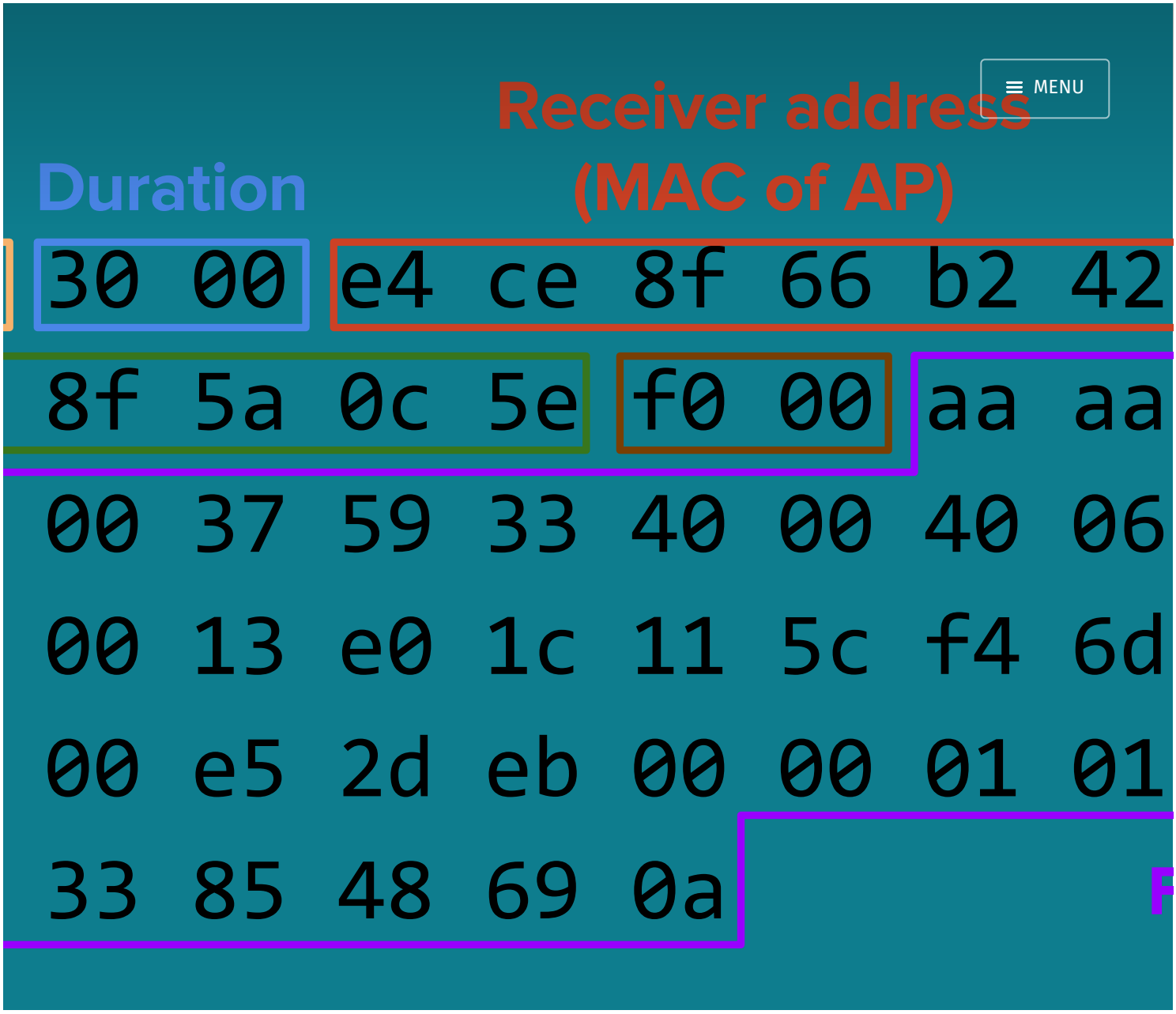


Understanding the 802.11 Wireless LAN MAC frame format

Fraida Fund

20 MARCH 2017 on education, wireless, layer 2, 802.11

In this experiment, we will capture traffic on an 802.11 network and observe the 802.11 MAC frame fields using Wireshark. You will have to download and install Wireshark to run this experiment. You should do this *before* your reservation on the wireless testbed.

It should take about 60-120 minutes to run this experiment, but you will need to have reserved that time in advance. This experiment uses wireless resources, and you can only use wireless resources on GENI during a reservation.

To reproduce this experiment on GENI, you will need an account on the GENI Portal, and you will need to have joined a project. You should have already uploaded your SSH keys to the portal. The project lead of the project you belong to must have enabled wireless for the project. Finally, you must have reserved time on the WITest testbed, and you must run this experiment during your reserved time. (Alternatively, you can use the "outdoor" or "sb4" testbeds at ORBIT, with some modifications to the instructions.)

- Skip to Results
- Skip to Run my experiment

Background

Figure 1 shows the 802.11 MAC frame format:

Octets	2	2	6	6	6	2	6	0 to 2312	4
	FC	D/I	Address	Address	Address	SC	Address	Frame body	CRC

FC = Frame control
D/I = Duration/connection ID
SC = Sequence control

Figure 1: IEEE 802.11 MAC frame format. Image from William Stallings "Data and Computer Communications".

The following excerpt from William Stallings "Data and Computer Communications" explains these fields:

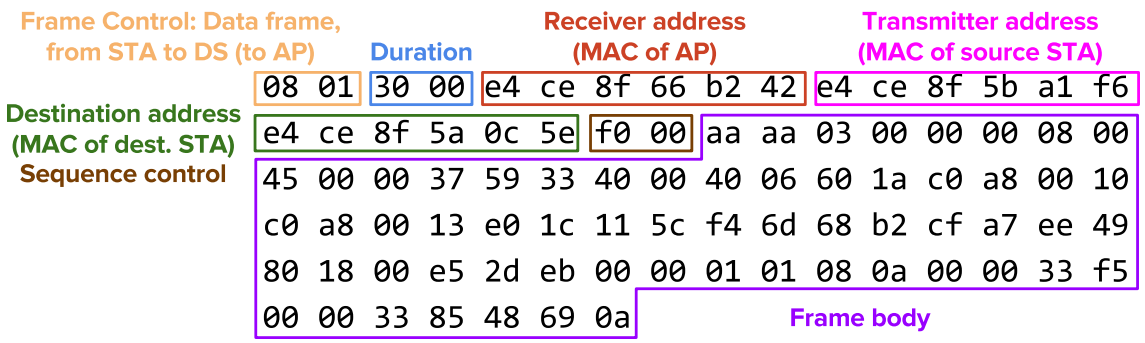
> - **Frame Control**: Indicates the type of frame (control, management, or data) and provides control information. Control information includes whether the frame is to or from a DS, fragmentation information, and privacy information.
> - **Duration/Connection ID**: If used as a duration field, indicates the time (in microseconds) the channel will be allocated for successful transmission of a MAC frame. In some control frames, this field contains an association, or connection, identifier.
> - **Addresses**: The number and meaning of the 48-bit address fields depend on context. The transmitter address and receiver address are the MAC addresses of stations joined to the BSS that are transmitting and receiving frames over the wireless LAN. The service set ID (SSID) identifies the wireless LAN over
> which a frame is transmitted. For an IBSS, the SSID is a random number gen-
> erated at the time the network is formed. For a wireless LAN that is part of a
> larger configuration the SSID identifies the BSS over which the frame is trans-
> mitted; specifically, the SSID is the MAC-level address of the AP for this BSS
> (Figure 17.4). Finally the source address and destination address are the MAC addresses of stations, wireless or otherwise, that are the ultimate source and
> destination of this frame. The source address may be identical to

the transmit-
ter address and the destination address may be identical to the receiver
address.

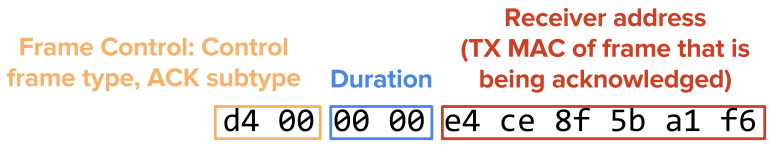
- **Sequence Control**: Contains a 4-bit fragment number subfield, used for frag- mentation and reassembly, and a 12-bit sequence number used to number
frames sent between a given transmitter and receiver.
- **Frame Body**: Contains an MSDU or a fragment of an MSDU. The MSDU is a LLC protocol data unit or MAC control information.
- **Frame Check Sequence**: A 32-bit cyclic redundancy check.

Results

In this experiment, we identify the 802.11 frame fields from the raw hex dump of a captured packet:



We also observe the acknowledgment of this frame:



as well as the second transmission of the frame (from AP to destination STA) and its acknowledgment.

Run my experiment

In this experiment, I used the WITest wireless testbed. To reserve time on this testbed, log in to https://witestlab.poly.edu by clicking on the user icon in the top right corner and then choosing "Log in with GENI". Click on "Testbed" and "Make a reservation", then click on the grid squares corresponding to the date/time you want. For more information, see the WITest reservation tutorial.

At your reserved time, SSH into the WITest console (`witestlab.poly.edu`) using your GENI Wireless username and keys associated with that account.

(Alternatively, you can use the "outdoor" or "sb4" testbeds at ORBIT, with some modifications to the instructions.)

Prepare the testbed

First, load the `wifi-experiment.ndz` disk image onto four testbed nodes: node16, node17, node18, and node19. From the WITest console, run:

```
omf-5.4 load -i wifi-experiment.ndz -t
omf.witest.node16,omf.witest.node17,omf.witest.node18,omf.witest.node
19
```

(Note that the command above is all one line, and there are no spaces between the commas and resource names.)

When this process finishes successfully, turn the nodes on with

```
omf tell -a on -t
omf.witest.node16,omf.witest.node17,omf.witest.node18,omf.witest.node
19
```

Wait a few minutes for the nodes to boot. Then, open four terminals. In each, SSH into the WITest console (with your GENI wireless username and keys), and from there, into each of the nodes (with username "root").

Set up the access point

One node is designated to act as the wireless access point. I used node17 for this. On the terminal of this node, run

```
ifconfig wlan0 up
```

to make sure the wireless interface is up. Then, bring up an access point on channel 11 with

```
create_ap -c 11 -n wlan0 witestlab-exp
```

and leave this running.

Set up STA connectivity

Two of the nodes are designated as wireless station devices. (I used node16 and node19 for this.) On these, you will need to connect to the wireless network and set up an IP address.

First, run

```
ifconfig wlan0 up
```

to bring up the wireless interface.

Then, connect to the wireless access point with ESSID "witestlab" that is operating on channel 11:

```
iwconfig wlan0 essid witestlab-exp channel 11
```

To verify that you are connected, use

```
iwconfig wlan0
```

The output of this command should look something like this:

```
wlan0     IEEE 802.11abgn  ESSID:"witestlab-exp"
          Mode:Managed  Frequency:2.462 GHz  Access Point:
E4:CE:8F:66:B2:42
          Bit Rate=18 Mb/s   Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=70/70  Signal level=-32 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:7   Missed beacon:0
```

Make a note of the MAC address of the access point, which is also the BSSID. Here, it is E4:CE:8F:66:B2:42.

Finally, set an IP address on the wireless interface with

```
ifconfig wlan0 192.168.0.X
```

where "X" is the node number (e.g. 16 on node16, etc.). Verify that they can reach one another over the wireless network, e.g. on node16 run

```
ping -c 1 192.168.0.19
```

Also use

```
ifconfig wlan0
```

to find out and make a note of the MAC address of each wireless interface.

Set up monitoring

Next, we will set up the fourth node (I used node18 for this) to monitor 802.11 network and capture frames that are transmitted on the air. We will use the Aircrack software platform for this.

First, make sure the wireless interface is *down*:

```
ifconfig wlan0 down
```

Wait a few seconds, then set up a "monitor" interface called "mon0" with

```
airmon-ng start wlan0
```

and then run (replacing the bold part as required):

```
airodump-ng -c 11 --bssid E4:CE:8F:66:B2:42 --output-format pcap -w wla
```

to begin capturing traffic on that interface. (Note that this all one command, to run on one line.) Here,

- `-c 11` says to capture on channel 11,
- `--bssid E4:CE:8F:66:B2:42` is the BSSID of the wireless access point that we want to capture (see `iwconfig wlan0` output in previous section. Use the BSSID of *your* network that you observed, which may be different from mine),
- `--output-format pcap -w wlan-capture` specifies the format to capture in, and a prefix to use for the output file name,
- `mon0` is the interface that we will listen on.

