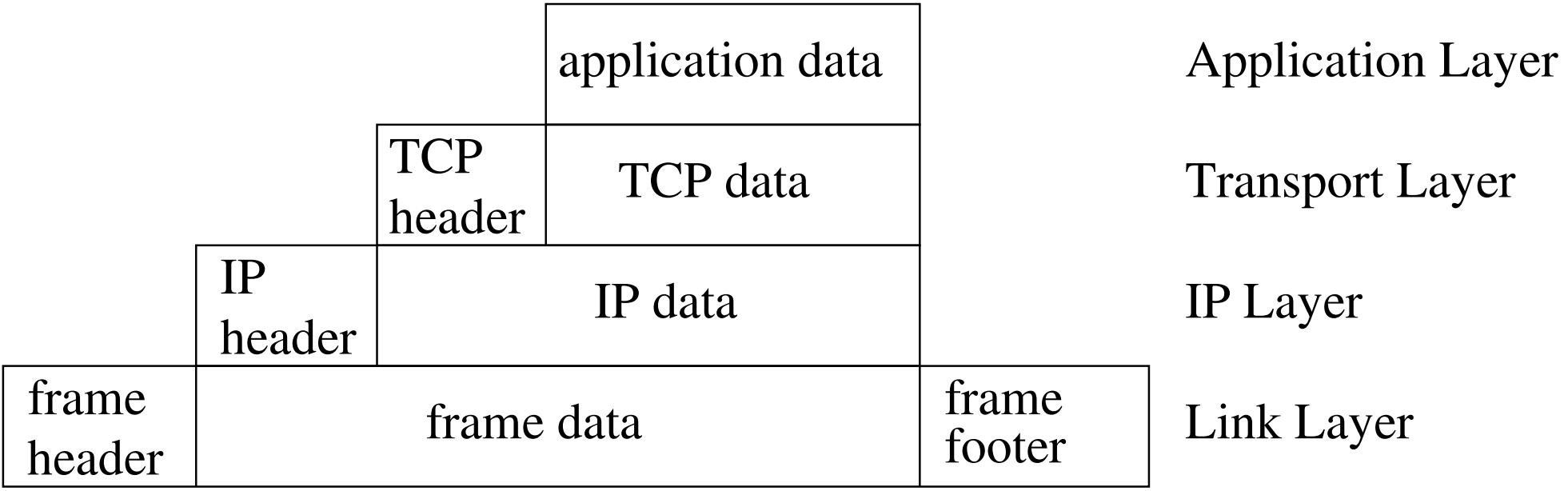
# ARP Spoofing


application data
Application Layer
TCP header
TCP data
Transport Layer
IP header
IP data
IP Layer
frame header
frame data
frame footer
Link Layer

# MAC Address

- every device that connects to a network has a **network interface**
  - these have identifiers called MAC address
  - Media Access Control
  - 6 octets
    - written in hex
    - separated by colons
    - e.g., 00:0c:29:91:d0:93

# MAC Address

- often used (unfortunately) to authenticate on a network
  - only a particular MAC address can use a particular ethernet wall jack
- MAC address are assigned at the factory but can be changed
- change is through software, so the network driver uses a new one

README GPL-3.0 license

GNU MAC Changer
===============

GNU MAC Changer is an utility that makes the maniputation of MAC addresses of network interfaces easier.

 - Web site: http://www.gnu.org/software/macchanger
 - Repository: http://github.com/alobbs/macchanger

All the best,
Alvaro Lopez Ortega <alvaro@alobbs.com>

```c
int
mc_net_info_set_mac (net_info_t *net, const mac_t *mac)
{
        int i;

        for (i=0; i<6; i++) {
                net->dev.ifr_hwaddr.sa_data[i] = mac->byte[i];
        }

        if (ioctl(net->sock, SIOCSIFHWADDR, &net->dev) < 0) {
                perror ("[ERROR] Could not change MAC: interface up or insufficient permissions");
                return -1;
        }

        return 0;
}
```

