Digital electronics

J.-M Friedt

FEMTO-ST/time & frequency department

jmfriedt@femto-st.fr

slides at `jmfriedt.free.fr`

February 19, 2025

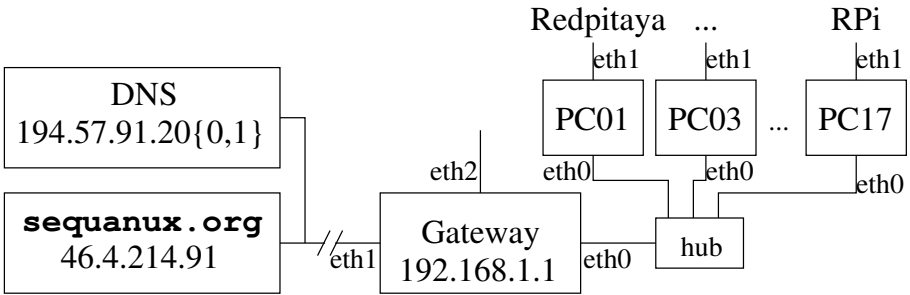
Plan

7 lessons/lab sessions (4-hour long schedules):

1. Executive environments: principles and introduction, getting started with FreeRTOS
2. FreeRTOS, RTEMS, Nuttx ... multitasking and associated methods to make sure shared data and resources are kept in known states (mutex & semaphore)
3. Using the scheduler, mutex and semaphores to solve the "philosopher problem"
4. Embedded systems with GNU/Linux – POSIX compatible operating system
   Architecture of an operating system, kernel v.s userspace
   Internet connectivity and networking
5. Accessing hardware resources from userspace – memory translation from physical to virtual address space (Memory Management Unit) – `/dev/mem`
6. Accessing hardware resources from a web server – internet connected instrument
7. From userspace to kernel space: character device (*char device*) for communicating between users and the kernel

Network [1] configuration

7 Application layer (DNS, FTP, NTP, HTTP ...)
6 Presentation layer (ASCII, EBDIC ...)
5 Session layer (socket)
4 Transport layer (UDP, TCP, ICMP)
3 Network layer (IPv4, IPv6, ARP – arp = link between 2 & 3, route)
2 Data link layer – MAC (AppleTalk, 802.3 "Ethernet": ifconfig)
1 Physical layer (RS232, 10BaseT, 802.11 "Wi-Fi")



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

Networking commands

- `ifconfig` for interface configuration: define IP (logical) address or possibly the MAC (hardware) address
- `route` for routing configuration
- default routing to the interface subnet
- private network address range whose packet will not propagate on the internet: **10.x.x.x**, **172.16.x.x** to **172.31.x.x**, **192.168.x.x**
- all computers have a self address: **127.0.0.1**
- "hot potato routing": check if the IP address matches a local subnet, otherwise route to the default gateway
- UDP (Datagram) = broadcast, connectionless communication, no service quality but low latency/overhead
- TCP (Transmission Control Protocol) = connected communication making sure each packet has reached the destination in the right order ("TCP stack") [2]
- ICMP (ping): network management
- IP-domain name: DNS (set in `/etc/resolv.conf`)

[2] request retransmission if packet is lost, arranges received packets transmitted through different routes in the right order

Networking commands: from net-tools to ip

net-tools package commands	ip command options
ifconfig -a	ip addr or ip a
ifconfig eth0	ip addr show eth0
ifconfig eth0 IP_ADDR	ip addr add IP_ADDR/24 dev eth0
ifconfig eth0:1 IP_ADDR2	ip addr add IP_ADDR2/24 dev eth0 label eth0:1
ifconfig eth0 IP_ADDR netmask 255.255.255.0	ip addr add IP_ADDR/24 dev eth0
route add default gw IP_GW	ip route add default via IP_GW
ifconfig eth0 up	ip link set eth0 up
ifconfig eth0 down	ip link set eth0 down
	ip addr del IP_ADDR dev eth0
arp -a	ip neigh

