Network sockets

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

Goal

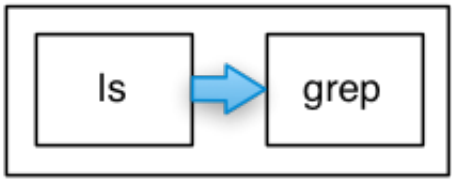
- A program running on a computer can access any of the files on the disk (with sufficient permissions)
- How does the program access data on a different computer?
- We do it all time with the web. A web browser shows a file that actually comes from a different computer
- We need background on networks so we understand how to:
  - pull data from a network
  - create a Web server to provide data services

Questions

- What exactly is the web?
- What are the components?

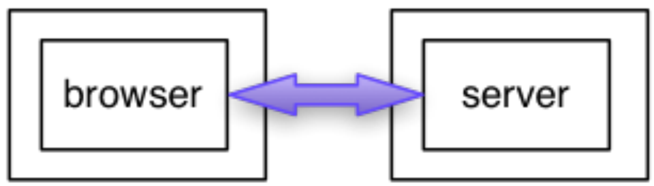
From pipes to sockets

- Separate processes on the same computer may share data and synchronize via *pipes*; For example: **ls | grep Aug**
- That pipes the output of **ls** to the input of **grep** using the UNIX **pipe**() function; sets up a one-way data flow from one process to another



What about connecting processes on separate computers?

- Python provides access to OS *sockets* that allow two or more processes on the same or different computers to send/receive data very much like a 2-way phone connection



- We can treat sockets just like files or any other stream of data from a programming perspective
- Have to keep in mind that it is across a slow link on a network versus in memory on the same machine

Physical layer

- Most of your laptops are connected by radio (WiFi) to a radio transceiver that sends data over a wired connection (Ethernet)
- That transceiver sends data down to a central server room that has a fiber-optic connection to an Internet service provider
- The data from there could go by satellite or fiber-optic cable under the ocean, depending on source and target locations
- You can think of this as the electrical plumbing of the Internet
- This layer we can simply assume exists and knows how to transmit data, albeit at different rates and with different latency

Internet protocol: IP

- The lowest level abstraction above the hardware
- *Please distinguish IP protocol from ethernet, wireless, or any other physical networking mechanism*
- This is a data protocol that sits on top of some physical network

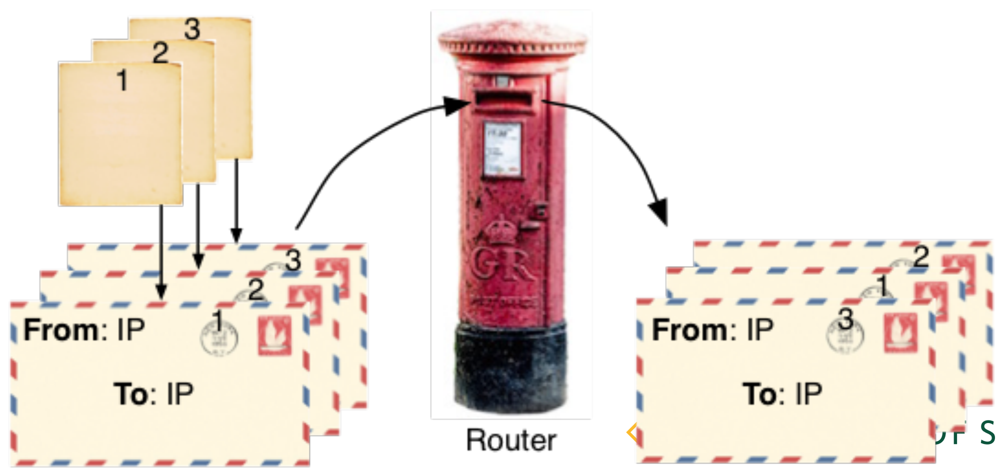
IP is an addressing, fragmentation protocol

- It breaks all communications into *packets*, chunks of data up to 65536 bytes long.
- Packets are individually *routed* from source to destination.
- IP is allowed to drop packets; i.e., it is an unreliable protocol.
- There are no acknowledgements and no retransmissions.
- There is no flow-control saying "*You're sending data too fast!*"

IP is fire-and-forget like a postal letter

- You write a mutli-page letter (this is the data you are sending)
- Put each page of the letter inside a different envelope (the IP packet)
- Address the envelope (using an IP address)
- Put your return address on the envelope (your local IP address)
- Send the letter

We have no way of knowing whether an IP packet was received. If we send a second letter one day after the first, the second one may be received before the first.