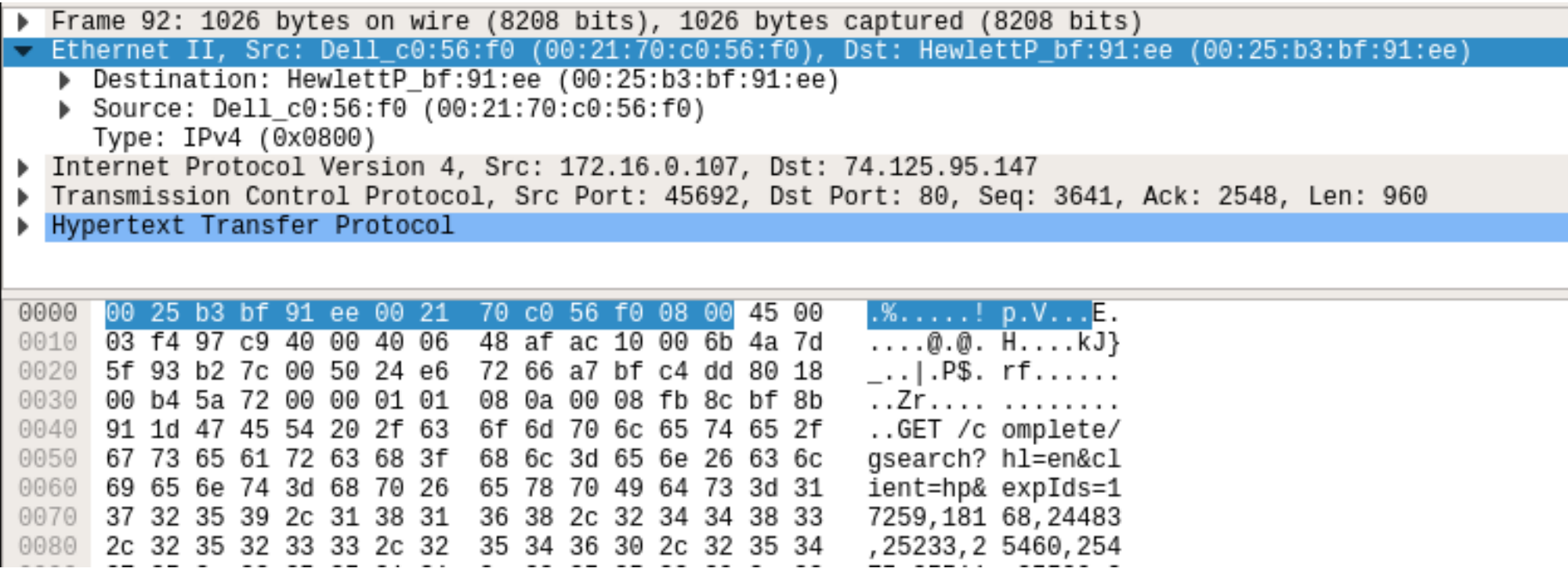
The MAC address is self reported in the packet.
Not reliable identification / authentication

The MAC address is self reported in the packet.
Not reliable identification / authentication
Yet it is widely used.

The MAC address is self reported in the packet.
Not reliable identification / authentication
Yet it is widely used.
Even here, today, with non-trivial cost.

# Link Layer

- individual units that are sent are **data frames**
- these frames are sent to MAC address
  - not IP addresses!
- IP addresses are resolved to MAC addresses