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether cc:7e:e7:5f:cf:6e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.55/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 192.168.2.1/32 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::ce7e:e7ff:fe5f:cf6e/64 scope link
       valid_lft forever preferred_lft forever
```

Netmask (subnet size): /24: first 24 bits of the mask set to 1, equivalent to 255.255.255.0

GNU/Linux network configuration

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. routing table with automatic association of the "appropriate" interface with each subnetwork + *default gateway*,

```
jmfriedt@dhcp-221:~$ /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
```

GNU/Linux Wi-Fi configuration

```
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 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
```

GNU/Linux Wi-Fi configuration

```
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"
```

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

Automating network configuration

- Network interface default settings in `/etc/network/interfaces`
- Example

```
auto lo
iface lo inet loopback

iface usb0 inet dhcp

auto eth0
iface eth0 inet static
  address 192.168.1.55
  netmask 255.255.255.0
  gateway 192.168.1.1
```

- manually activate or deactivate an interface: `ifup` and `ifdown` [3] (in `/sbin`)
- prevent Network Manager from messing with `ifup` settings: edit configuration in `/etc/NetworkManager/conf.d/` and add

```
[keyfile]
unmanaged-devices=interface-name:eth*
```

or `nmcli dev set eth0 managed no` or

```
[ifupdown]
managed=False
```

[3] `ifupdown` package for Debian GNU/Linux

TCP server (C)

- a server *waits* for connections, a client *connects* to a server to initiate communication
- `telnet`: the "universal" interactive TCP client
- `netcat`: UDP/TCP command line swiss army knife

```c
#include <sys/socket.h>
#include <resolv.h>
#include <unistd.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];
 // socket type (AF = IPv4, STREAM=TCP)
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 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);                          // wait for incoming connection
 while (1)
 {struct sockaddr_in client_addr;
  int mysize,clientfd;
  unsigned int 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));
  mysize=recv(clientfd, buffer, MAXBUF, 0);
  send(clientfd, buffer, taille, 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*

TCP server (Python)

- a server *waits* for connections
- a client *connects* to a server to initiate communication
- `telnet`: the "universal" interactive TCP client
- `netcat`: UDP/TCP command line swiss army knife

```
import socket
import string
while True:
    sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('127.0.0.1', 4242))
    print("Waiting for connection")
    sock.listen(1)
    conn, addr = sock.accept()
    with conn:
        print('connected from ',addr)
        while True:
            data=conn.recv(1)
            if data:
                data=data.decode()
                print(data)
                if 'q' in data:
                    sock.shutdown(socket.SHUT_RDWR)
                    sock.close()
                    break
```

Easiest way to test a TCP server: `telnet IP port`

UDP server/client[4]

A server waits for a connection

```
#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 start a transaction

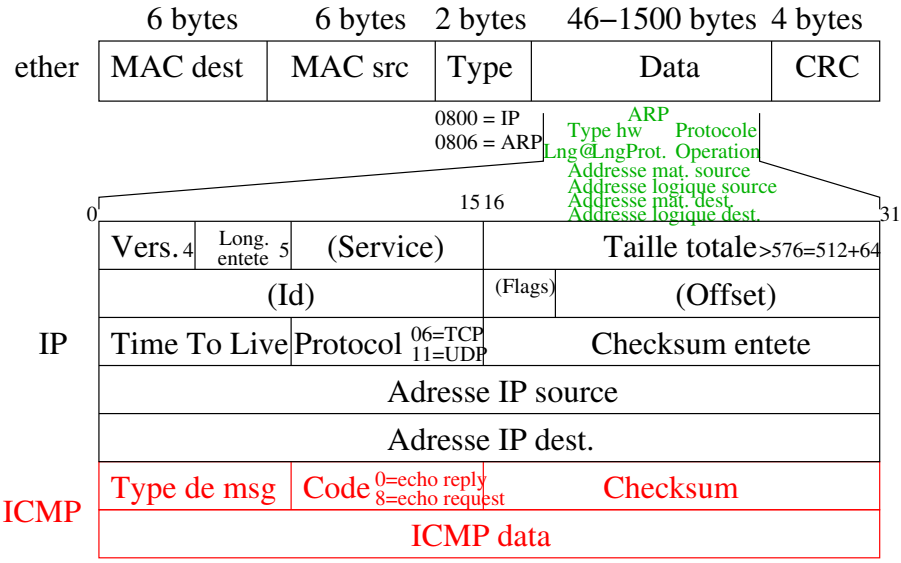
```
#include <sys/socket.h>
#include <resolv.h>
#include <arpa/inet.h>

