Digital communication: internet

J.-M Friedt

FEMTO-ST/time & frequency

jmfriedt@femto-st.fr

slides at `jmfriedt.free.fr`

March 14, 2021

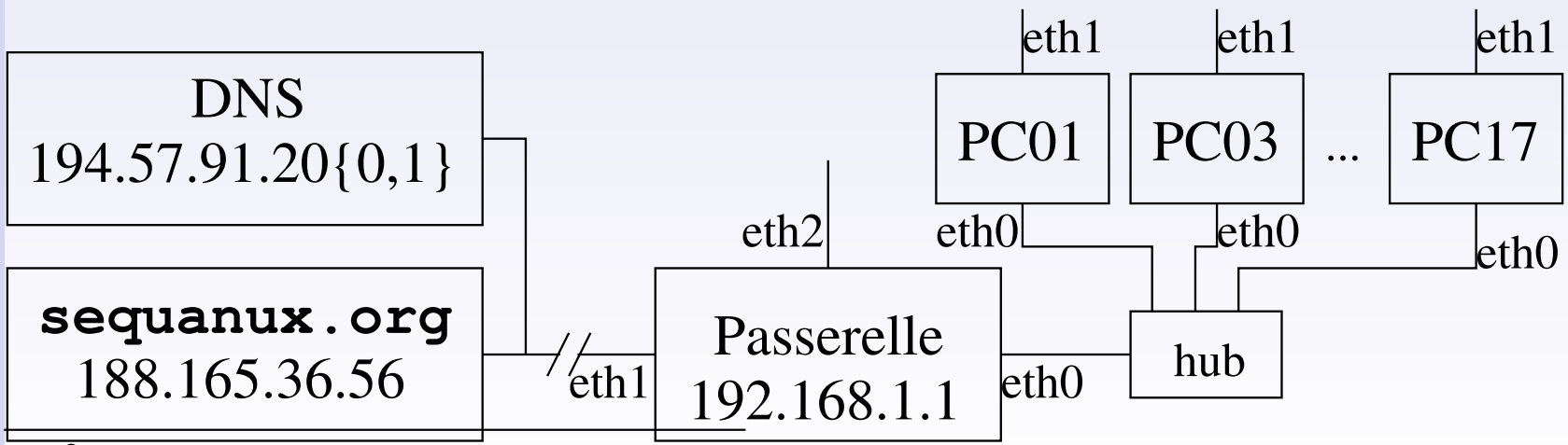
Objective of this presentation

- Understand some of the protocols used on the Internet [1]
- Principle of packet routing and network configuration
- Introduction to some of the most useful protocols
- Associated tools to trigger transactions and fetch answers
- Use for scripting transaction requests and traffic analysis, analyze security issues

[1] K. Hafner & M. Lyon, *Where Wizards Stay Up Late*, Simon & Schuster (1998)

Internet compatible digital communication

- IP (*Internet Protocol*): decentralized packet routing [2]
- interfaces define how packets are transmitted to the next communication link
- routing rules define the destination of each data packet
- a maximum time of live of each packet limits the number of routing steps
- MAC address: physical address of each interface, unique
- IP address: virtual adresse associated with a geographic location
- MAC-IP relation: ARP
- IP-domain name relation: DNS (`/etc/resolv.conf`)



[2] W.R. Stevens, *TCP/IP Illustrated Vol. 1*, Addison & Wesley (1994)

OSI layers

- physical layer: Ethernet, 802.11 Wi-Fi (collision handling, data packets)
- presentation layer: IP
- session layer: ICMP, UDP, TCP, ARP
- notions de client-serveur et sockets $\Rightarrow$ `fork()`

OSI model of network hierarchical layers [3]

	Layer/function	example
7	Application	(DNS, FTP, NTP ...)
6	Data representation	(ASCII, EBDIC, ...)
5	Session	(socket)
4	Transport	(TCP, UDP)
3	Network Packets	(IP)
2	Medium Access Control	(AppleTalk, 802.3 "ethernet")
1	Physical	(RS232, Ethernet 10BaseT, 802.11 "Wi-Fi")

Any computer supporting IP defines local address: 127.0.0.1 (localhost)

[3] J.-M Friedt, *Decoding Meteor M2: QPSK, Viterbi, Reed Solomon and JPEG* FOSDEM 2019, video at `https://archive.fosdem.org/2019/schedule/event/sdr_meteorm2n/` and `http://jmfriedt.free.fr/glmf_meteor_eng.pdf`

OSI layers

- physical layer: Ethernet, 802.11 Wi-Fi (collision handling, data packets)
- presentation layer: IP
- session layer: ICMP, UDP, TCP, ARP
- notions de client-serveur et sockets $\Rightarrow$ `fork()`

OSI model of network hierarchical layers [3]

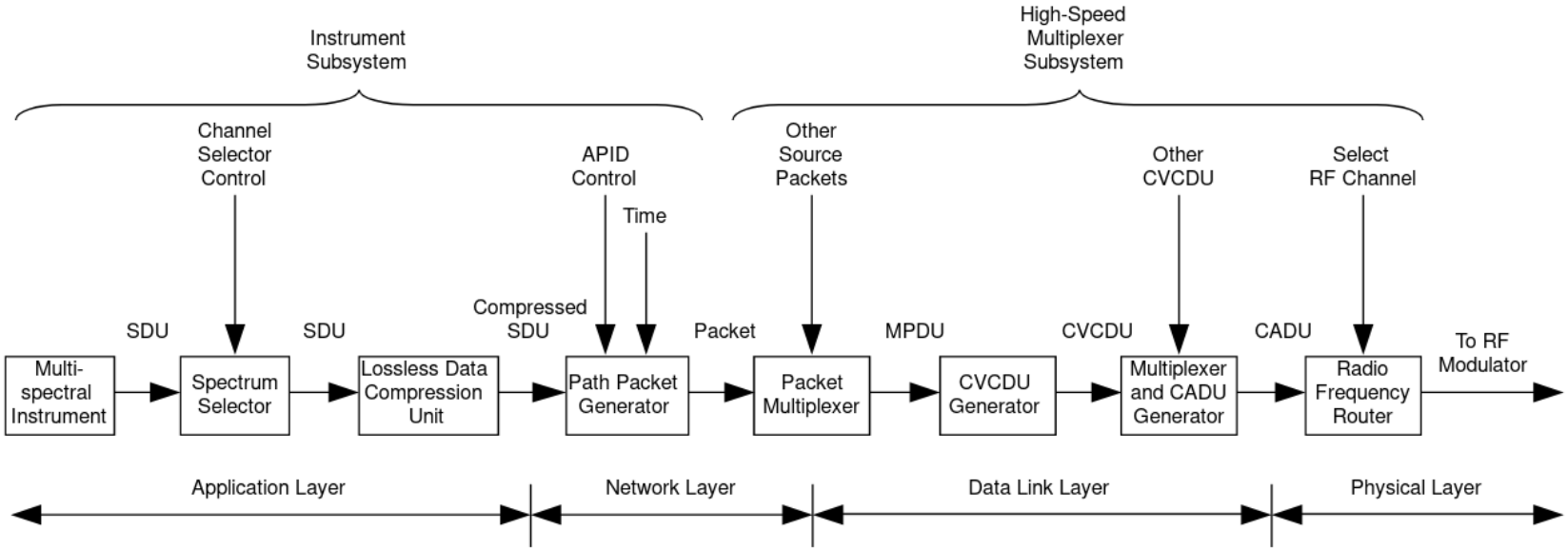


Figure 5-1: The Space Segment

`https://public.ccsds.org/Pubs/120x0g2s.pdf`, p.26

Any computer supporting IP defines local address: 127.0.0.1 (localhost)

[3] J.-M Friedt, *Decoding Meteor M2: QPSK, Viterbi, Reed Solomon and JPEG*

Network configuration on GNU/Linux

1. *interfaces*: `ifconfig eth0 192.168.1.1`

```
jmfriedt@dhcp-221:~/$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:48:54:55:09:6D
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::248:54ff:fe55:96d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1203876 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1616275 errors:0 dropped:0 overruns:1 carrier:0
          collisions:397422 txqueuelen:1000
          RX bytes:666243681 (635.3 MiB)  TX bytes:1888543246 (1.7 GiB)
          Interrupt:10 Base address:0xe800

eth1      Link encap:Ethernet  HWaddr 00:48:54:39:44:7A
          inet addr:172.16.120.21  Bcast:172.16.120.255  Mask:255.255.255.0
          inet6 addr: fe80::248:54ff:fe39:447a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5818227 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3024232 errors:0 dropped:0 overruns:1 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3049792520 (2.8 GiB)  TX bytes:1779165735 (1.6 GiB)
          Interrupt:11 Base address:0xec00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
...
```