Leave that running.

Send traffic over the network

Now, we will send some traffic over the network.

(I used node19 as my server and node16 as my client, but you can make substitutions as necessary if you are using other nodes.)

On node19, run the `netcat` application to receive incoming traffic (here, on port 4444):

```
netcat -l 4444
```

On node16, connect to that `netcat` instance by specifying the IP address of node19, and the port on which you are running the `netcat` server:

```
netcat 192.168.0.19 4444
```

Now, send a few characters in each direction and make sure you see it mirrored on the other end. (You should some activity in the monitoring window, too, as some packets are captured.) Then use Ctrl+C to stop the `netcat` processes and also the `airodump-ng` traffic capture.

On the monitoring node, use `ls` to verify that there is a new capture file. The capture file should have the prefix "wlan-capture", followed by a number that is incremented each time you repeat this process, e.g. "wlan-capture-01.cap".

Here's a video of this part of the experiment:

Analyze captures

Next, we will analyze our packet capture using Wireshark. Wireshark is a free network protocol analyzer that is available for Windows, Mac, and Linux operating systems. Download Wireshark and install it on your own PC or laptop.

You'll need to transfer the capture file from the testbed to your laptop.

On the monitor node, use `ls` to identify the name of the capture file you want to transfer. You will use `scp` to transfer it.

Assuming the file is "wlan-capture-01.cap", on your laptop, run

```
scp -o "StrictHostKeyChecking no"  -o "ProxyCommand ssh -i /PATH/TO/KEY
```

(but subtitute your own username where it says `GENI-WIRELESS-USERNAME`, and the path to your own key where it says `/PATH/TO/KEY`). Note that this is all one command, on one line.

Note: In general, the syntax of `scp` is

```
scp [OPTIONS] source destination
```

Here,

- The `[OPTIONS]`, which in this case allow you to "hop" through to the node via the testbed console, are:

```
-o "StrictHostKeyChecking no" -o "ProxyCommand ssh -i /PATH/TO/KEY GENI
```

- The source file is (possibly with a different file name or node name):

```
root@node18:/root/wlan-capture-01.cap
```

- The destination is `.`, the bash shortcut that indicates "put the file here, in my current working directory":

```
.
```

After this file is transferred to your laptop, you can open it in Wireshark. Once you have it open, make some changes to the way it is displayed:

- In the menu, choose "View" > "Name Resolution" and make sure all are *un*-checked.
- Also under "View", turn off "Colorize Packet List".

Near the top of the Wireshark display, there is a line where you can enter a display filter. Enter the filter

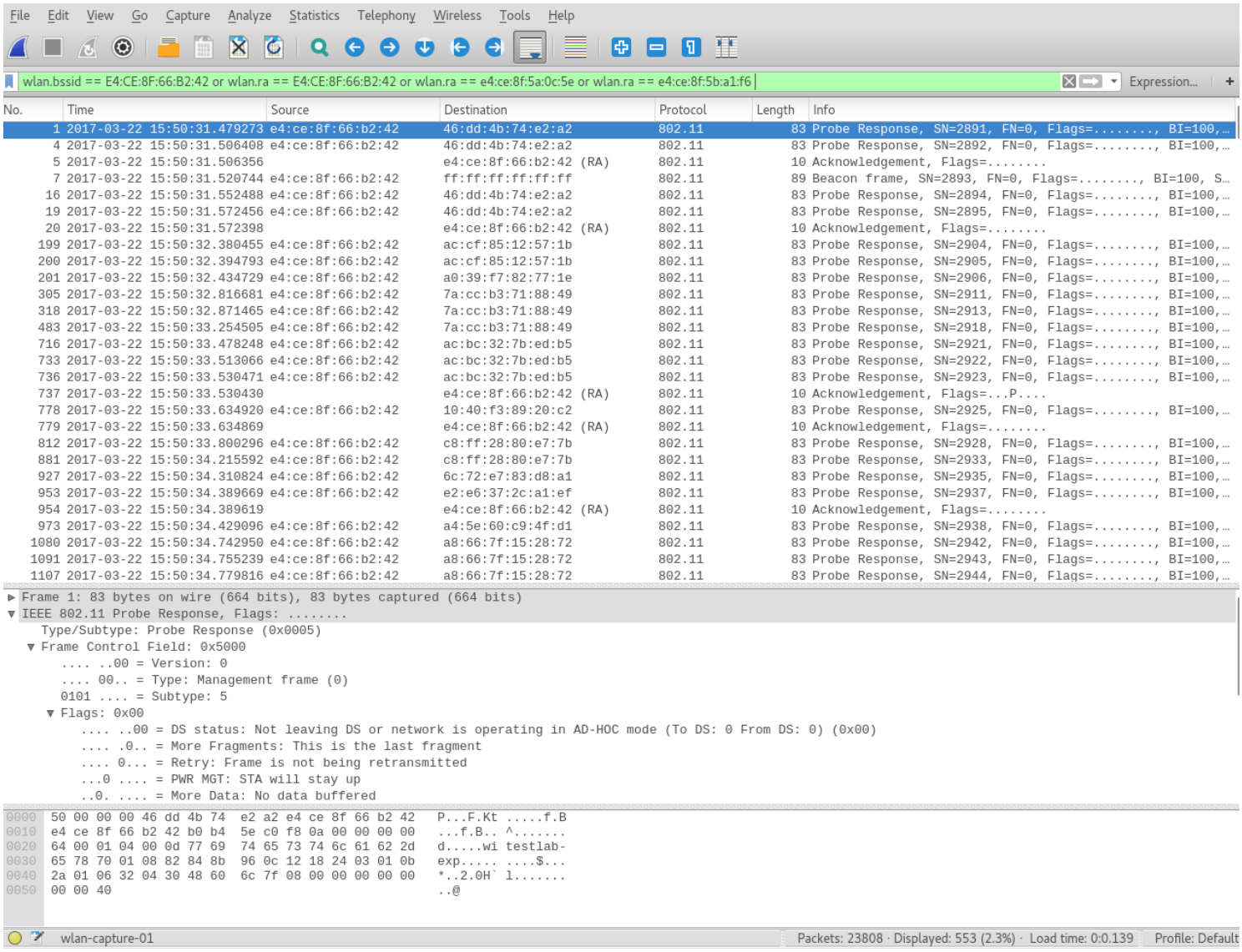
```
wlan.bssid == E4:CE:8F:66:B2:42 or wlan.ra == E4:CE:8F:66:B2:42 or wlan
```

where the last two MAC addresses are the MAC addresses of your two wireless stations (here, `e4:ce:8f:5a:0c:5e` and `e4:ce:8f:5b:a1:f6`) and the first two are the MAC address of your access point.

