GLab: Frequenty Asked Questions

Christian Grothoff, Hansjürg Wenger

May 6, 2021

1 C Programming

1.1 How do I print a MAC address?

Use something like this:

```
static void
print_mac (const struct MacAddress *mac)
{
  print ("%02X:%02X:%02X:%02X:%02X:%02X\n
         mac->mac[0], mac->mac[1],
         mac->mac[2], mac->mac[3],
         mac->mac[4], mac->mac[5]);
}
```

1.2 How do I compare two MAC addresses?

Use something like this:

```
static int
maccmp (const struct MacAddress *mac1,
        const struct MacAddress *mac2)
{
  return memcmp (mac1, mac2, sizeof (struct MacAddress));
}
```

1.3 How do I print an IPv4 address?

Use something like this:

```
#include <arpa/inet.h>

static void
print_ip (const struct in_addr *ip)
{
  char buf[INET_ADDRSTRLEN];
```

```
  print ("%s\n
         inet_ntop (AF_INET,
                    ip,
                    buf,
                    sizeof (buf)));
}
```

1.4 How do pointers work again?

Consider this code:

```
int i = 5;
int *j = &i;
int k = *j;
```

i is an integer. When you write just i, you will get the value 5. i is stored in computer memory. When you write $\&i$, you get the address where i is stored in memory. j is a pointer to an integer. So the value of j is an address in memory where we expect to find an integer. If we want to read the integer at that address, we write $*j$ to dereference the address j.

1.5 How do arrays work again?

Arrays in C are represented as pointers to the first element of the array.

Consider this code:

```
int i[] = { 5 , 6 };
int *j = i;
i[0] + j[1]; // 11
```

Here i is declared as an array with two values, 5 and 6. j is, like i, a pointer. You can treat pointers like arrays and vice-versa. The only difference is that `sizeof(i)` will give you the size of the array as the C compiler knows how large the array is in memory, while `sizeof(j)` will give you the size of (any) pointer.

The following code generates exactly the same situation as the fragment above:

```
int i[2];
int *j = i;
*i = 5;
*(j+1) = 6;
```

1.6 How do I get a pointer to the IPv4 header?

A slightly unclean[1] minimalistic solution that works fine on your CPUs would look like this:

[1] due to unaligned pointer access

```
static void
handle_frame (uint16_t interface,
      const void *frame,
      size_t frame_size)
{
  struct EthernetHeader *eh = frame;
  struct IPv4Header *ip = (struct IPv4Header *) &frame[1];

  if (frame_size < sizeof (struct EthernetHeader) +
                   sizeof (struct IPv4Header))
    fail ();
  // use(ip); here!
}
```

1.7 Why does my `printf` not work?

`printf` prints to `stdout`, which in your setup is the network-driver which expects to receive frames to be transmitted. To output to the console, you need to give a special command to the network-driver. This is provided by the `print()` function, which thus replaces `printf` for your project. You can also directly use `fprintf (stderr, ...)` to write to `stderr` for logging. Note that required output to the user (for grading) must be generated using `print`.

1.8 I am using `print`, and it still does not work!

You might be having a problem with terminal discipline. `print`ing to `stdout` is by default buffered until a newline is encountered. Make sure to terminate your output with "
n".

2 Setup

2.1 Why do I get "permission denied" when running the network-driver, even as root?

The latest version of the Ubuntu operating system refuses to grant RAW socket access to binaries that are in the `/home` directory, even if run by `root`. Copying the binary to `/root` or /tt /usr/sbin makes it work:

```
# cp network-driver /usr/sbin/network-driver
# /usr/sbin/network-driver eth0 - ./parser
```

2.2 What should my network topology look like?

Figure 1 shows the suggested setup. If you are a team of three students, you may want to connect a third notebook. If you are using the Banana PI, you must use Ethernet USB adapters instead of the Ethernet ports of the PI for the connections to the notebooks for the `hub`. `switch` and `vswitch` programs.