2. *rout*ing table, automaticall filled for each sub-domain + gateway (*default gateway*),

```
jmfriedt@dhcp-221:~/eagle$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
172.16.120.0    *               255.255.255.0   U     0      0        0 eth1
default         172.16.120.254  0.0.0.0         UG    0      0        0 eth1
```

Network configuration on GNU/Linux

The "old way": `ifconfig`

- `ifconfig -a`: all interfaces
- `ifconfig eth0 IP`: configure interface eth0
- `ifconfig eth0 up`: activate interface eth0
- `ifconfig eth0 down`: deactivate interface eth0
- `route add default gw IP2`: default routing gateway
- `route -n`: dump routing table
- `ifconfig eth0`: display statistics

The "new way": `ip`

- `ip a`
- `ip a add IP dev eth0`
- `ip link set dev eth0 up`
- `ip link set dev eth0 down`
- `ip route add default via IP2`
- `ip r`
- `ip -s -h -a link`

Debian GNU/Linux: `net-tools` package to restore `ifconfig`

WiFi configuration on GNU/Linux

```
root@satellite:/home/jmfriedt# iwlist scan | grep -B5 FreeW
          Cell 07 - Address: EE:40:00:73:33:02
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=55/70  Signal level=-55 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
--
          Cell 08 - Address: 2A:C1:B0:8A:83:C1
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=26/70  Signal level=-84 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"
--
          Cell 11 - Address: EE:40:00:73:33:03
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=55/70  Signal level=-55 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"
--
          Cell 22 - Address: 22:2A:C4:65:D5:D2
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=17/70  Signal level=-93 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
--
          Cell 25 - Address: 92:84:62:B4:1C:56
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=16/70  Signal level=-94 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
--
          Cell 28 - Address: 68:A3:78:00:F5:C5
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=19/70  Signal level=-91 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
--
          Cell 29 - Address: 68:A3:78:00:F5:C6
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=16/70  Signal level=-94 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"
--
          Cell 33 - Address: 14:0C:76:B4:06:99
                    Channel:7
                    Frequency:2.442 GHz (Channel 7)
                    Quality=25/70  Signal level=-85 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"
--
          Cell 35 - Address: 14:0C:76:B4:06:98
                    Channel:7
                    Frequency:2.442 GHz (Channel 7)
                    Quality=25/70  Signal level=-85 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
```

WiFi configuration on GNU/Linux

```
root@satellite:/home/jmfriedt# iwlist scan | grep -B5 FreeW--
          Cell 07 - Address: EE:40:00:73:33:02
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=55/70  Signal level=-55 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"                        --
--
          Cell 08 - Address: 2A:C1:B0:8A:83:C1
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=26/70  Signal level=-84 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"                 --
--
          Cell 11 - Address: EE:40:00:73:33:03
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=55/70  Signal level=-55 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"                 --
--
          Cell 22 - Address: 22:2A:C4:65:D5:D2
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=17/70  Signal level=-93 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
          Cell 25 - Address: 92:84:62:B4:1C:56
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=16/70  Signal level=-94 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
          Cell 28 - Address: 68:A3:78:00:F5:C5
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=19/70  Signal level=-91 dBm
                    Encryption key:off
                    ESSID:"FreeWifi"
          Cell 29 - Address: 68:A3:78:00:F5:C6
                    Channel:4
                    Frequency:2.427 GHz (Channel 4)
                    Quality=16/70  Signal level=-94 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"
          Cell 33 - Address: 14:0C:76:B4:06:99
                    Channel:7
                    Frequency:2.442 GHz (Channel 7)
                    Quality=25/70  Signal level=-85 dBm
                    Encryption key:on
                    ESSID:"FreeWifi_secure"
```

```
ifdown wlan0            # OR ifconfig wlan0 down
iwconfig wlan0 essid "FreeWifi" ap EE:40:00:73:33:02
ifup wlan0              # OR dhclient wlan0
```

Debian GNU/Linux: `ifup` and `ifdown` provided by `ifupdown` and configured in `/etc/network/interfaces` (`man interfaces`)

Using telnet to reach high level protocols – SMTP

SMTP: port 25
(cf /etc/services)

```
$ telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 rugged ESMTP Exim 4.84
HELO moi.ici
250 rugged Hello moi.ici [::1]
MAIL FROM: moi@ici.maison
250 OK
RCPT TO: jmfriedt@localhost
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
simple email
.
250 OK id=1YfXdU-0004Cl-PT
QUIT
221 rugged closing connection
Connection closed by foreign host.
```

```
You have mail in /var/mail/jmfriedt
jmfriedt@rugged:~$ mail
Mail version 8.1.2 01/15/2001.  Type ? for
"/var/mail/jmfriedt": 1 message 1 new
>N  1 moi@ici.maison     Fri May 08 08:59
Message 1:
From moi@ici.maison Fri May 08 08:59:12 201
Envelope-to: jmfriedt@localhost
Delivery-date: Fri, 08 May 2015 08:59:12 +0
From: moi@ici.maison
Date: Fri, 08 May 2015 08:59:12 +0200

simple email
```

Using `telnet` to reach high level protocols – HTTP

- Requesting a web page through `telnet` – HTML web page structure
- HTTP [4] [5]: port 80 (Apache web server) [6]

```
jmfriedt@rugged:~$ telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
GET http://localhost/
Hello World
This is a test.
Connection closed by foreign host.
```

or accessing through a proxy

```
telnet proxy-www.univ-fcomte.fr 3128
GET http://jmfriedt.free.fr/ HTTP/1.1
```

- Most useful protocols: **read RFC** at `www.ietf.org/rfc`
  HTTP = RFC 2068 (Jan. 1997) and 2616 (Jun. 1999), SMTP = RFC 772 [7] (Sep. 1980) and 821 (Aug. 1982)

[4] J. Gillies & R. Cailliau, *How the Web Was Born: The Story of the World Wide Web*, Oxford Univ. Press (2000)

[5] T. Berners-Lee, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*, HarperBusiness (2000)

[6] W.R. Stevens, *TCP/IP Illustrated Vol. 3*, Addison & Wesley (1996), chapitre 13 "HTTP: Hypertext Transfer Protocol".

[7] `tools.ietf.org/html/rfc772`

Using `telnet` to reach high level protocols – FTP

- Requesting a file through `telnet` – FTP transactions
- FTP [8] (two connection – commands on port 21 and file transfer)

commande ...

```
$ telnet localhost 21
USER anonymous
EPSV
229 Extended Passive mode
            OK (|||36982|)
LIST
RETR fichier
```

... et data

```
$ telnet localhost 36982
-rw-r--r-- 0    May 8 09:50 fichier
-rw-r--r-- 1002 May 8 09:47 fichier_de_tes
Connection closed by foreign host.
$ telnet localhost 36982
ceci est un fichier de test
Connection closed by foreign host.
```

- Most useful protocols: **read RFC** at `www.ietf.org/rfc`
  FTP = RFC114 and followups, RFC542 (Aug. 1973), RFC959 (Oct. 1985)

[8] W.R. Stevens, *TCP/IP Illustrated Vol. 1*, Addison & Wesley (1994), chapitre 27 "FTP: File Transfer Protocol".

Routing protocols

- IP: computer identification. Coupled with ARP, each name+domain is associated with a unique address.
- ICMP (ping), network management.
- UDP: datagram (non-connected) mode. Packets are broadcast on the network with no constraint on latency or being received.
- TCP: connected mode. Packets are received and their order is controlled. Resource intensive (TCP/IP **stack**) [9]
- services, protocol used and port to which the served is connected is standardized: `/etc/services`
- too litte computational power makes an embedded system **vulnerable to DoS attacks**.