#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;
}
```

Cross-compile for the targeted embedded board – portability by embedding OS

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

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

Raw IP and ICMP



Question: what is the ICMP identifier in the IP packet header? `sudo tcpdump -i lo -x -e` **during** `ping lo`

Raw IP and ICMP

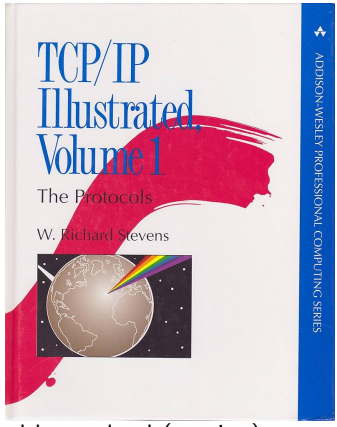
```
$ arp -a
? (192.168.0.1) at 00:13:d3:8d:d3:97 [ether] on eth0
$ arp -d 192.168.0.1
$ ping 192.168.0.11

# 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 28
        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), length 46
        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, seq 1, length 64
        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

A good understanding of network protocols is needed for applications related to

- security: firewall or eddicated router (`iptables`), packet filtering, quality of service and minimizing packet handling duration,
- remote instrument control. Even without consider data security, avoid DoS attacks.
- all OSI layers are often not needed for embedded applications: understand the whole communication chain to only keep the mandatory parts (e.g. raw-IP).

In our examples, the argument to `socket` defines the transport protocol (`SOCK_DGRAM` (UDP), `SOCK_STREAM` (TCP)).

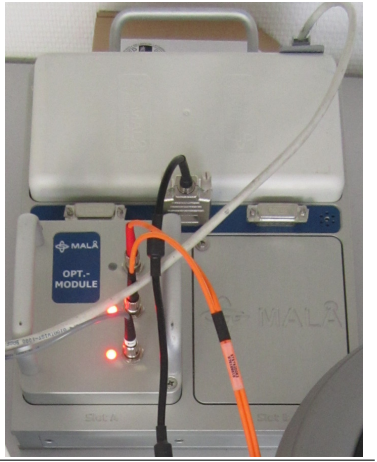


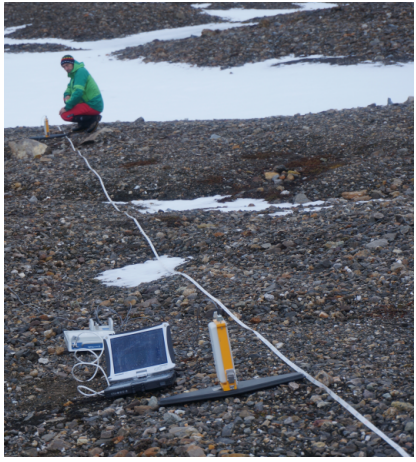
`SOCK_RAW` $\Rightarrow$ 20 byte header including source and destination IPs followed by payload (routing)

```
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
```

Raw Ethernet

For light, low power embedded systems: raw-Ethernet (Malå ProEx [5]), no routing (point to point link between PC and RADAR control unit)

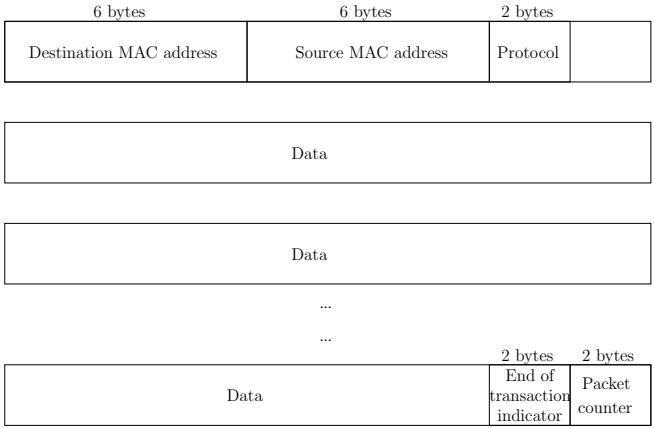




[5] `http://sourceforge.net/projects/proexgprcontrol/`,
A. Hugeat, J.-M Friedt, *A low cost approach to acoustic filters acting as GPR cooperative targets for passive sensing*, IWAGPR 2015