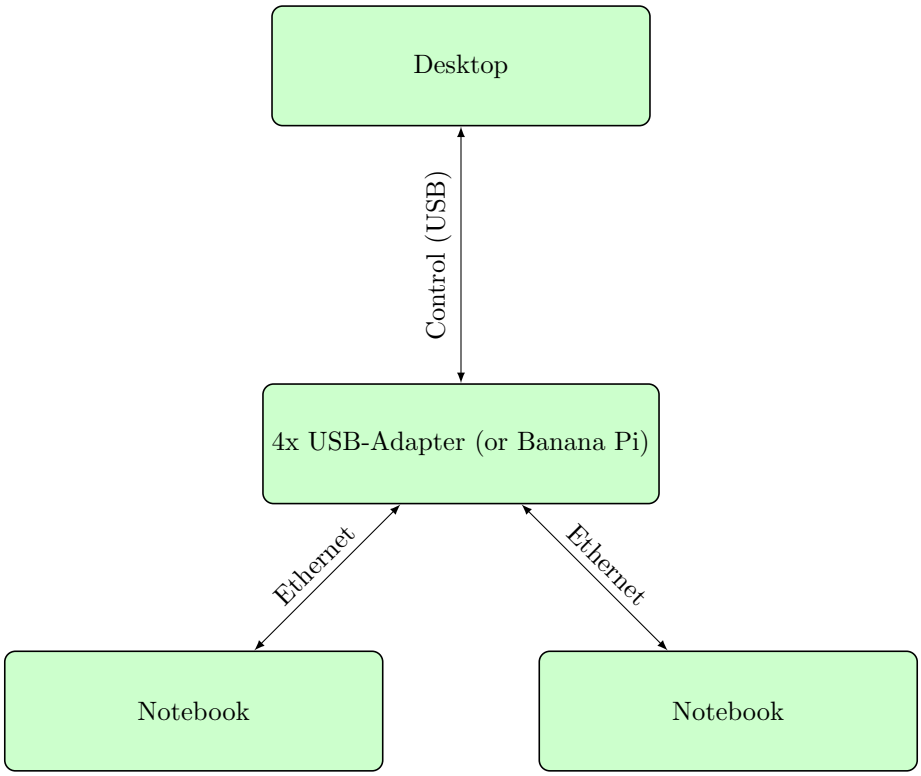

Figure 1: Suggested network topology.

2.3 My (new) code does not seem to work at all. I've been ignoring the *clock skew* warning from `make`. What is going on?

The PI does not have a real-time clock. Thus, after a reboot, it can find itself in the past. Your latest edits receive time stamps that pre-date the binaries it created in the past before its time travels. Make builds programs using timestamps. Thus, `make` did NOT rebuild your code, but it figured something might be fishy because your binaries are from the future. As your binary is your old binary, your new features do not work at all.

Solution: If you ever get a *clock skew* warning, run `make clean` to delete all old binaries.

2.4 My notebook cannot ping the desktop

This is normal, as your switch/router logic does not forward the traffic to the desktop's host operating system. Your notebooks can only reach systems that *your* hub, switch or router allows them to reach.

2.5 How do I debug my code *without* `fprintf()`?

First, make sure you compiled your code with debug symbols and without optimizations (`gcc -g -O0`). Then, launch your program as usual:

```
# ./network-driver IFCs - prog ARGs
```

Then, in another shell, check with `ps` which PID your "prog" process has:

```
# ps ax | grep prog
```

Now you can attach `gdb` to your running program (substitute $PID for the PID you got from `ps`):

```
# gdb prog
(gdb) attach $PID
```

You should now set a breakpoint at a location where you want to start debugging, and then continue execution until the breakpoint is reached:

```
(gdb) break handle_frame
(gdb) continue
```

Then, use "CTRL-x a" to enable GDB's code inspection mode to see where you are executing. Use "n(ext)" to execute the next statement. Use "s(tep)" to step into a function call. Use "print" or "x" (eXamine) to inspect variables and memory. Use "cont(inue)" to run until the next breakpoint. Use "q(uit)" to exit `gdb`. Note that your process will continue.

Final note: this only works if your program does not crash before you can attach `gdb`. Enable core dumps and inspect the crash with gdb if your program crashes:

```
# ulimit -c 999999
# echo core > /proc/sys/kernel/core_pattern
# ./network-driver ... # reproduce crash
# gdb prog core
(gdb) bt full # view stack trace at time of crash
```

Read the `gdb` manual for more information on how to use the GNU Debugger!

2.6 I get not network traffic on the interface. Why?

The most common cause is that you did not enable the network adapter, so the interface is still physically down. For each network interface, run:

```
# ip link set up dev INTERFACENAME
```

You can check that it worked using:

```
# ip link list
```

If you still see a "DOWN" in the line, check the cable.

Your USB port may also supply insufficient power to the USB adapter. In this case, try another port.

3 Protocol details

3.1 MTU = 1500 or 1514?

When we traditionally speak of an MTU of 1500 for Ethernet, this *excludes* the Ethernet header (and CRC, and preamble). When reporting the MTU in an ICMP "fragmentation needed" message, the Ethernet header is also excluded (layer 3!).

Note that the MTU command-line argument (!) for the `router` *excludes* the Ethernet header, but **internally** the `mtu` member of the `struct Interface` em includes the Ethernet header (see check in `forward_to`). So you must subtract the size of the Ethernet header when generating the ICMP message!