[9] E. Gergori, *Small Footprint ColdFire TCP/IP Stack*, Freescale AN5307 : 15 kB RAM et 35 kB flash

Client/server TCP/IP

```
#include <sys/socket.h>
#include <resolv.h>
#include <strings.h>
#include <arpa/inet.h>

#define MY_PORT         9999
#define MAXBUF          1024

int main()
{int sockfd;
 struct sockaddr_in self;
 char buffer[MAXBUF];

 sockfd = socket(AF_INET, SOCK_STREAM, 0);  // ICI LE TYPE DE SOCKET

 bzero(&self, sizeof(self));
 self.sin_family = AF_INET;
 self.sin_port = htons(MY_PORT);
 self.sin_addr.s_addr = INADDR_ANY;

 bind(sockfd, (struct sockaddr*)&self, sizeof(self));
 listen(sockfd, 20);

 while (1)
 {struct sockaddr_in client_addr;
  int clientfd,addrlen=sizeof(client_addr);

  clientfd = accept(sockfd, (struct sockaddr*)&client_addr, &addrlen);
  printf("%s:%d connected\n", inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
  send(clientfd, buffer, recv(clientfd, buffer, MAXBUF, 0), 0);

  close(clientfd);
 }
 close(sockfd);return(0);  // Clean up (should never get here)
}
```

1. socket (protocol)
2. bind (port)
3. listen (blocking wait)
4. accept
5. send/recv
6. close

Beware of *endianness*:
Internet, Java: big-endian
x86, ARM are little-endian

Easiest way of testing a TCP server: `telnet IP port`

Client/serveur UDP/IP

UDP communication [10] : A server is waiting for connections

```
#include <sys/socket.h>
#include <resolv.h>
#include <arpa/inet.h>

#define BUFSIZE         1024

void alltoupper(char* s)
{while ( *s != 0 ) *s++ = toupper(*s);}

int main()
{ char buffer[BUFSIZE];
  struct sockaddr_in addr;
  int sd, addr_size, bytes_read;

  sd = socket(PF_INET, SOCK_DGRAM, 0);
  addr.sin_family = AF_INET;
  addr.sin_port = htons(9999);
  addr.sin_addr.s_addr = INADDR_ANY;
  bind(sd, (struct sockaddr*)&addr, sizeof(addr));
  do {bzero(buffer, BUFSIZE);addr_size = BUFSIZE;
      bytes_read=recvfrom(sd,buffer,BUFSIZE,0, \
         (struct sockaddr*)&addr,&addr_size);
      printf("Connect: %s:%d %s\n",inet_ntoa(addr.sin_addr),\
         ntohs(addr.sin_port), buffer);
      alltoupper(buffer);
      sendto(sd,buffer,bytes_read,0,(struct sockaddr*)&addr, \
         addr_size);
  } while ( bytes_read > 0 );
  close(sd);return 0;
}
```

A client connects to a server to trigger transactions

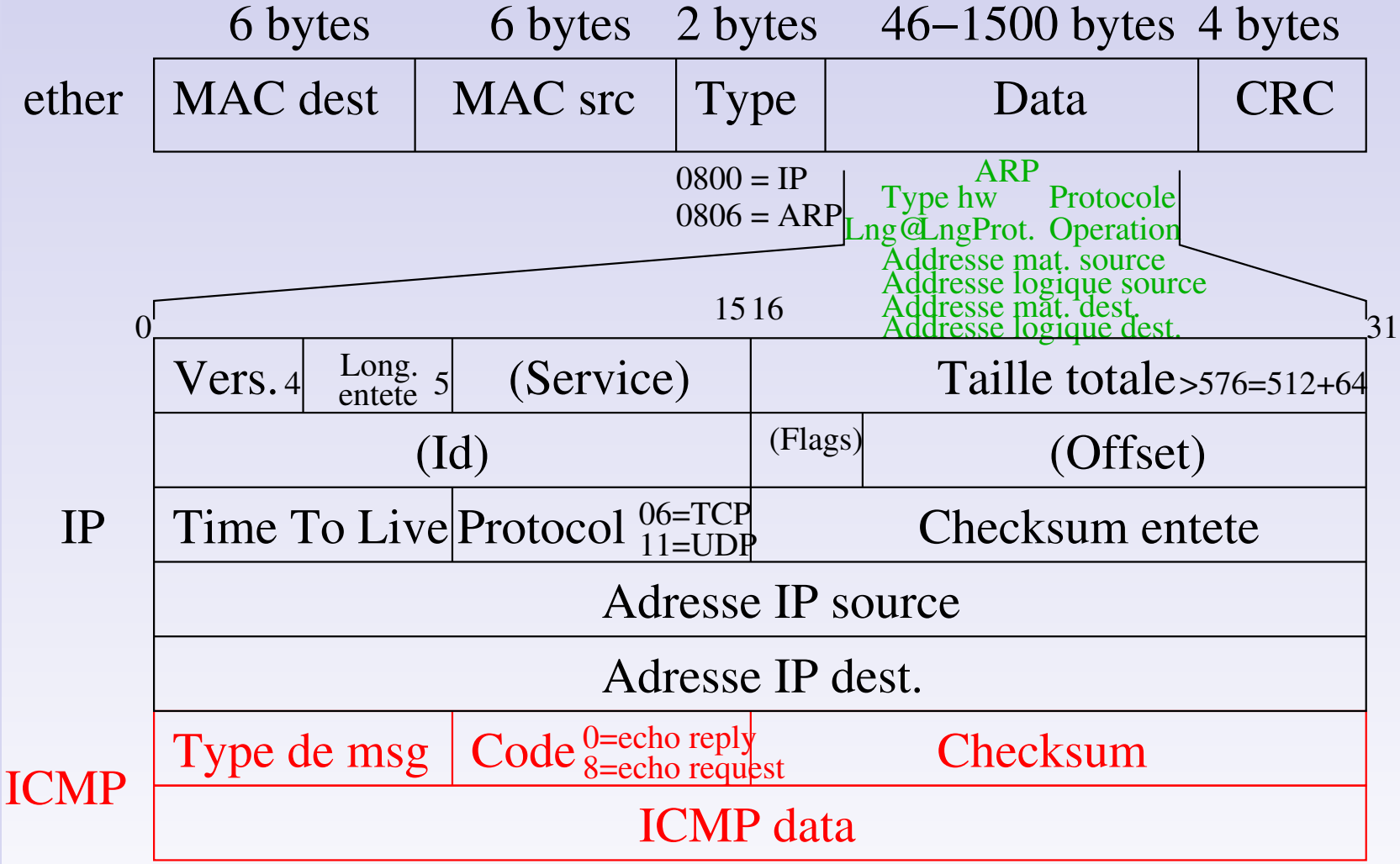
```
#include <sys/socket.h>
#include <resolv.h>
#include <arpa/inet.h>

#define BUFSIZE         1024

int main(int argc, char **argv)
{     char buffer[BUFSIZE];
      struct sockaddr_in addr;
      int sd, addr_size;

      if ( argc != 2 )
         {printf("usage: %s <msg>\n", argv[0]);exit(0);}
      sd = socket(PF_INET, SOCK_DGRAM, 0);
      addr.sin_family = AF_INET;
      addr.sin_port = htons(9999);
      inet_aton("127.0.0.1", &addr.sin_addr);
      sendto(sd, argv[1], strlen(argv[1])+1, 0, \
         (struct sockaddr*)&addr, sizeof(addr));
      bzero(buffer, BUFSIZE);
      addr_size = sizeof(addr);
      recvfrom(sd,buffer,BUFSIZE,0,(struct sockaddr*)&addr,\
        &addr_size);
      printf("Reply: %s:%d %s\n",inet_ntoa(addr.sin_addr), \
        ntohs(addr.sin_port), buffer);
      close(sd);
      return 0;
}
```

or `echo "toto" | nc -u IP 9999`

Cross-compilation to embedded targets – use of embedded OS

[10] http://www.cs.utah.edu/~swalton/listings/sockets/programs/

Raw IP et ICMP



Question: what is the protocol identified of ICMP in the IP header?