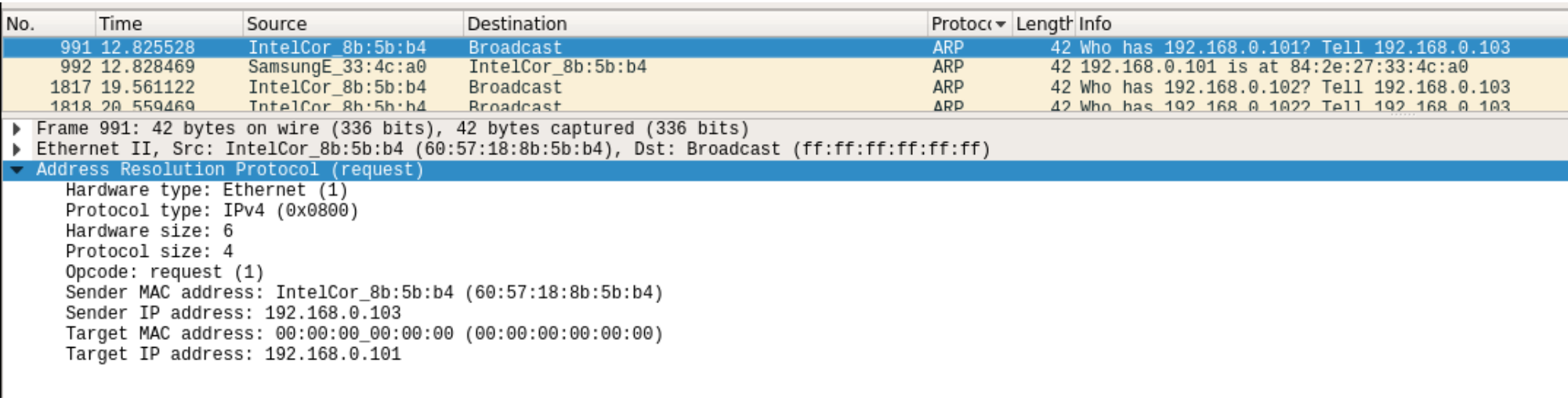

Frame 92: 1026 bytes on wire (8208 bits), 1026 bytes captured (8208 bits)
Ethernet II, Src: Dell_c0:56:f0 (00:21:70:c0:56:f0), Dst: HewlettP_bf:91:ee (00:25:b3:bf:91:ee)
Destination: HewlettP_bf:91:ee (00:25:b3:bf:91:ee)
Source: Dell_c0:56:f0 (00:21:70:c0:56:f0)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 172.16.0.107, Dst: 74.125.95.147
Transmission Control Protocol, Src Port: 45692, Dst Port: 80, Seq: 3641, Ack: 2548, Len: 960
Hypertext Transfer Protocol
0000 00 25 b3 bf 91 ee 00 21 70 c0 56 f0 08 00 45 00 .%.....! p.V...E.
0010 03 f4 97 c9 40 00 40 06 48 af ac 10 00 6b 4a 7d ....@.@. H....kJ}
0020 5f 93 b2 7c 00 50 24 e6 72 66 a7 bf c4 dd 80 18 _..|.P$. rf......
0030 00 b4 5a 72 00 00 01 01 08 0a 00 08 fb 8c bf 8b ..Zr.... ........
0040 91 1d 47 45 54 20 2f 63 6f 6d 70 6c 65 74 65 2f ..GET /c omplete/
0050 67 73 65 61 72 63 68 3f 68 6c 3d 65 6e 26 63 6c gsearch? hl=en&cl
0060 69 65 6e 74 3d 68 70 26 65 78 70 49 64 73 3d 31 ient=hp& expIds=1
0070 37 32 35 39 2c 31 38 31 36 38 2c 32 34 34 38 33 7259,181 68,24483
0080 2c 32 35 32 33 33 2c 32 35 34 36 30 2c 32 35 34 ,25233,2 5460,254

# ARP protocol

- ARP is address resolution protocol
- used to translate IPs to MAC addresses
- based on broadcast over network

