CSSE2310/7231 — A.1

Layers continued & Programming

Note

Remember there is a distinction between *bandwidth* and *latency*.

- Bandwidth = how much information you can send per unit time. (bits/s)
- Latency = How long it takes for something to start (Time)

Layers

5 Application
4 Transport
3 Network
2 Data link
1 Physical

Where do addresses come from?

- ports — low ports aside, you can say which port you want to use when you set up a network connection.
- IP —
  - internal configuration
    - The administrator of the device chooses one.
    - If they choose badly it might not work. (more later)
  - external configuration
    - On startup the device can ask another device what address it should use
    - DHCP (more later)
- MAC
  - interfaces will have a default address set in hardware.
  - The address that is used could be the default or could be changed by software.

v4??

IPv4 has 32bit addresses. In theory that's $4,294,967,296$ addresses.

- N.A.T.[1] has probably helped reduce demand a bit.
- Addresses weren't allocated very efficiently.
  - US military has at least $134,217,728$ reserved[2].
  - Apple, Ford and Daimler have $16,777,216$ each.

For a while now, big blocks of addresses were handed out to regional "registries" and organisations which divided them up into smaller blocks to give to smaller organisations. . . . all the way down to your ISP giving you an address.

[1] later

[2] That's a decrease from what they had

We need more addresses

- The world ran out[3] of IPv4 addresses in 2011.
- The solution the IETF had proposed was IPv6 with 128bit addresses.

That's
$\ldots 340, 282, 366, 920, 938, 463, 463, 374, 607, 431, 768, 211, 456$
addresses.
Which is a lot.

[3] At the top level anyway

v6?

Why aren't we using IPv6 in the teaching and assessment in this course?

- It is possible to write code which is IP version agnostic.
  - So you wouldn't really learn anything different (at this level)
- Do you want 39 digit numbers in your exams?
  - Yes I know about the :: contraction convention

client / server

Client $\leftrightarrow$ Server

Server — a process which waits for requests from clients.

- eg: A web server waits on port 80 for browsers to connect and request pages.
- `sshd` waits for connections on port 22.

Client — a process which submits requests to a server.

- A web client connects to a server on port 80 and requests pages.
- `ssh` or `putty` connects to the server on port 22.

Client $\leftrightarrow$ Server

Unfortunate terminology:

- "server" refers to a process running on a machine.
  - Note: Server on port x, not port x on the server.
- "server" is also often used to refer to the hardware that the server process runs on.
  - We run servers on your servers ...

TCP Connections

- The "client / server" distinction applies to single connections.
- For any TCP connection there is always a client and a server.
- An application acting in a peer-to-peer mode, may have some connections for which it was the server and others for which it was the client.
- Once a connection has been established, there is no difference between what a client can do and what a server can do.

ncat / nc

You want to be able to debug your network code without needing both your client and your server working. Netcat to the rescue.
To start a server (it's `-ell`):

```
$ nc -4 -l 43210
```

Using IPv4 listen on port 43210.
To connect to your server

```
$ nc -4 localhost 43210
```

Using IPv4 connect to port 43210 on a computer called `localhost`.

Notes

- Connections are bi-directional (you can type into either and it gets sent to the other end).
- Ports are machine-wide so you will need to pick a number that noone else is using.
- On moss you can also use the machine name `localhost4` which only has a IPv4 address.
- `localhost` is defined on most systems to give **an** IP address of the machine you are on.
  - Remember, machines can have multiple names and multiple IPs

Warning:

- There are multiple versions of netcat out there.
- We will be using the version installed on moss. (In debian this is in the `netcat-openbsd` package.
- You may need to check the doco for the version on your system.

Client steps

1. Find out the address of the machine you wish to connect to.
2. Make a socket
   - A socket is an abstraction of a network endpoint.
   - There are different types of socket, we'll be using `SOCK_STREAM`
   - Sockets are file-descriptors
3. `connect()` to the server.
4. Wrap socket descriptor for nicer IO
   - You should `dup()` the descriptor before calling `fdopen()`

net1.c

See `net1.c`

So `struct addrinfo`

- contains a `struct sockaddr*`
- which is actually a `struct sockaddr_in*`
  - which contains a struct in_addr
    - which contains a in_addr_t which is an unsigned integer of some sort.

Why?

- The levels of indirection may seem execessive, but most of the time you won't be dealing with directly with all that detail.
- This allows different types of sockets to be used.
- Network byte order may be different to your machine's natural order.

See `net2.c`

Server

See `net3.c`

Server steps

1. Make a socket
2. (Optional) set parameters
3. `bind()` the socket to a port.
   - Otherwise it becomes:
   - "I have a server can you guess where?"
4. Set the socket to `listen()` for connections.
5. Call `accept()` to get allow a connection
   - Use the new fd to interact with the client

See `net3b.c` for options.

See `net4.c`

This would not be normal for a server

- normally you want people to connect to a known port
- we are doing it to avoid port collisions during the assignment.

ntohs?

`ntohs` = **n**etwork **to** **h**ost **s**hort
Converts a 16 bit value from network representation to the machines normal ordering.
Others:

- `ntohl`
- `htons`
- `htonl`

Multi-programming

`accept()` is a blocking call. If you want to be able to work on a connection **and** wait for new connections you need extra workers. Possibilities:

- `fork()` and let the new child handle the new connection.
- create a pthread to handle the new connection.
- use a non-blocking call to check for IO and connections (not in this course).