Raw IP & ICMP

```
$ arp -a
? (192.168.0.1) at 00:13:d3:8d:d3:97 [ether] on eth0
$ arp -d 192.168.0.1
$ ping 192.168.0.11

# tcpdump  -i eth0 -XX                          # XX = show ethernet header
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:06:33.462038 ARP, Request who-has 192.168.0.1 tell 192.168.0.11, length
        0x0000:  ffff ffff ffff cc7e e75f cf6e 0806 0001  .......~._.n....
        0x0010:  0800 0604 0001 cc7e e75f cf6e c0a8 000b  .......~._.n....
        0x0020:  0000 0000 0000 c0a8 0001                 ..........
16:06:33.462590 ARP, Reply 192.168.0.1 is-at 00:13:d3:8d:d3:97 (oui Unknown
        0x0000:  cc7e e75f cf6e 0013 d38d d397 0806 0001  .~._.n..........
        0x0010:  0800 0604 0002 0013 d38d d397 c0a8 0001  ................
        0x0020:  cc7e e75f cf6e c0a8 000b 0000 0000 0000  .~._.n..........
        0x0030:  0000 0000 0000 0000 0000 0000            ............
16:06:33.462607 IP 192.168.0.11 > 192.168.1.1: ICMP echo request, id 1906,
        0x0000:  0013 d38d d397 cc7e e75f cf6e 0800 4500  .......~._.n..E.
        0x0010:  0054 1191 4000 4001 a6bb c0a8 000b c0a8  .T..@.@.........
        0x0020:  0101 0800 ff64 0772 0001 e9c2 4c55 c90c  .....d.r....LU..
        0x0030:  0700 0809 0a0b 0c0d 0e0f 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
```

Raw IP & ICMP

Good understanding of network protocols useful for applications related to

- security: firewall or dedicated router. Packet filtering (`iptables`) with short packet management latency
- remote instrument control. Even without considering data security, prevent DoS attacks.
- handling all layers is often not needed on embedded systems: understanding the whole communication hierarchy allows for selecting only the useful parts (e.g raw-IP).

In previous examples, `socket` argument defined the transport protocol (`SOCK_DGRAM` (UDP), `SOCK_STREAM` (TCP)).

`SOCK_RAW` $\Rightarrow$ 20 byte header with source and destination IP @, followed by payload to be communicated (routed)

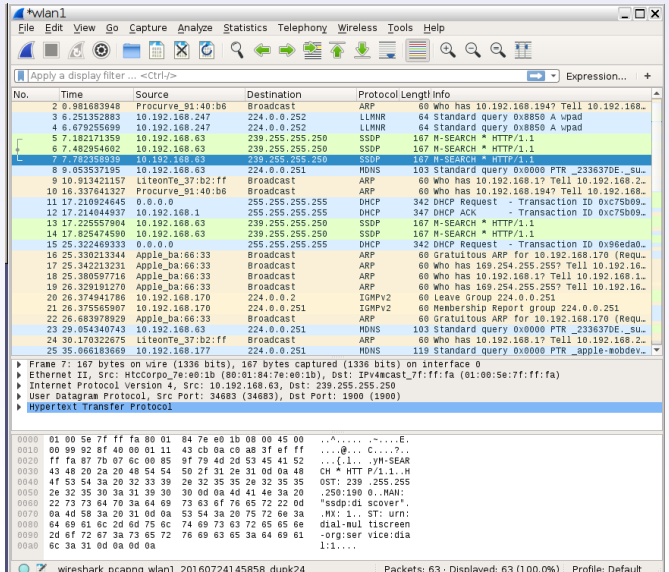
```
45 00 00 34 19 a9 00 00 3c ff 66 20 7f 00 00 01    <- source = 127.0.0.1
7f 00 00 01 30 30 30 30 30 30 30 30 30 30 30 30    <- dest = 127.0.0.1
```

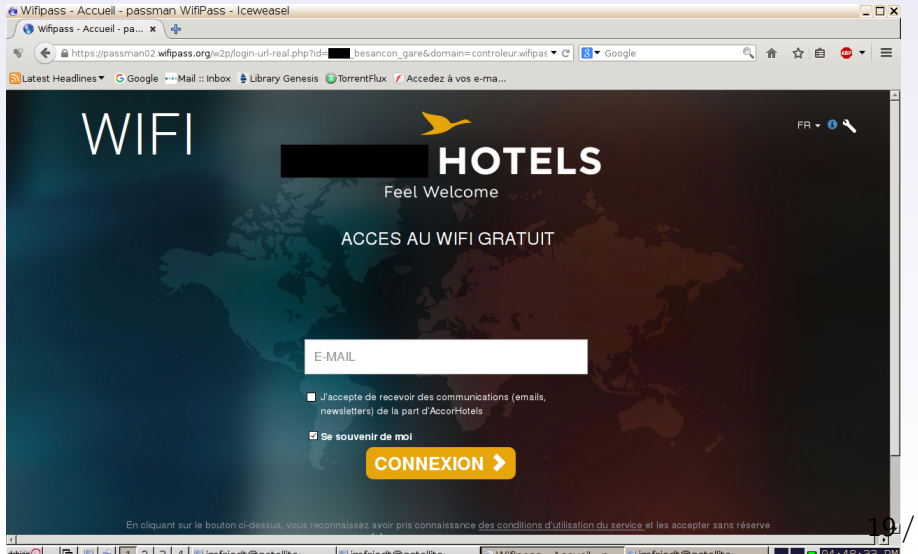
MAC spoofing

- most modern interfaces (Ethernet, WiFi) configure the hardware MAC address by software

```
# /sbin/ifconfig eth0
eth0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      inet 192.168.1.55  netmask 255.255.255.0  broadcast 192.168.1.255
      ether cc:7e:e7:5f:cf:6e  txqueuelen 1000  (Ethernet)
[...]
# /sbin/ifconfig eth0 hw ether c4:d9:87:12:ea:3c
# /sbin/ifconfig eth0
eth0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      inet 192.168.1.55  netmask 255.255.255.0  broadcast 192.168.1.255
      ether c4:d9:87:12:ea:3c  txqueuelen 1000  (Ethernet)
[...          ^^^^^^^^^^^^^^^^^ ]
```

- captive portals identify registered customers by their MAC address ...



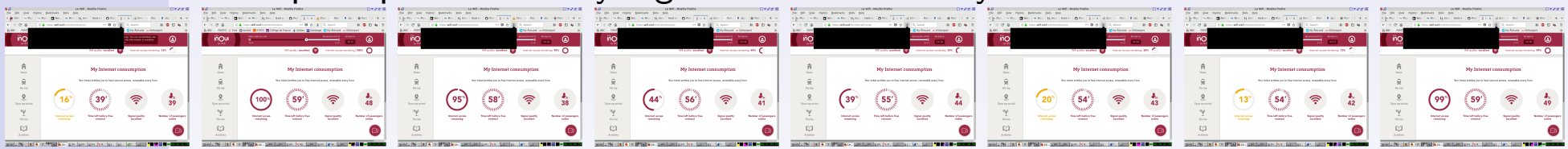


MAC spoofing

- most modern interfaces (Ethernet, WiFi) configure the hardware MAC address by software

```
# /sbin/ifconfig eth0
eth0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      inet 192.168.1.55  netmask 255.255.255.0  broadcast 192.168.1.255
      ether cc:7e:e7:5f:cf:6e  txqueuelen 1000  (Ethernet)
[...]
# /sbin/ifconfig eth0 hw ether c4:d9:87:12:ea:3c
# /sbin/ifconfig eth0
eth0: flags=4098<BROADCAST,MULTICAST>  mtu 1500
      inet 192.168.1.55  netmask 255.255.255.0  broadcast 192.168.1.255
      ether c4:d9:87:12:ea:3c  txqueuelen 1000  (Ethernet)
[...          ^^^^^^^^^^^^^^^^^^ ]
```

- captive portals identify registered customers by their MAC address ...

