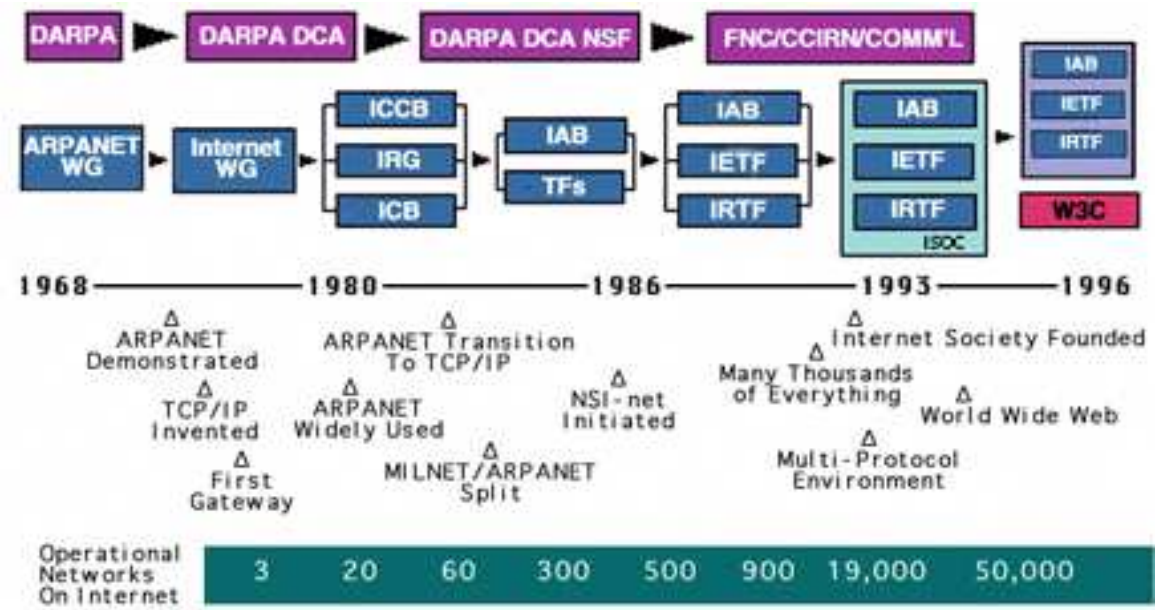
CS214 - Advanced UNIX

Lecture 9 - Networking

Ymir Vigfusson

The Internet



[From http://www.isoc.org/internet/history/brief.shtml]

Motivation

What happens when I type *http://www.cornell.edu/index.html* into my browser?

- A *resolver* realizes “www.cornell.edu” is not an IP address, the unique host identifier on the Internet.
- It contacts its *domain name server* (DNS) to resolve the hostname into an IP address.

Motivation

- The DNS server in turn asks the root name servers for authoritative servers for ".edu", asks those for "cornell.edu" and finally asks the cornell.edu name server for "www.cornell.edu".
- The final answer is 128.253.161.221

Motivation

- The browser now opens a socket, and tells the operating system to connect to 128.253.161.221, port 80 (for HTTP).
- The browser can now send requests and receives answers through this *TCP* (Tranfer Control Protocol) connection, much like it would when interacting with a UNIX file.

Motivation

- The TCP protocol abstracts away the fact that data is actually broken up into smaller pieces called *packets*, some of which may need to be retransmitted or arrive in the wrong order.

Motivation

- What the browser now transmits over the open socket is a *HTTP request*, which is a part of the HTTP protocol. It may look like this.

```
GET /index.html HTTP/1.0
```

Motivation

- The server sends back the contents of *index.html* to the client.
- For any image in the HTML file there is a separate GET request sent for that particular image. Sometimes this is done over the same connection, sometimes a new one is spawned for each one.
- The browser can now render the HTML contents on the screen.

Networks

- Hosts on the Internet communicate via unique IP addresses associated with them. An IP address has the format *xxx*.*xxx*.*xxx*.*xxx* where each *xxx* is a number between 0 and 255.
- You could visit e.g. `www.whatismyip.com` to see what IP address your computer uses.
- The command `ifconfig -a` will list all network interfaces and associated IP addresses. For instance `inet addr:192.168.1.104` tells me I'm 192.168.1.104.

Networks

- Each host has a list of *routers* or *gateways* that govern through what path Internet packets get sent.
- See `route -n` to see your current routes.

```
Dest.   Gateway       Genmask  ... Iface
0.0.0.0 192.168.1.1   0.0.0.0      eth1
```

- says that traffic to *all* hosts should be routed through 192.168.1.1 on network interface card 1 (in this case, my wireless card).

Networks

- There aren't enough IP addresses for all hosts, and also for security reasons it makes sense to have internal networks.
- The IP addresses 10.x.x.x and 192.168.x.x are reserved for *local* networks.
- The router (here 192.168.1.1) is the gateway for all traffic from the LAN (local area network) and the Internet, and has another *external* Internet IP address.

Networks

- When you want to send a packet to *128.253.161.221*, the packet first goes to the router at 192.168.1.1, which then sends it out using its external IP, and so forth.
- When a packet arrives back to the router, it recognizes that really the packet was supposed to go to 192.168.1.104 and so it gets forwarded to me.
- To see the routing path of a packet, try e.g. `traceroute www.cornell.edu`.

Ports

- To differentiate between services like HTTP, HTTPS, DNS, IRC, POP3, MSN, etc. the packets carry a particular *port* number.
- For instance, if google.com receives a packet for port 80, the default port for HTTP, they will answer and treat it like someone opening a homepage.

Ports

- If the packet is for port 443, it is HTTPS, encrypted HTTP traffic, and gets handled by a different application.
- If the packet is e.g. for port 31337, and no application is listening for that port, the packet gets discarded.

Ports

- Services can *bind* to particular ports.
- For instance, if you set up an Apache web-server, it is a continuously running background process (a *daemon*) that is bound on port 80 and awaits connection.

Ports

- There are many programs that allow us to connect to particular ports on a given host.
- Use `telnet www.google.com 80` and try typing something.
- Now try port 81. It should hang since the packet got dropped.
- '*Connection refused*' means that the host explicitly told you that no service was listening on that port.

Telnet

- One of the first protocols that was implemented was *telnet*
- It allows users to authenticate themselves on other machines and spawn a shell if they have access.
- For security reasons telnet connections are rarely used today since passwords are easy to sniff and sessions easy to hijack.

SSH

- The protocol that's most common today is SSH, secure shell.
- It has the same general interface as telnet, but is guaranteed to encrypt the session in the same fashion as HTTPS web traffic is secured.
- You type `ssh user@host` to start a connection.

Security

- Packets can be forged or *spoofed*. There is no guarantee that the source IP address is the person who sent it.
- TCP comes with a feature (sequence numbers) that makes spoofing infeasible since you have to guess a random number.
- The UDP (Unreliable Datagram Protocol) protocol, which has no sessions, and is commonly used for games and other high throughput things can be easily spoofed.

Security

- The services running on every open port need to be secure.
- A whole lot of services have been *exploitable*, meaning that specially crafted invalid input can cause the remote user to take control of the service and gain access to the server.
- Many of these have to do with malformed input, triggering *buffer overflows* and *format string bugs* for C services, and injected commands.

Security

- Imagine a service is open on port 5000. It's run by the following shell script. Can you break it?

```
printf "Password: " ; read -s ID
if ! grep "^$ID\$" passwords >/dev/null; then
   echo "Incorrect";exit
fi
#... do authenticated stuff ...
```