Then hit "Enter" or "Return" to filter your capture to the frames of interest:

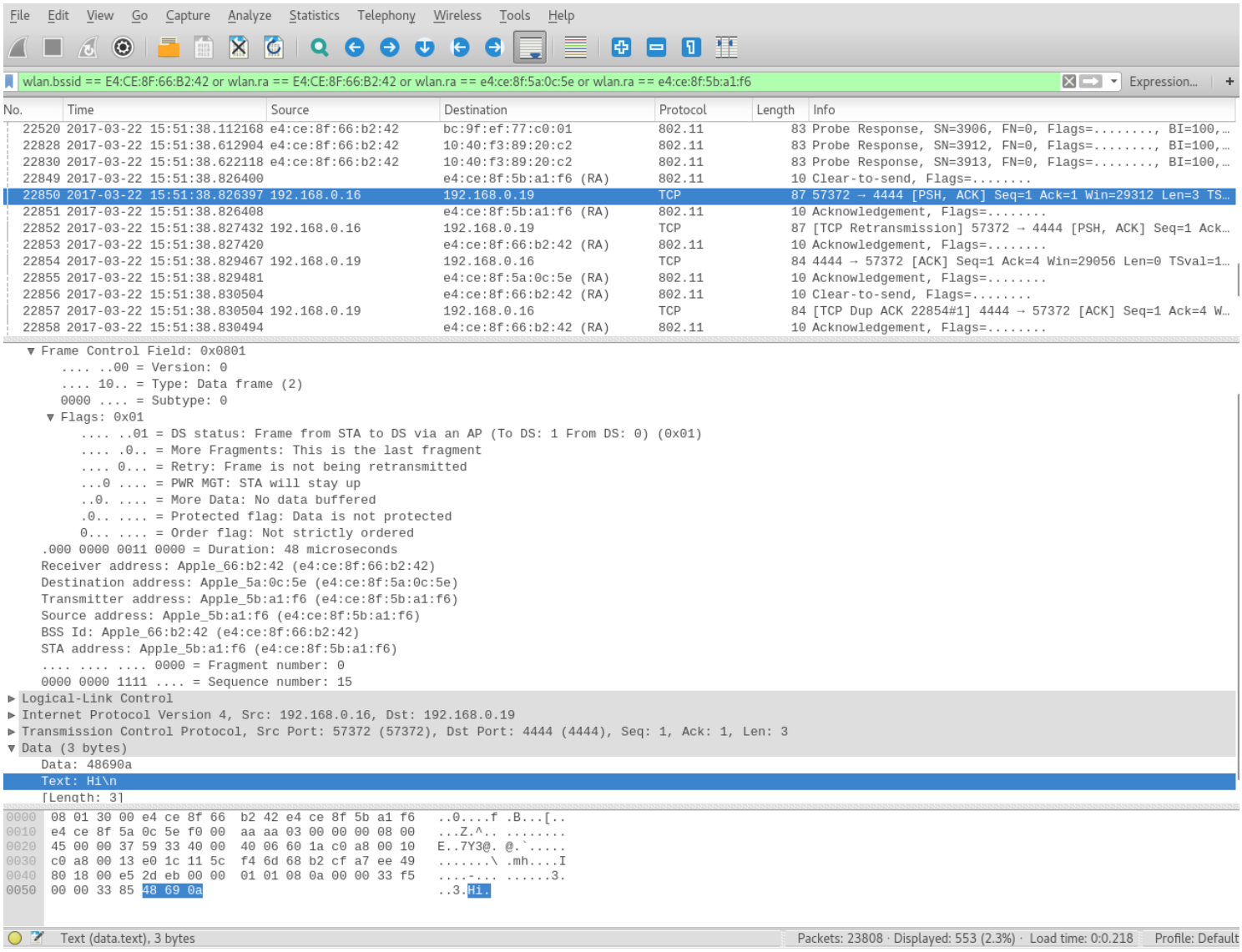
- those with the value `E4:CE:8F:66:B2:42` in the "BSSID" frame field, or
- those with either `E4:CE:8F:66:B2:42` (the MAC address of the AP), `e4:ce:8f:5a:0c:5e` or `e4:ce:8f:5b:a1:f6` (the MAC address of one of the two wireless stations) in the "Receiver Address" frame field.

The filter bar should turn green if your syntax is correct:



In Wireshark, the top part of the window will show the (ordered) list of captured packets. In the middle part, you can see details of the selected packet, including the headers at each layer of the network stack. At the bottom, you will see the complete packet contents, in both hex format (on the left) and ASCII (on the right).

Scroll through the packet list and find the (first) packet that carries the message you sent with `netcat`. You should be able to actually see the text of your message:



We are going to trace this message from the time that it is transmitted, until it is received.

In the middle pane, expand the "IEEE 802.11" header section to see the details of the 802.11 frame header. We will go through them one at a time.

Note: If you can't find the captured message, it may not have been picked up by the monitor. Go back to the Set up monitoring section of these instructions and repeat the steps from there on.