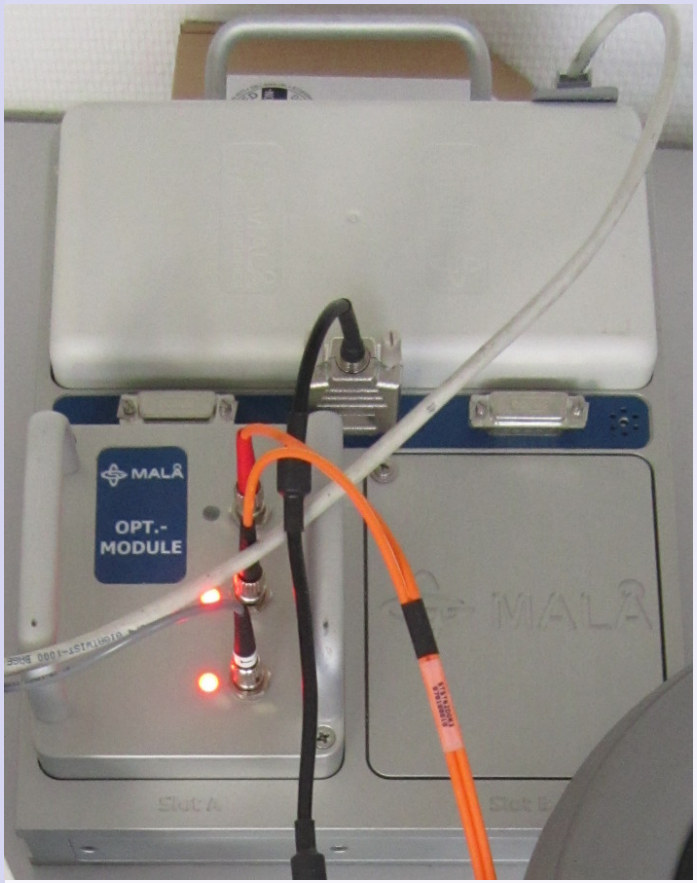


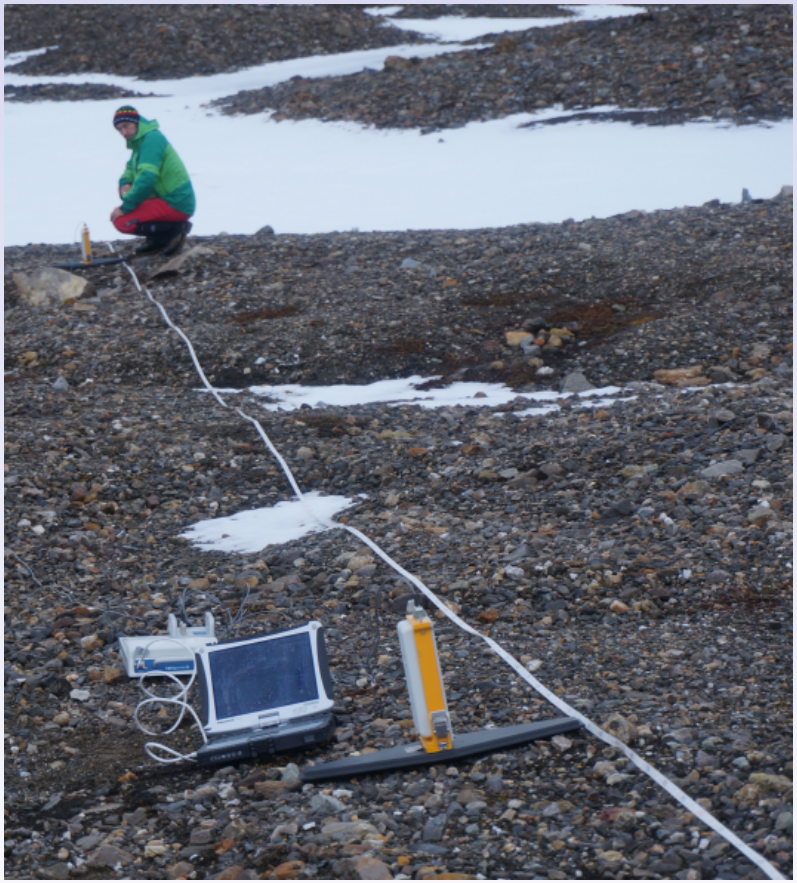
Resetting download quota on the SNCF WiFi connection

J.-M Friedt, *Modélisation et utilisation d'une parabole : application au wifi*, Opensilicium **20** 44–55 (Oct.-Dec. 2016)

Raw Ethernet

Embedded systems with little computational power: raw-Ethernet (Malå ProEx [11]), no routing (point to point communication PC-RADAR)

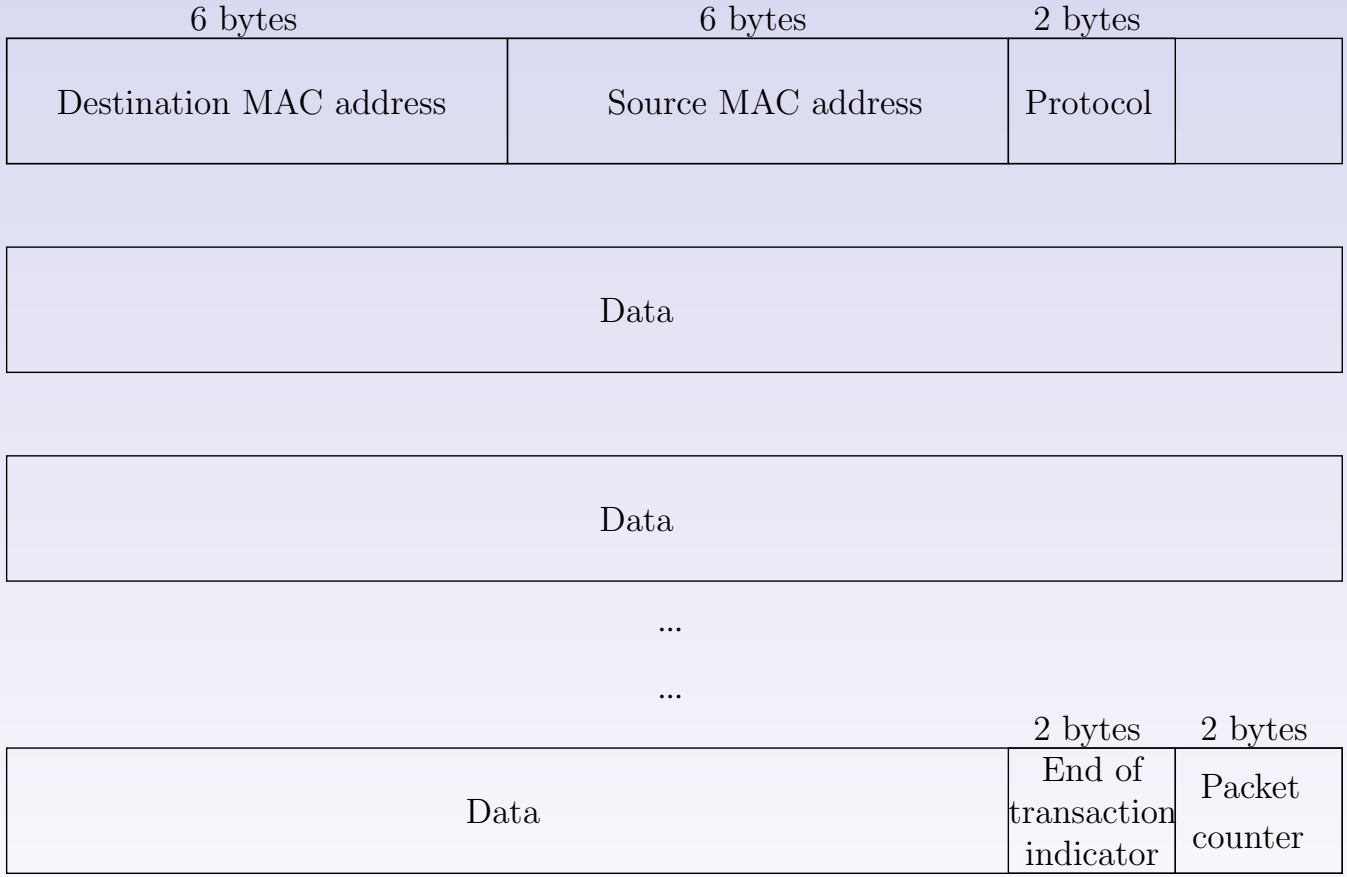




[11] `http://sourceforge.net/projects/proexgprcontrol/`,
A. Hugeat, J.-M Friedt, *A low cost approach to acoustic filters acting as GPR cooperative targets for passive sensing*, IWAGPR 2015