IP Addresses

- IP uses *IP addresses* to define source/target
- IPs are 32 bit numbers represented as four 8-bit numbers separated by periods, such as 172.16.198.184
- When you try to visit www.cnn.com in your browser, the computer must first translate www.cnn.com to an IP address
- Then the browser can make a connection to the web server on the target machine identified by the IP address
- You can think of this as the "phone number" of a machine

Special addresses

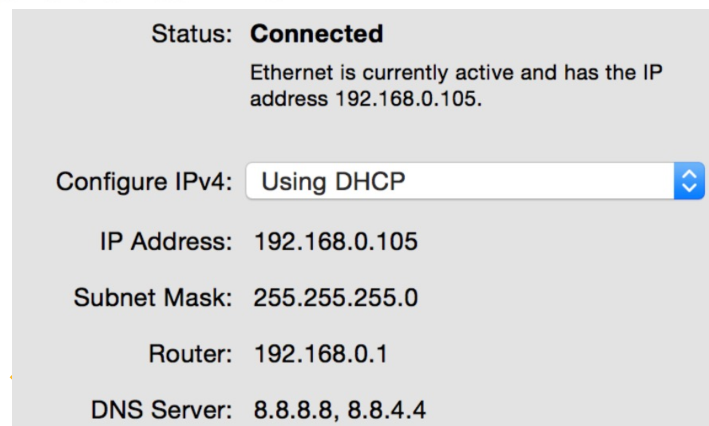
- Behind firewalls, people often use 192.168.x.y and use NAT (*network address translation*) in their firewall to translate an outside address (a real IP) to the special IPs behind the wall
- In this case there is an external or public IP address and a private IP address
- My varmint.cs.usfca.edu machine has public IP 138.202.170.154 but internal IP 10.10.10.51
- 127.0.0.1 is a way to say localhost or "my machine"

Privacy issue

- A good security feature is to hide your machines from outside
- For example, all machines from within IBM's firewall probably look like the exact same IP address to the outside world (such as in web server log files)
- That is one reason you cannot use an IP address to identify users for a web server applicatio.

Exercise: Python code to find IP address

Exercise: Install and use package `netifaces` via `import netifaces as ni` then call `ni.ifaddresses('en0')[ni.AF_INET][0]['addr']` (on linux it might be `eth0` not `en0`; on mac might be en1) (or on el capitan mac and earlier, you can do just `socket.gethostbyname(socket.gethostname())`) to figure out what your IP address is. If this pops up with 127.0.0.1 ("localhost") then you will need to go to your laptop network configurationto find your IP address. Solution



DNS -- Domain Name Service

- DNS is a distributed database that maps domain names to IP addresses using a series of distributed DNS servers
- It is really just a dictionary that maps a name to an IP address
- Here is an example query using a UNIX tool

```
$ nslookup www.usfca.edu
Server:         138.202.170.254
Address:        138.202.170.254#53

Non-authoritative answer:
Name:   www.usfca.edu
Address: 104.239.221.147
```

Exercise

Exercise: use `gethostbyname` from the Python `socket` package to look up `www.usfca.edu`'s IP address. The IP address should be the same you get from the commandline with `nslookup`. It'll be something like `104.239.221.147`. Use `gethostname()` to determine your laptop's hostname. Solution Confirm it's the same as what `hostname` from the commandline prints:

```
$ hostname
beast.local
```