Frame control field

The "type" in the "Frame Control" field indicates that the frame is a "Data" frame (as opposed to "Control" or "Management".

```
Frame Control Field: 0x0801
    .... ..00 = Version: 0
    .... 10.. = Type: Data frame (2)
    0000 .... = Subtype: 0
```

The flags in the "Frame Control" field show that

- the frame is being transmitted *from* a wireless station to the distribution system (DS) through the AP.
- the frame is not one of the first of a set of fragments of higher-layer PDU that has been split up into multiple frames.
- the frame is not a retransmission of a frame that was already transmitted, but not acknowledged.

```
Flags: 0x01
            .... ..01 = DS status: Frame from STA to DS via an AP (To
DS: 1 From DS: 0) (0x01)
    .... .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
   .0.. .... = Protected flag: Data is not protected
   0... .... = Order flag: Not strictly ordered```
```

In the hex dump, these binary flags - the bits 0000 1000 0000 0001 - are visible as hex digits 08 01:

```
08 01 30 00 e4 ce 8f 66 b2 42 e4 ce 8f 5b a1 f6
e4 ce 8f 5a 0c 5e f0 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

Duration field

Next, we see that 48 microseconds have been allocated for this transmission:

```
.000 0000 0011 0000 = Duration: 48 microseconds
```

We can also pick this out in the hex dump - 48 in decimal is 30 00 in hex:

```
08 01 30 00 e4 ce 8f 66 b2 42 e4 ce 8f 5b a1 f6
e4 ce 8f 5a 0c 5e f0 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

Addresses

Next, we can pick out the MAC addresses in the hex dump. In data frames, and when the network is in infrastructure mode, the address

fields depend on the value of the DS flags in the frame control field:

	To DS bit	From DS bit	Address 1 (receiver)	Address 2 (transmitter)	Address 3	Address 4
To AP	1	0	BSSID (MAC of AP)	MAC address of transmitting STA	MAC address of destination STA	Not used
From AP	0	1	MAC address of destination STA	BSSID (MAC of AP)	MAC address of source STA	Not used

Recall that for this frame, the To DS bit was 1 and the From DS bit was 0, corresponding to the first row in this table.

First, we see the receiver address, which is the MAC address of the device where this frame will be received next. In this case, the frame is sent from one station to another via the AP, so the receiver address is the MAC address of the AP:

```
08 01 30 00 e4 ce 8f 66 b2 42 e4 ce 8f 5b a1 f6
e4 ce 8f 5a 0c 5e f0 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

Next, we see the transmitter address, which is the MAC address of the wireless station that sent the frame:

```
08 01 30 00 e4 ce 8f 66 b2 42 e4 ce 8f 5b a1 f6
e4 ce 8f 5a 0c 5e f0 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

This is followed by the destination address. The destination address is the MAC address of the wireless station to which this frame should eventually be delivered:

```
08 01 30 00 e4 ce 8f 66 b2 42 e4 ce 8f 5b a1 f6
e4 ce 8f 5a 0c 5e f0 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

Wireshark helpfully summarizes the addresses that can be inferred from these fields:

```
Receiver address: e4:ce:8f:66:b2:42
Destination address: e4:ce:8f:5a:0c:5e
Transmitter address: e4:ce:8f:5b:a1:f6
Source address: e4:ce:8f:5b:a1:f6
BSS Id: e4:ce:8f:66:b2:42
STA address: e4:ce:8f:5b:a1:f6
```

Sequence number

The next field includes the sequence number, 15:

```
0000 0000 1111 .... = Sequence number: 15
```

and the fragment number, 0:

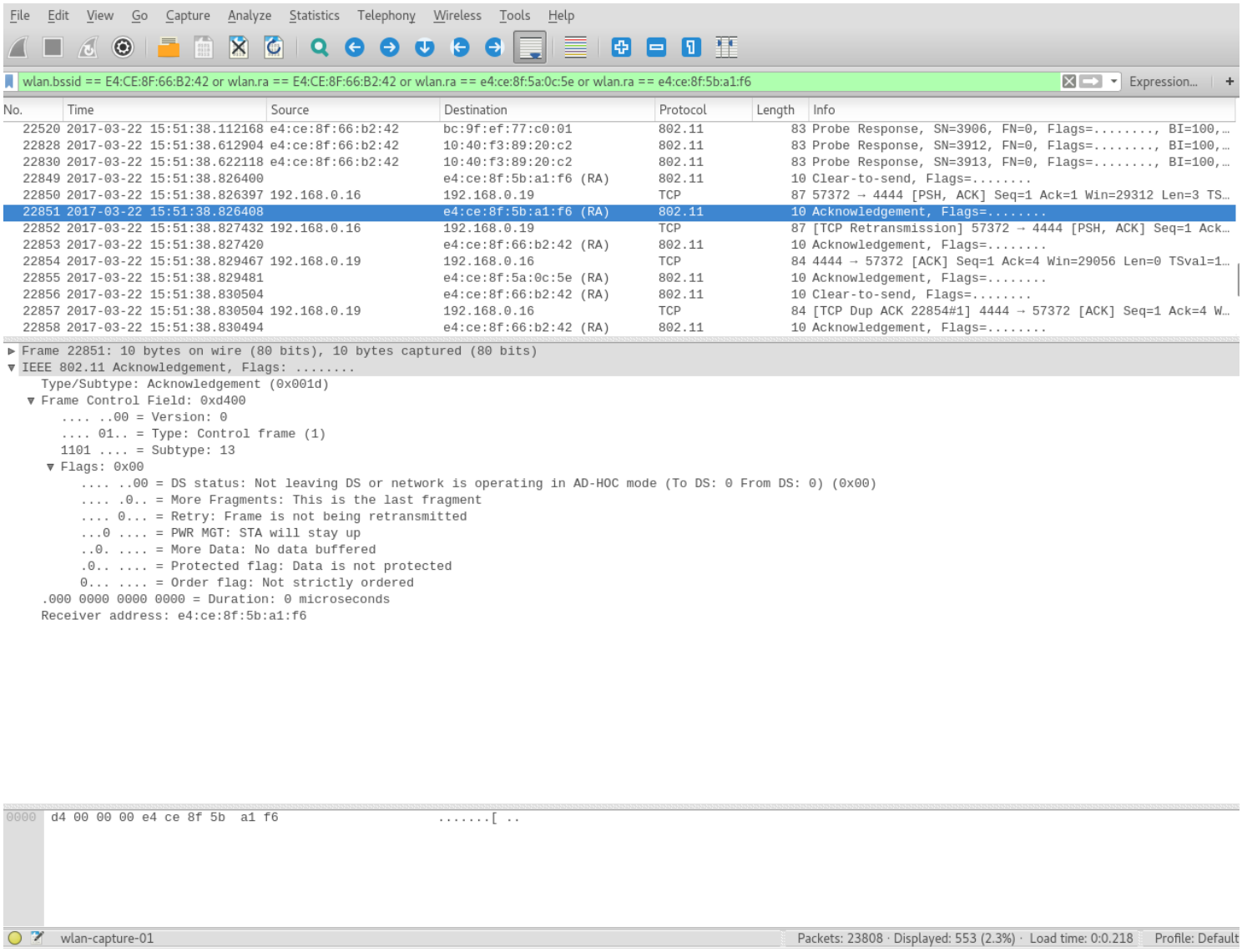
```
.... .... .... 0000 = Fragment number: 0
```

These binary digits - 0000 0000 1111 0000 - are visible in the hex dump as well:

```
08 01 30 00 e4 ce 8f 66 b2 42 e4 ce 8f 5b a1 f6
e4 ce 8f 5a 0c 5e f0 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

Acknowledgment

Next, look in the capture file for the 802.11 acknowledgment for that data frame. It should be the next frame in the capture:

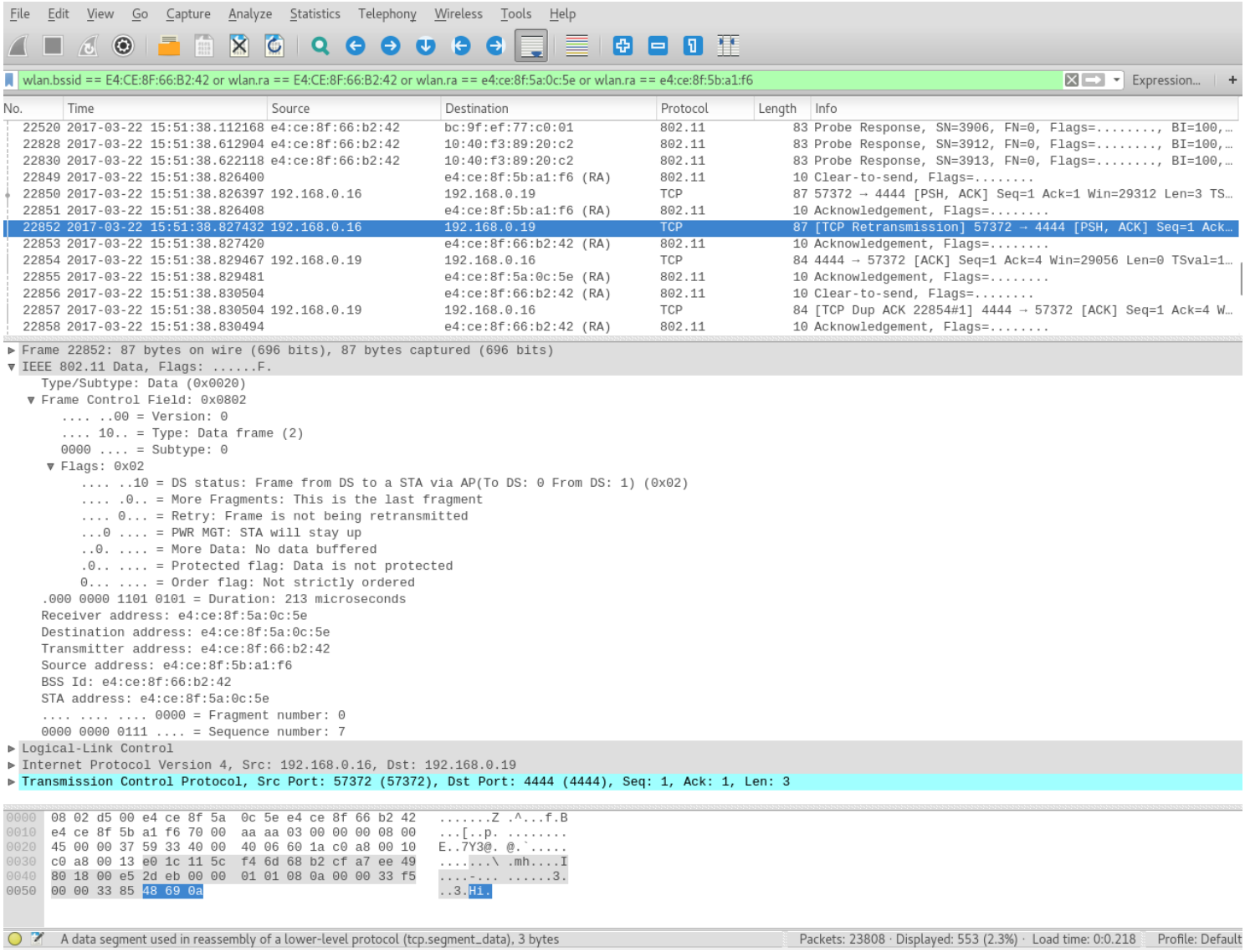


ACK frames have only three fields:

- **Frame Control** bits are set to 1101 to indicate an ACK frame.
- **Duration** field is set to 0 if acknowledging a complete data frame or the final fragment in a fragment burst.
- **Receiver address** field is set to the Address 2 field (transmitter address) of the frame that is being acknowledged.

Retransmission from AP to destination

Next, in our capture, we can observe that the same data payload is transmitted from the AP to the destination STA:



Here, we see some changes in the headers:

- The DS bits have changed, to indicate that this frame is sent from the AP to the distribution system.
- The duration is different.
- The addresses are different. We are now following the scheme in the second row in the table in the Addresses section, and the receiver and transmitter addresses are different now because the

frame is going from the AP to the destination STA, rather than from the source STA to the AP.