Raw Ethernet

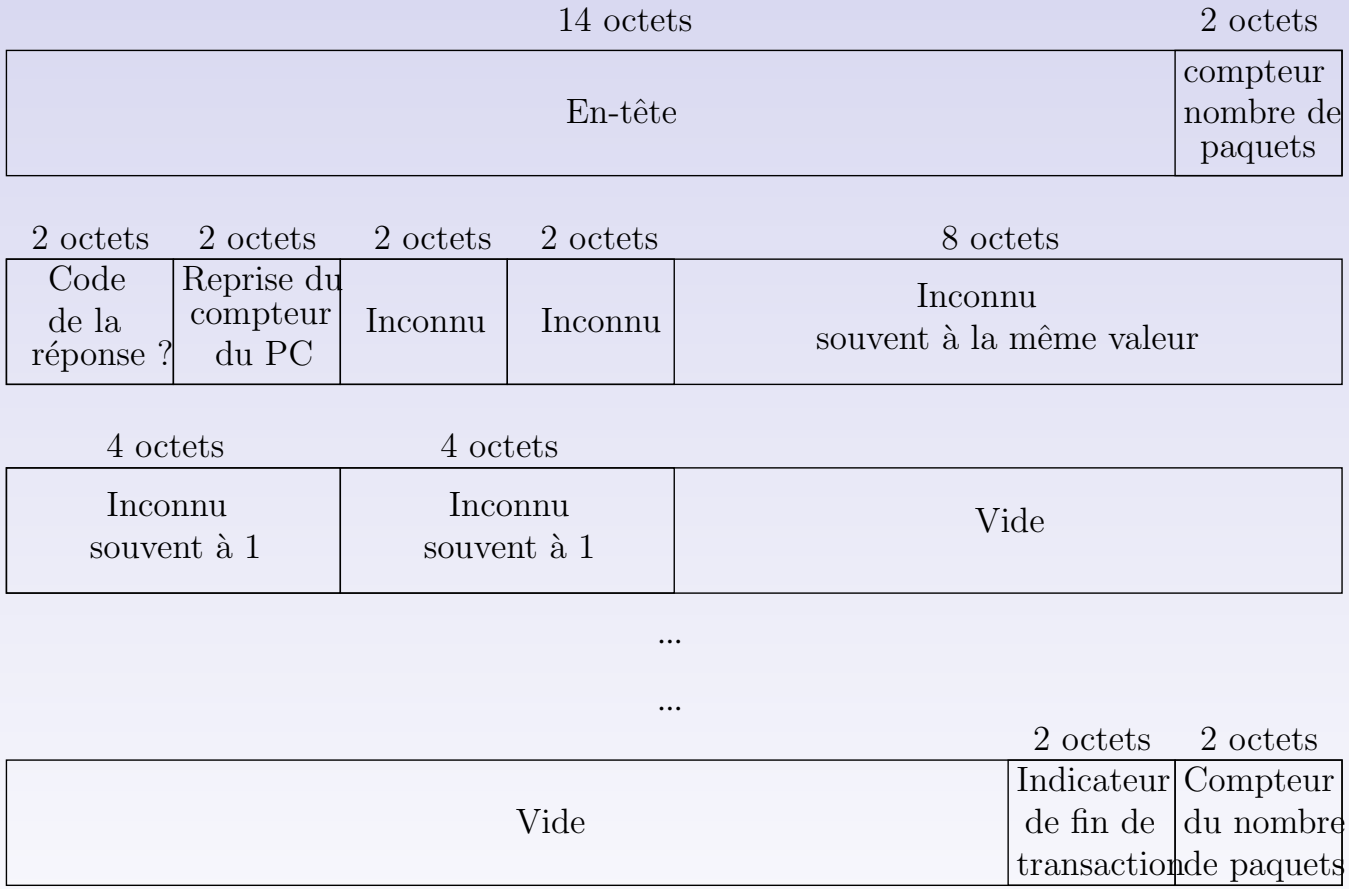
Embedded systems with little computational power: raw-Ethernet (Malå ProEx [11]), no routing (point to point communication PC-RADAR)



[11] `http://sourceforge.net/projects/proexgprcontrol/`,
A. Hugeat, J.-M Friedt, *A low cost approach to acoustic filters acting as GPR cooperative targets for passive sensing*, IWAGPR 2015

Raw Ethernet

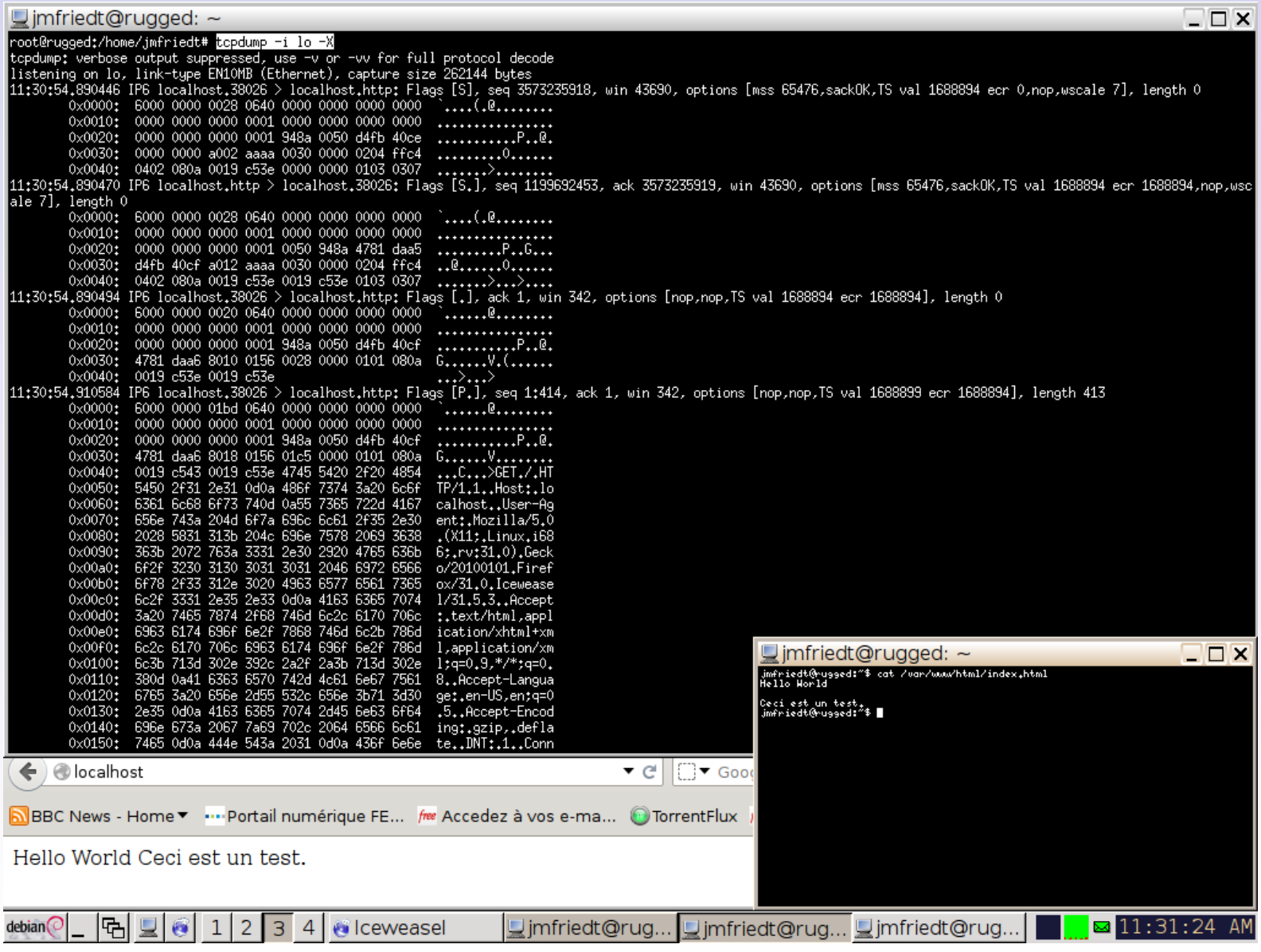
Embedded systems with little computational power: raw-Ethernet (Malå ProEx [11]), no routing (point to point communication PC-RADAR)