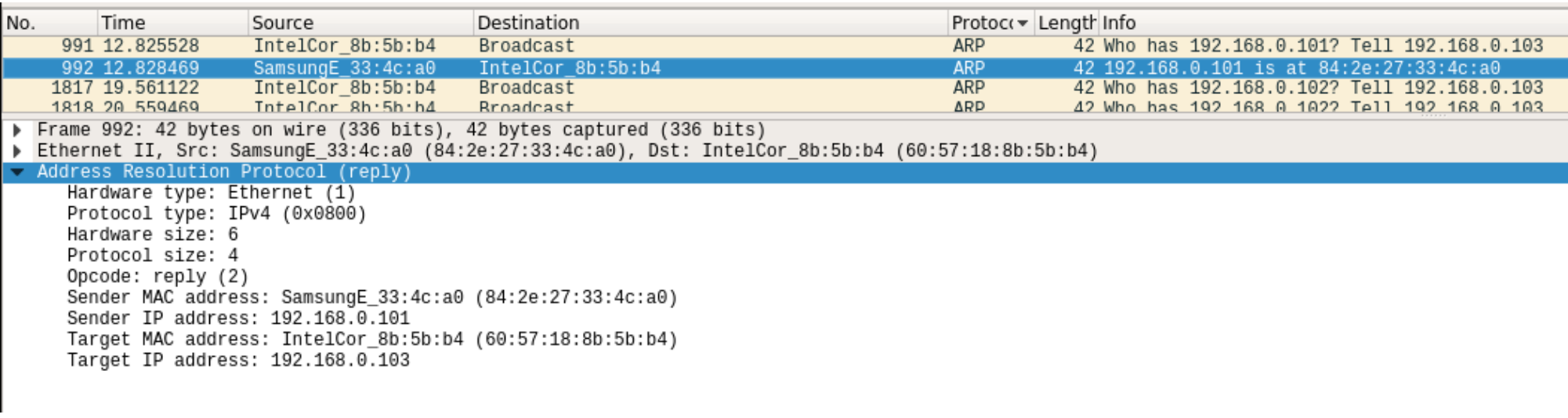
# ARP protocol

- Someone says: "I have IP 10.0.0.123 and MAC 00:0c:29:91:d0:93. Who has IP 10.0.0.12?"
- The machine with IP 10.0.0.12 replies: "Tell 00:0c:29:91:d0:93 that 10.0.0.12 is 00:0c:29:91:34:23"
- machines that hear this cache the result
  - i.e., set 10.0.0.12 $\rightarrow$ 00:0c:29:91:34:23
  - any ARP they see they update
  - idea was to save network bandwidth in the pre-switch era



| No. | Time | Source | Destination | Protoc | Length | Info |
|---|---|---|---|---|---|---|
| 991 | 12.825528 | IntelCor_8b:5b:b4 | Broadcast | ARP | 42 | Who has 192.168.0.101? Tell 192.168.0.103 |
| 992 | 12.828469 | SamsungE_33:4c:a0 | IntelCor_8b:5b:b4 | ARP | 42 | 192.168.0.101 is at 84:2e:27:33:4c:a0 |
| 1817 | 19.561122 | IntelCor_8b:5b:b4 | Broadcast | ARP | 42 | Who has 192.168.0.102? Tell 192.168.0.103 |
| 1818 | 20.559469 | IntelCor_8b:5b:b4 | Broadcast | ARP | 42 | Who has 192.168.0.102? Tell 192.168.0.103 |

```
▸ Frame 991: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
▸ Ethernet II, Src: IntelCor_8b:5b:b4 (60:57:18:8b:5b:b4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_8b:5b:b4 (60:57:18:8b:5b:b4)
    Sender IP address: 192.168.0.103
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.0.101
```


No. Time Source Destination Protoc Length Info
991 12.825528 IntelCor_8b:5b:b4 Broadcast ARP 42 Who has 192.168.0.101? Tell 192.168.0.103
992 12.828469 SamsungE_33:4c:a0 IntelCor_8b:5b:b4 ARP 42 192.168.0.101 is at 84:2e:27:33:4c:a0
1817 19.561122 IntelCor_8b:5b:b4 Broadcast ARP 42 Who has 192.168.0.102? Tell 192.168.0.103
Frame 992: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: SamsungE_33:4c:a0 (84:2e:27:33:4c:a0), Dst: IntelCor_8b:5b:b4 (60:57:18:8b:5b:b4)
Address Resolution Protocol (reply)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (2)
Sender MAC address: SamsungE_33:4c:a0 (84:2e:27:33:4c:a0)
Sender IP address: 192.168.0.101
Target MAC address: IntelCor_8b:5b:b4 (60:57:18:8b:5b:b4)
Target IP address: 192.168.0.103

# ARP protocol

- ARP has no authentication
- anyone can create ARP messages
- there is no guarantee that the MAC address for the IP is correct
- ARP messages can be created spuriously
- this enables MitM
  - e.g., by ARP poisoning the gateway (10.0.0.1)

# MitM on ARP

- Eve sends ARP reply to Alice to map Bob's IP to Eve's MAC
- Eve sends ARP reply to Bob to map Alice's IP to Eve's MAC
- Alice now sends all Bob-bound messages to Eve
- Bob now sends all Alice-bound messages to Eve
  - Eve is now **on-path**
  - Eve can forward traffic, monitor traffic, modify traffic

# DoS on ARP

- DoS (denial of service)
- Eve sends bad ARP replies to machines, mapping IP addresses to non-existent MACs
- no traffic is getting through

ARP Spoofing requires local access (e.g., LAN)

ARP Spoofing requires local access (e.g., LAN)
This means that it works for shared Wi-Fi, public Wi-Fi

ARP Spoofing requires local access (e.g., LAN)
This means that it works for shared Wi-Fi, public Wi-Fi
Once on path, easy to mount other attacks
(e.g., TLS stripping)

# How to deal with ARP Attacks

- only let in trusted users to a network
- detect multiple occurances of the same MAC address on a LAN
- statically define the ARP table