Raw Ethernet

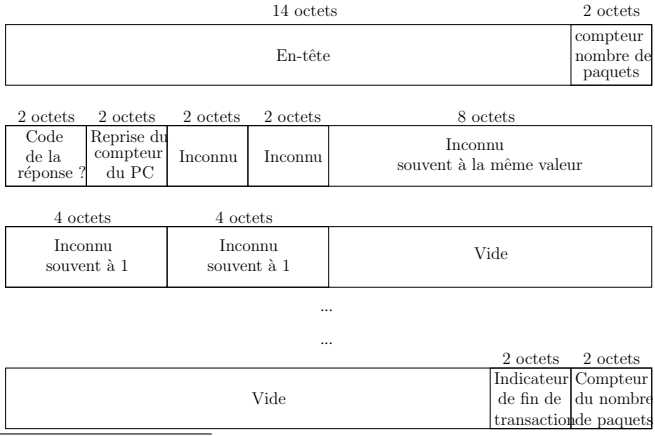
For light, low power embedded systems: raw-Ethernet (Malå ProEx [5]), no routing (point to point link between PC and RADAR control unit)



[5] `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

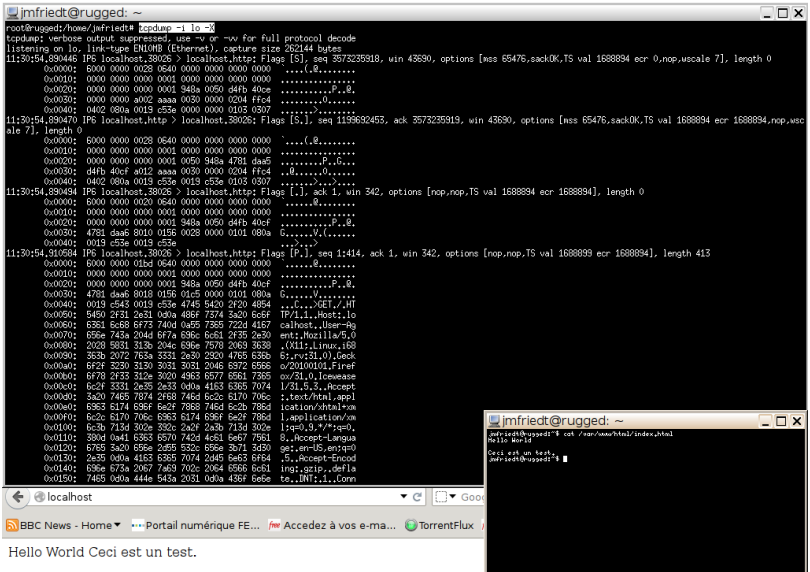
For light, low power embedded systems: raw-Ethernet (Malå ProEx [5]), no routing (point to point link between PC and RADAR control unit)



[5] `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 to packets ... tcpdump

