e100.net        TCP        54 56816 > http [FIN,
  21 0.036850  pz-in-f104.1e100.net        lorikeet.cs.washington.edu  TCP        60 http > 56816 [ACK]
```

```
⊞ Frame 18: 1281 bytes on wire (10248 bits), 1281 bytes captured (10248 bits)
⊞ Ethernet II, Src: Cisco_15:44:80 (00:18:74:15:44:80), Dst: Dell_d5:10:8b (00:25:64:d5:10:8b)
⊞ Internet Protocol Version 4, Src: pz-in-f104.1e100.net (74.125.127.104), Dst: lorikeet.cs.washington.e
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 56816 (56816), Seq: 13711, Ack: 113, Len
⊞ [11 Reassembled TCP Segments (14937 bytes): #6(1430), #7(1430), #8(1228), #9(1430), #10(1430), #11(123
⊞ Hypertext Transfer Protocol
⊞ Line-based text data: text/html
```

```
0000   00 25 64 d5 10 8b 00 18  74 15 44 80 08 00 45 00   .%d..... t.D...E.
0010   04 f3 eb a2 00 00 2f 06  4e 16 4a 7d 7f 68 80 d0   ....../. N.J}.h..
0020   02 97 00 50 dd f0 24 10  f9 fb 49 db 4e 70 50 19   ...P..$. ..I.NpP.
0030   00 5a 18 6e 00 00 2e 48  69 73 74 6f 72 79 2e 69   .Z.n...H istory.i
0040   6e 69 74 69 61 6c 69 7a  65 28 27 2f 27 29 7d 29   nitializ e('/')})
0050   3b 69 66 28 67 6f 6f 67  6c 65 2e 6a 26 26 67 6f   ;if(goog le.j&&go
```

Frame (1281 bytes)
Reassembled TCP (14937 bytes)
Frame (frame), 1281 bytes
Packets: 21 Displayed: 21 Marked: 0 Load time: 0:00.002
Profile: Default

Figure 5: Inspecting a HTTP "200 OK" response

Step 3: Packet Structure

To show your understanding of packet structure, draw a figure of an HTTP GET packet that shows the position and size in bytes of the TCP, IP and Ethernet protocol headers. Your figure can simply show the overall packet as a long, thin rectangle. Leftmost elements are the first sent on the wire. On this drawing, show the range of the Ethernet header and the Ethernet payload that IP passed to Ethernet to send over the network. To show the nesting structure of protocol layers, note the range of the IP header and the IP payload. You may have questions about the fields in each protocol as you look at them. We will explore these protocols and fields in detail in future labs.

To work out sizes, observe that when you click on a protocol block in the middle panel (the block itself, not the "+" expander) then Wireshark will highlight the bytes it corresponds to in the packet in the lower panel and display the length at the bottom of the window. For instance, clicking on the IP version 4 header of a packet in our trace shows us that the length is 20 bytes. (Your trace will be different if it is IPv6, and may be different even with IPv4 depending on various options.) You may also use the overall packet size shown in the Length column or Frame detail block.

Turn-in: Hand in your packet drawing.

Step 4: Protocol Overhead

Estimate the download protocol overhead, or percentage of the download bytes taken up by protocol overhead. To do this, consider HTTP data (headers and message) to be useful data for the network to carry, and lower layer headers (TCP, IP, and Ethernet) to be the overhead. We would like this overhead to be small, so that most bits are used to carry content that applications care about. To work this out, first look at only the packets in the download direction for a single web fetch. You might sort on the Destination column to find them. The packets should start with a short TCP packet described as a SYN ACK, which is the beginning of a connection. They will be followed by mostly longer packets in the middle (of roughly 1 to 1.5KB), of which the last one is an HTTP packet. This is the main portion of the download. And they will likely end with a short TCP packet that is part of ending the connection. For each packet, you can inspect how much overhead it has in the form of Ethernet / IP / TCP headers, and how much useful HTTP data it carries in the TCP payload. You may also look at the HTTP packet in Wireshark to learn how much data is in the TCP payloads over all download packets.

Turn-in: Your estimate of download protocol overhead as defined above. Tell us whether you find this overhead to be significant.

Step 5: Demultiplexing Keys

When an Ethernet frame arrives at a computer, the Ethernet layer must hand the packet that it contains to the next higher layer to be processed. The act of finding the right higher layer to process received packets is called demultiplexing. We know that in our case the higher layer is IP. But how does the Ethernet protocol know this? After all, the higher-layer could have been another protocol entirely (such as ARP). We have the same issue at the IP layer – IP must be able to determine that the contents of IP message is a TCP packet so that it can hand it to the TCP protocol to process. The answer is that protocols use information in their header known as a "demultiplexing key" to determine the higher layer.

Look at the Ethernet and IP headers of a download packet in detail to answer the following questions:

1. *Which Ethernet header field is the demultiplexing key that tells it the next higher layer is IP? What value is used in this field to indicate "IP"?*
2. *Which IP header field is the demultiplexing key that tells it the next higher layer is TCP? What value is used in this field to indicate "TCP"?*

Turn-in: Hand in your answers to the above questions.

Explore on your own

We encourage you to explore protocols and layering once you have completed this lab. Some ideas:

- Look at a short TCP packet that carries no higher-layer data. To what entity is this packet destined? After all, if it carries no higher-layer data then it does not seem very useful to a higher layer protocol such as HTTP!
- In a classic layered model, one message from a higher layer has a header appended by the lower layer and becomes one new message. But this is not always the case. Above, we saw a trace in which the web response (one HTTP message comprised of an HTTP header and an HTTP payload) was converted into multiple lower layer messages (being multiple TCP packets). Imagine that you have drawn the packet structure (as in step 2) for the first and last TCP packet carrying the web response. How will the drawings differ?
- In the classic layered model described above, lower layers append headers to the messages passed down from higher layers. How will this model change if a lower layer adds encryption?
- In the classic layered model described above, lower layers append headers to the messages passed down from higher layers. How will this model change if a lower layer adds compression?

[END]