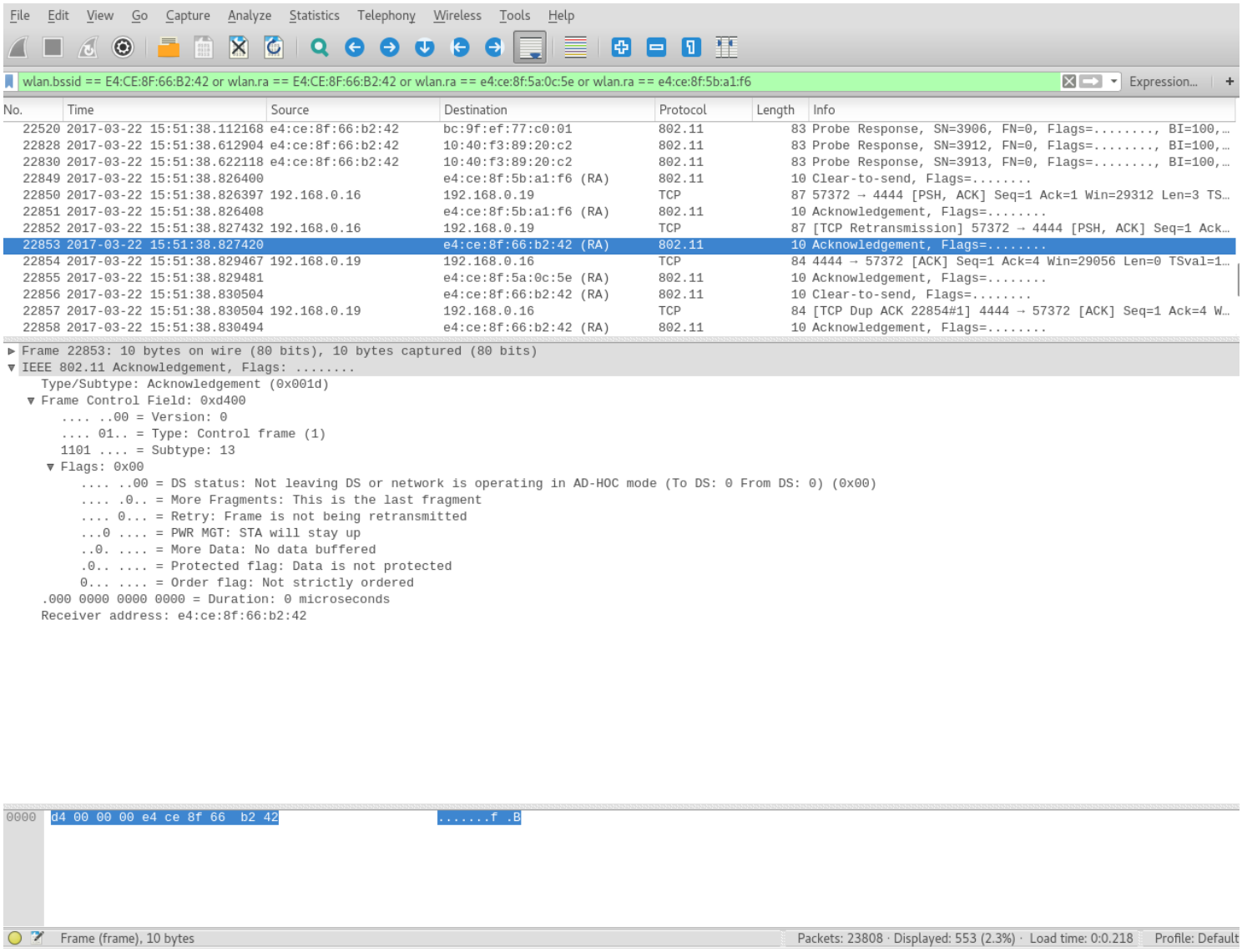
- The sequence number is different. This number indicates where a frame belongs in the sequence of frames sent from a transmitter address, to a receiver address. The STA1 to AP sequence numbers are incremented separately from the AP to STA2 sequence numbers.

But, the *data payload* of the frame, including all the higher-layer headers and the text of the message, are the same:

```
08 02 d5 00 e4 ce 8f 5a 0c 5e e4 ce 8f 66 b2 42
e4 ce 8f 5b a1 f6 70 00 aa aa 03 00 00 00 08 00
45 00 00 37 59 33 40 00 40 06 60 1a c0 a8 00 10
c0 a8 00 13 e0 1c 11 5c f4 6d 68 b2 cf a7 ee 49
80 18 00 e5 2d eb 00 00 01 01 08 0a 00 00 33 f5
00 00 33 85 48 69 0a
```

Second acknowledgment

Finally, we see that this frame, too, is acknowledged by its receiver:



The receiver address of the ACK is the MAC address of the AP. This receive ACK lets the AP know that the frame was received at the destination STA.

Notes

Exercise

Annotate the packets you captured following the tutorial above, as I have done in the Results section. Show **four** annotated packets:

- The first frame carrying the message you typed into `netcat`

- its 802.11 acknowledgment
- its retransmission by the AP
- the 802.11 acknowledgment of the retransmission.

Then, modify the experiment so that the network operates in ad-hoc mode, rather than in infrastructure mode, as follows:

First, reconfigure each of the two wireless station nodes so that they are in ad-hoc mode:

```
modprobe ath9k
ifconfig wlan0 down
iwconfig wlan0 mode ad-hoc
iwconfig wlan0 channel 11
iwconfig wlan0 essid 'witest-adhoc'
ifconfig wlan0 up
```

Verify connectivity with

```
iwconfig wlan0
```

(it may take a few moments for the connection to be established). Note that the BSSID will be a random MAC address, like this:

```
wlan0     IEEE 802.11abgn  ESSID:"witest-adhoc"
          Mode:Ad-Hoc  Frequency:2.462 GHz  Cell: FA:3D:4C:02:AD:87
          Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
```

```
          Encryption key:off
          Power Management:off
```

Make a note of this BSSID address. Then, configure an IP address on each with `ifconfig` as before, and verify connectivity with `ping`.

On the monitoring node, run

```
airodump-ng -c 11 --bssid FA:3D:4C:02:AD:87 --output-format pcap -w wla
```

using the BSSID you noted in the previous step.

Use `netcat` as before to send a message from one wireless station to another, then use Ctrl+C to stop everything. On the monitoring node, use `ls` to verify that there is a new capture file. The capture file should still have the prefix "wlan-capture", but the number is incremented, e.g. "wlan-capture-02.cap".

Transfer this capture file to your computer, and observe it in Wireshark. Filter as before, but using the new BSSID, e.g.

```
wlan.bssid == FA:3D:4C:02:AD:87 or wlan.ra == FA:3D:4C:02:AD:87 or
wlan.ra == e4:ce:8f:5a:0c:5e or wlan.ra == e4:ce:8f:5b:a1:f6
```

Show the frame(s) carrying the message you typed into `netcat`. Which header fields are different, compared to the previous experiment? What else is different about operation in ad-hoc mode vs. infrastructure mode? Explain.