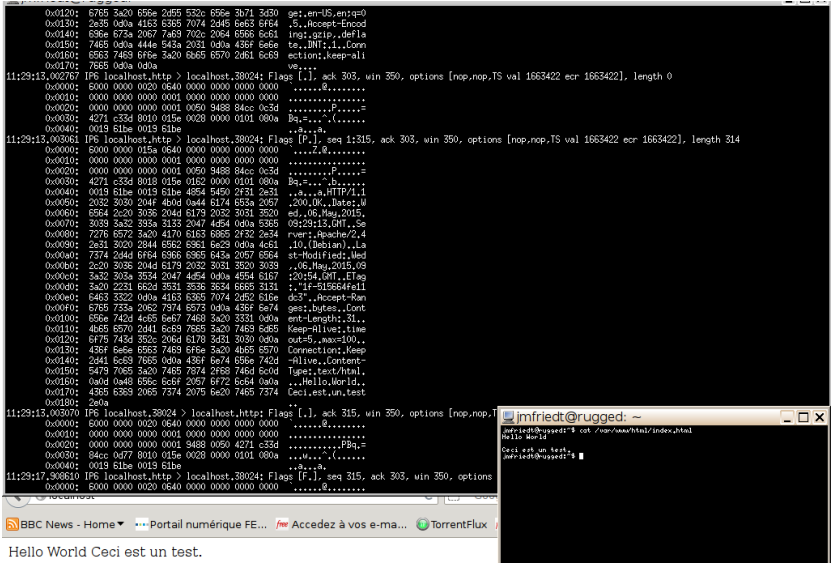
Example: `tcpdump port 80 -i lo -X`



Hello World Ceci est un test.

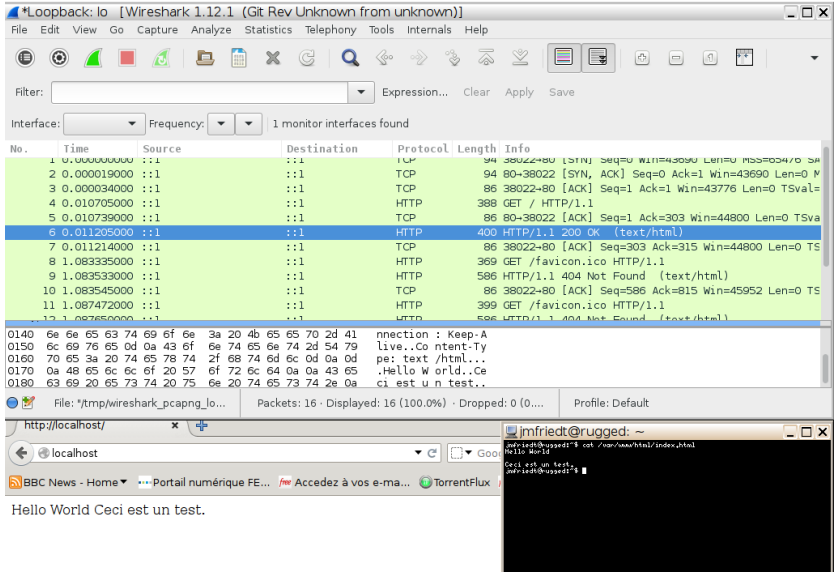
Listening to packets ... tcpdump

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



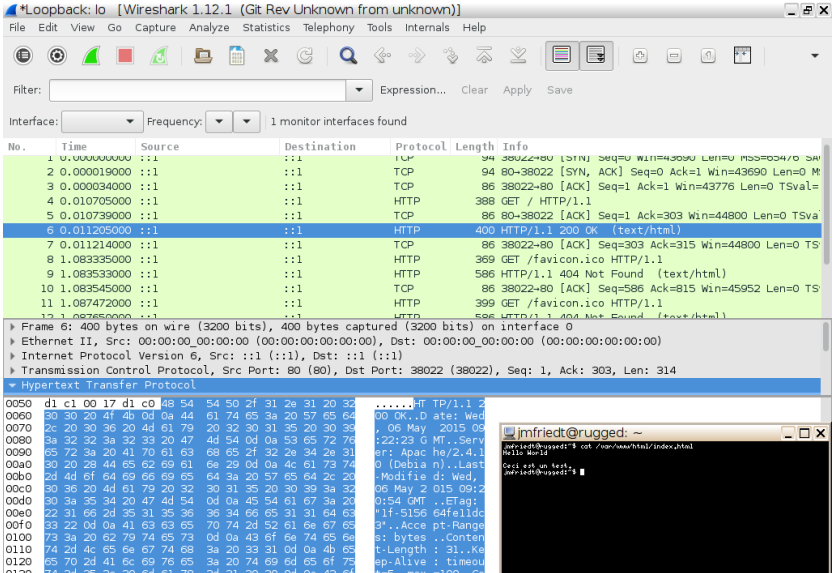
Listening to packets ... wireshark

Former-ethereal, wireshark for a graphical user interface to analyzer packet contents



Listening to packets ... wireshark

Former-ethereal, wireshark for a graphical user interface to analyzer packet contents