[11] `http://sourceforge.net/projects/proexgprcontrol/`,
A. Hugeat, J.-M Friedt, *A low cost approach to acoustic filters acting as GPR cooperative targets for passive sensing*, IWAGPR 2015

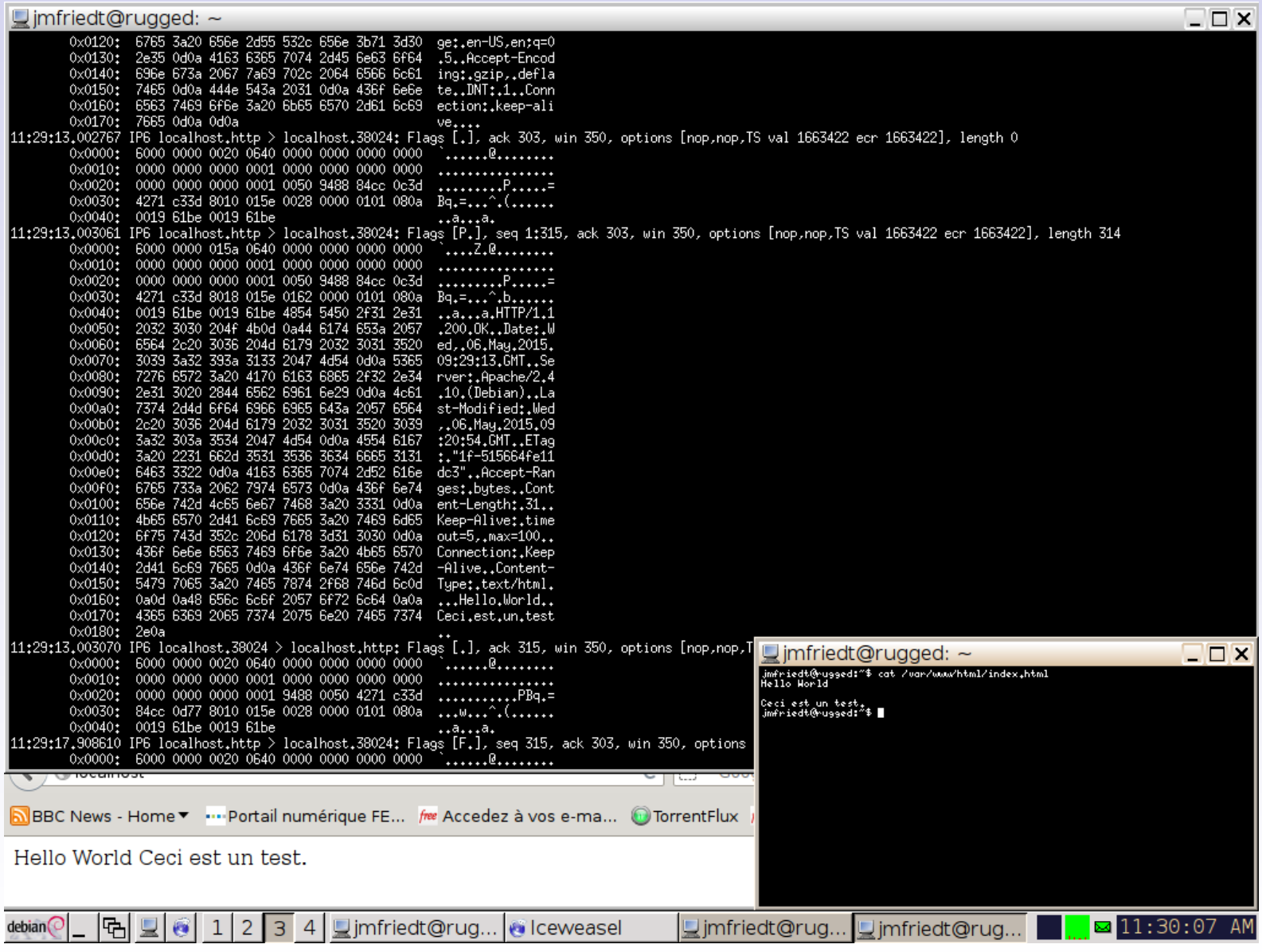
Listening packets ... tcpdump

Example: `tcpdump port 80 -i lo -X`



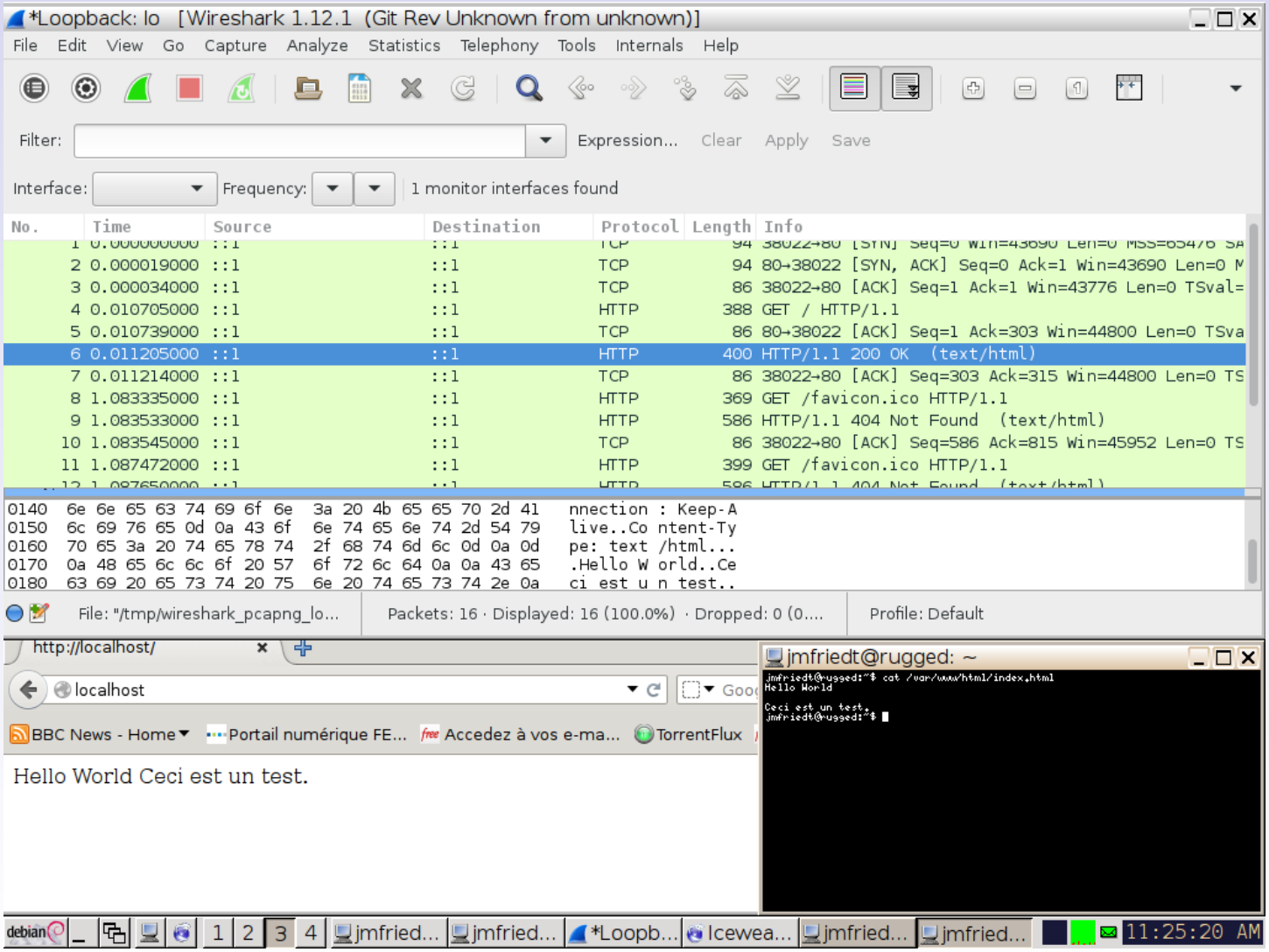
Listening packets ... tcpdump

Example: `tcpdump port 80 -i lo -X`



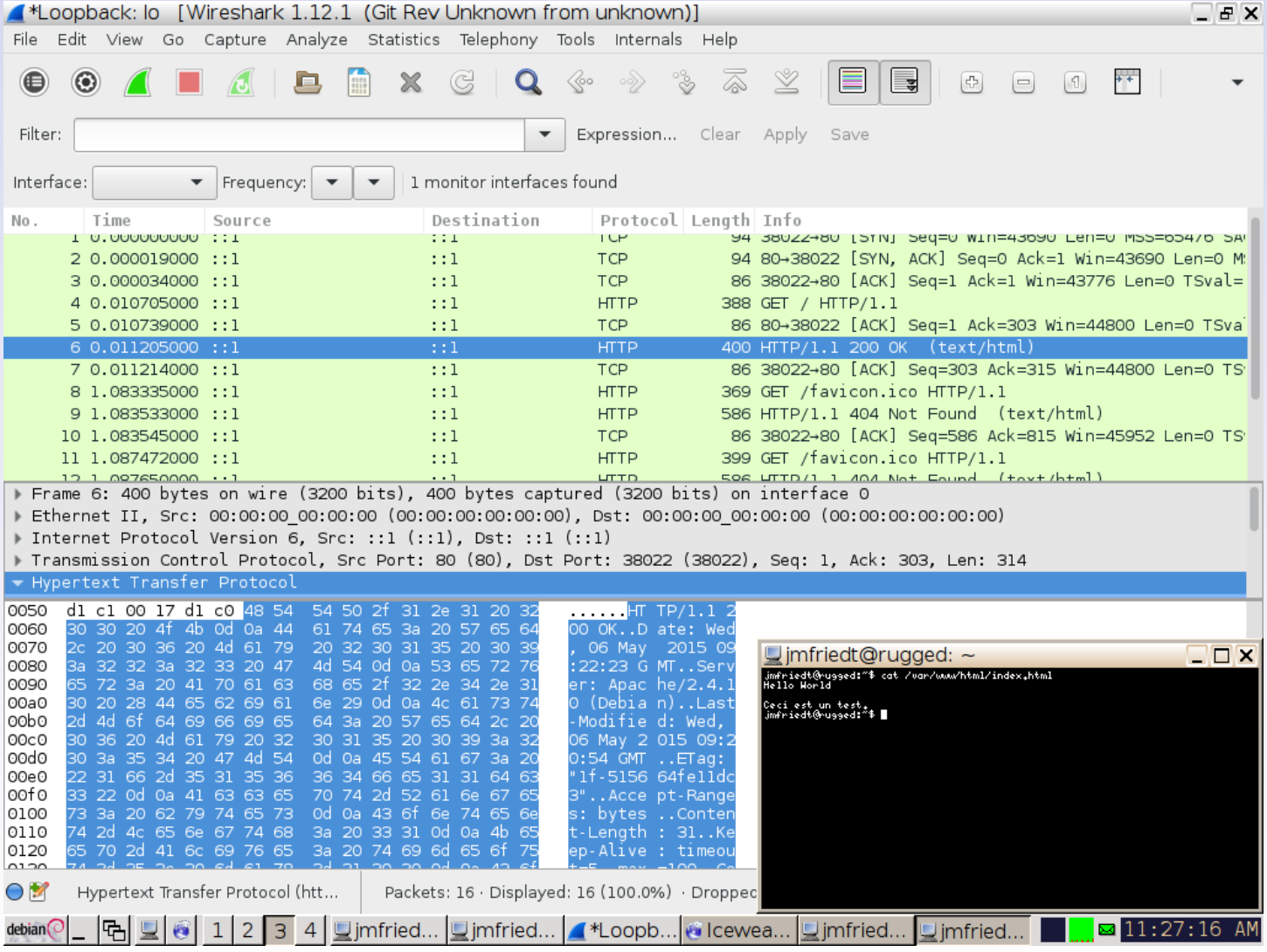
Listening packets ... wireshark

Ex-ethereal, wireshark provides a graphical user interface for analyzing packets



Listening packets ... `wireshark`

Ex-`ethereal`, `wireshark` provides a graphical user interface for analyzing packets



TCP/IP tools

- traceroute

```
jmfriedt@vm1:~$ traceroute www.whitehouse.gov
traceroute to www.whitehouse.gov (23.214.186.191), 30 hops max, 60 byte pac
 1  10.10.0.254 (10.10.0.254)  0.377 ms  0.370 ms  0.362 ms
 2  vss-5b-6k.fr.eu (46.105.123.252)  0.954 ms  0.952 ms  0.947 ms
 3  rbx-g1-a9.fr.eu (178.33.100.29)  2.509 ms  2.509 ms  2.505 ms
 4  * * *
 5  ldn-5-6k.uk.eu (213.251.128.18)  48.180 ms * *
 6  * * *
 7  ae10.mpr2.lhr2.uk.zip.zayo.com (64.125.31.194)  6.492 ms  8.722 ms  7.2
 8  ae5.mpr1.lhr15.uk.zip.zayo.com (64.125.21.10)  4.519 ms  4.515 ms  4.49
 9  94.31.61.250.IPYX-074083-001-ZYO.above.net (94.31.61.250)  4.504 ms  4.
10  a23-214-186-191.deploy.static.akamaitechnologies.com (23.214.186.191)
```

- nslookup

```
jmfriedt@rugged:~$ nslookup www.femto-st.fr
Server:         130.67.15.198
Address:        130.67.15.198#53

Non-authoritative answer:
Name:   www.femto-st.fr
Address: 195.83.19.10
```

HTTP tools

- Swiss army knife: nc (netcat)

```
echo -e "GET http://jmfriedt.free.fr HTTP/1.0\n\n" | nc jmfriedt.free.fr 80
```

- wget – GET method (answer: 0:47 1:57 2:25)

```