Using other wireless testbeds

You may also run this experiment on the "outdoor" testbed at ORBIT, using any group of four nodes that are available, close to one another, and have Atheros 9XXX cards. You can verify the availability of "outdoor" nodes and their capabilities as follows:

- Visit http://geni.orbit-lab.org and log in
- Click on "Control Panel"
- Click on "Status Page"
- Choose the "outdoor" tab
- Scroll down to the "WiFi" panel, and check the box next to "Ath9k". (This will mark with an X any node that has a WiFi interface in the Atheros 9xxx family.)

In the display, nodes that are available are shown as blue or green squares; nodes that are not available are shown as red squares. You can find out the name of a node by clicking on it, and then looking at the "Info" panel on the left. Use this information to make substitutions in the instructions, replacing unavailable nodes (if any; shown as red squares) with nearby available nodes that have Atheros 9xxx cards.

This experiment will also work on the "sb4" testbed at ORBIT, which currently has four Atheros 9xxx-equipped nodes: node1-3, node1-4, node1-5, and node1-6.

If using "sb4", when you first log in to the "sb4" console you should run

```
wget -qO- "http://internal2dmz.orbit-lab.org:5054/instr/setAll?att=0"
```

```
wget -qO- "http://internal2dmz.orbit-lab.org:5054/instr/selDevice?switc
wget -qO- "http://internal2dmz.orbit-lab.org:5054/instr/selDevice?switc
wget -qO- "http://internal2dmz.orbit-lab.org:5054/instr/selDevice?switc
wget -qO- "http://internal2dmz.orbit-lab.org:5054/instr/selDevice?switc
```

to reset sb4's programmable attenuation matrix to zero attenuation between all pairs of nodes.

Similarly, you may also run this experiment on any group of four Ath9k-capable nodes on the "grid" testbed at ORBIT. The "grid" testbed is generally in high demand, however, and is difficult to get time on.

If using another testbed, you will have to make some minor changes to these instructions.

- You'll need to SSH in to the console of the testbed you have reserved, e.g. "outdoor.orbit-lab.org" or "sb4.orbit-lab.org", instead of "witestlab.poly.edu".
- The `wifi-experiment.ndz` disk image is also available on ORBIT, but you will substitute the correct node names in the `omf load` and `omf tell` commands. For example, on sb4:

```
omf-5.4 load -i wifi-experiment.ndz -t node1-3.sb4.orbit-
lab.org,node1-4.sb4.orbit-lab.org,node1-5.sb4.orbit-lab.org,node1-
6.sb4.orbit-lab.org

omf tell -a on -t node1-3.sb4.orbit-lab.org,node1-4.sb4.orbit-
lab.org,node1-5.sb4.orbit-lab.org,node1-6.sb4.orbit-lab.org
```

- You will also have assigned different IP addresses to the wireless interfaces, so you should make sure to use the correct IP addresses in the `netcat` commands.

- Your `scp` command will have to change to reflect the testbed you are running on and the node on which you are monitoring traffic.

Using testbeds besides for ORBIT or WITest

To use another wireless testbed besides for ORBIT or WITest, you may need to install some software or do some other configuration steps that are already prepared on the `wifi-experiment.ndz` disk image on ORBIT/WITest.

To create the `wifi-experiment.ndz` disk image, I started from a baseline Ubuntu 14.04 disk image. Then I installed some software from the Ubuntu package repositories:

```
apt-get update
apt-get -y install git hostapd iproute2 dnsmasq iptables haveged
aircrack-ng
```

I installed the create_ap tool, which makes it easy to set up a device as a WiFi access point:

```
git clone https://github.com/oblique/create_ap.git
cd create_ap
make install
```

I also un-blacklisted the `ath9k` driver, i.e.

```
rm /etc/modprobe.d/blacklist-ath9k.conf
```

so that the `ath9k` module is loaded at boot. Alternatively, you can manually load the module on each boot with

```
modprobe ath9k
```

Fraida Fund

Read more posts by this author.

Brooklyn, NY *http://witestlab.poly.edu/~ffund/*

Share this post

Did you reproduce this experiment? Have useful information to share with other intrepid researchers? Post it here! Comments are posted following moderation.

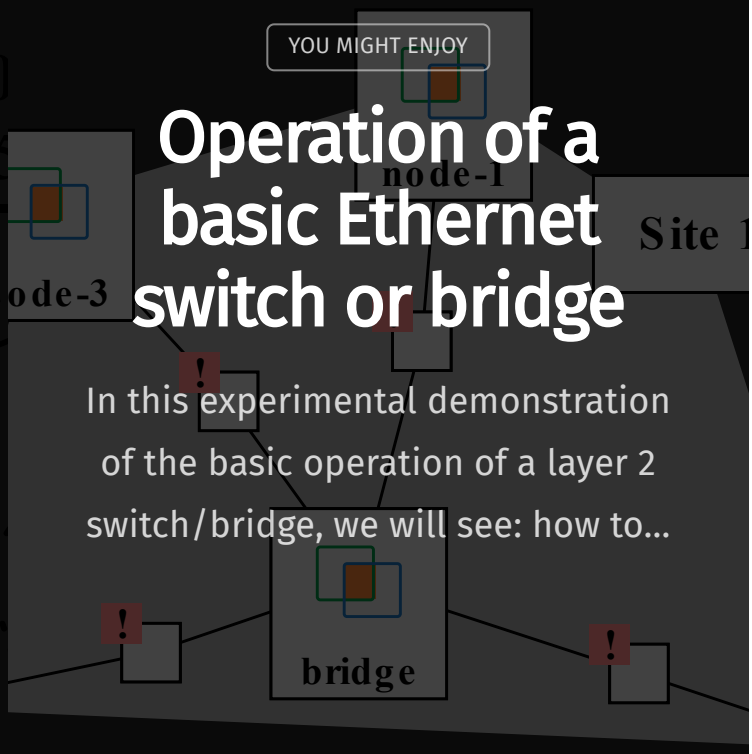
READ THIS NEXT

Channel planning in 802.11: understanding the effect of nearby networks

When neighboring 802.11 networks operate on the same channel, they compete with one another to use the shared...

YOU MIGHT ENJOY

Operation of a basic Ethernet switch or bridge



In this experimental demonstration of the basic operation of a layer 2 switch/bridge, we will see: how to...