TCP/IP tools

- traceroute

```
jmfriedt@vm1:~$ traceroute www.whitehouse.gov
traceroute to www.whitehouse.gov (23.214.186.191), 30 hops max, 60 byte packets
 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  7.263 ms
 8  ae5.mpr1.lhr15.uk.zip.zayo.com (64.125.21.10)  4.519 ms  4.515 ms  4.496 ms
 9  94.31.61.250.IPYX-074083-001-ZYO.above.net (94.31.61.250)  4.504 ms  4.500 ms  4.495 ms
10  a23-214-186-191.deploy.static.akamaitechnologies.com (23.214.186.191)  4.493 ms  4.488 ms  4.793 ms
```

- 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

- The swiss army knife: nc (netcat)

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

- wget – GET ethod (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=${dest}&\
REQ0JourneyStopsZ0A=7&trip-type=single&date=Me%2C+07.01.15&time=08%3A37&\
timesel=depart&returnTimesel=depart&optimize=0&infant-number=0&tariffTravellerType.1=E&\
tariffTravellerReductionClass.1=0&tariffTravellerAge.1=&qf-trav-bday-1=&tariffClass=2&\
start=1&qf.bahn.button.suchen=" | grep -A1 uratio | grep ^[0-9] | tr '\n' ' '
```

- curl – POST method [6]

```
month=12
year=2014
curl -s --cookie-jar test --data \
"anzahlprofile=1&einheit=meter&vonkurz=LFPG&nachkurz=LYR+&landetime=&zwpunkte=&\
adresse=&profil=profilangeben&day=15&month=$month&year=$annee&flightnumber=XX001&\
von=PARIS%2C+FRANCE++++++++++++++++++&uptime=00%3A30&obentime=04%3A00&flughoehe0=10000&\
downtime=00%3A30&nach=LONGYEARBYEN%2C+NORWAY++++++++++&send=Send+request" \
http://www.helmholtz-muenchen.de/epcard/eng_flugoutput.php | grep Sv
```

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

Probing the network

- scan network: `netdiscover`

```
Currently scanning: 192.168.69.0/16   |   Screen View: Unique Hosts

5 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 300
-------------------------------------------------------------------------------
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-------------------------------------------------------------------------------
192.168.1.1     e4:54:e8:ad:cb:0a      1      60  Dell Inc.
192.168.1.3     e4:54:e8:ad:ca:ae      2     120  Dell Inc.
192.168.2.1     e4:54:e8:ad:ca:ae      1      60  Dell Inc.
192.168.2.1     e4:54:e8:ad:cb:0a      1      60  Dell Inc.
...
```

- scan network: `sudo nmap -sn 192.168.1.1/24`

```
Starting Nmap 7.93 ( https://nmap.org ) at 2025-02-18 09:17 CET
Nmap scan report for 192.168.1.1
Host is up (0.0016s latency).
Nmap scan report for 192.168.1.3
Host is up (0.0015s latency).
Nmap scan report for 192.168.1.42
Host is up (0.00023s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.51 seconds
```

- list open ports: `sudo nmap 192.168.1.1`
- OS identification: `sudo nmap -O 192.168.1.1`

Internet documentation

Request for Comments (RFC) at `www.ietf.org/rfc`
Most common protocols

- HTTP = RFC 2068 (Jan. 1997) and 2616 (Jun. 1999),
- SMTP = RFC 772 [7] (Sep. 1980) and 821 (Aug. 1982)
- FTP = RFC114 et successeurs, RFC542 (Aug. 1973), RFC959 (Oct. 1985)

Available services, used protocol and connection port are standardized and documented:
`/etc/services`

Too low computational power makes vulnerable to **DoS attacks**.

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

Practical demonstration

1. configure the network so the embedded board can communicate through a TCP/IP (IP = Internet Protocol – `ifconfig`, TCP = connected protocol = `rlogin/ssh`) link with the host computer
2. launch a web server on the embedded board (`lighttpd`) and display the output of an `index.html` static web page
3. check that you can control the LED state through the standard `/sys/class/leds` interface
4. create a dynamic web interface (CGI: Common Gateway Interface) in which a script parses the argument of the URL and acts on the LED accordingly. Be careful with permissions (who owns the web server? who owns the LED?)