dest=Vesoul
wget -q -O- "http://reiseauskunft.bahn.de/bin/query.exe/fn?revia=yes&\
existOptimizePrice=1&country=FRA&\
dbkanal_007=L01_S01_D001_KIN0001_qf-bahn_LZ003&\
ignoreTypeCheck=yes&S=Besancon+Franche+Comte+TGV&REQ0JourneyStopsS0A=7&Z=${
REQ0JourneyStopsZ0A=7&trip-type=single&date=Me%2C+07.01.15&time=08%3A37&\
timesel=depart&returnTimesel=depart&optimize=0&infant-number=0&tariffTravel
tariffTravellerReductionClass.1=0&tariffTravellerAge.1=&qf-trav-bday-1=&tar
start=1&qf.bahn.button.suchen=" | grep -A1 uratio | grep ^[0-9] | tr '\n' '
```

- curl – POST method

```
month=12
year=2014
curl -s --cookie-jar test --data \
"anzahlprofile=1&einheit=meter&vonkurz=LFPG&nachkurz=LYR+&landetime=&zwpunk
adresse=&profil=profilangeben&day=15&month=$month&year=$annee&flightnumber=
von=PARIS%2C+FRANCE++++++++++++++++++&uptime=00%3A30&obentime=04%3A00&flugho
downtime=00%3A30&nach=LONGYEARBYEN%2C+NORWAY++++++++++&send=Send+request" \
http://www.helmholtz-muenchen.de/epcard/eng_flugoutput.php | grep Sv
```

Conclusion

Exercise: listen to the `lo` interface with `tcpdump` while running a ftp transaction. Can you see the login/password sent over the network?

Conclusion:

- Exploration of the various hierarchical layer of Internet network transactions (formally known as OSI layers)
- Description of high level protocols (SMTP, FTP, HTTP)
- Description of low level transactions (IP, TCP, UDP)
- Description of debugging tools (`tcpdump`, `wireshark`)

Perspectives:

- *data mining* [12] (automate collecting data over the Internet) [13]
- instrument control (embedded Linux)

[12] T. Segaran & J. Hammerbacher, *Beautiful Data – The Stories Behind Elegant Data Solutions*, O'Reilly Media (2009)

[13] J.-M Friedt, *Cartographier le bout du monde*, GNU/Linux Magazine France 185 (Sept. 2015)