Caching, etc…

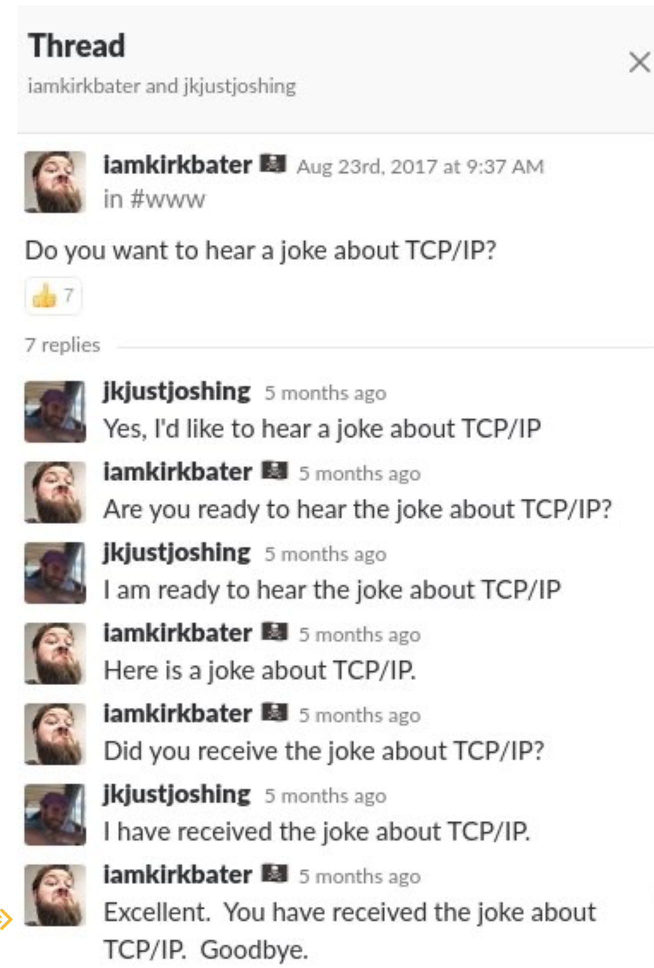
- DNS lookup is distributed so there isn't a single point of failure
- A single server would also get absolutely pounded by requests from the net and would be extremely expensive to maintain
- There are caches etc. that reduce the load on the DNS servers
- If we didn't have DNS, we would all have to memorize a constantly shifting set of IP addresses (as we did before smart phones)

TCP/IP

- TCP (*Transmission Control Protocol*) is another protocol, a reliable but slower one, sitting on top of IP. Believe it or not it comes from the 1970s
- **Reliable** (fault tolerant). TCP automatically deals with lost packets before delivering a complete "file" to a recipient
- **Stream-oriented** connections. We can treat the connection like a stream/file rather than packets
- Packets are **ordered** into the proper sequence at the target machine via use of *sequence numbers*
- **Control-flow** prevents buffer overflows etc...

TCP is like a phone connection

- TCP is like a phone connection versus the simple "fire and forget" letter stateless style of IP
- TCP connections are open for the duration of a communication (i.e., until you close the connection)



Sockets

- If the IP address is like an office building's main phone number, socket numbers are like the extension numbers for offices and are often called the *port*
- *T*the IP address and socket number uniquely identify an "office" (server process)
- You will see unique identifiers like 192.168.2.100:80 where 80 is the port
- We open sockets to ports in order to communicate with servers

Special port numbers

- Ports range from 1..65535
- Ports 1..255 are reserved for common, publicly-defined server ports like:
  - 80: HTTP (web)
  - 110: POP (mail)
  - 25: SMTP (mail)
  - 22: SSH (remote shell connections)
- Ports 1..1024 require root/superuser privileges to use

Connected to a port from commandline

- You can use telnet (putty on windows) to connect to remote ports to manually speak the protocol
- The most successful and long-lived protocols are simple and text based
- For example, here is how I connect to port 80, the Web server, at the University:

```
$ telnet www.usfca.edu 80
Trying 104.239.221.147...
Connected to www.usfca.edu.
Escape character is '^]'.
```

To escape/quit, use *control-]* and then quit.

Nobody’s home

- Just like in an office, it is possible that no process is listening at a port; i.e., there is no server waiting for requests at that port

```
$ telnet www.usfca.edu 81
Trying 104.239.221.147...
telnet: connect to address 104.239.221.147: Connection refused
telnet: Unable to connect to remote host
```

Exercise

- First, **brew install telnet** then use the **telnet** program from the commandline to connect to the following ports:
  - www.usfca.edu 80
  - www.cnn.com 80

Ya gotta speak the right language

- Just because you can open a connection to a port doesn't mean you can speak the right language
- Processes at ports all speak a specific, predefined, agreed-upon protocol like HTTP
- To effectively communicate you need to know both the address and the protocol
- **Exercise**: Use **telnet** to open a socket to **www.usfca.edu:80** and type **hi** or some other text after connecting. The server should respond with *Client sent a bad request*

Sending mail the hard way

- To send a piece of email, you need a mail client (even if it's telnet) that connects to an *SMTP* (Simple Mail Transfer Protocol by Jonathan B. Postel, 1982) and provides a packet of email with a target email address user@domain.com

```
~/tmp $ telnet smtp.usfca.edu 25
Trying 138.202.192.18...
Connected to smtp.usfca.edu.
Escape character is '^]'.
220 smtp.usfca.edu ESMTP Postfix
...
```

Sample SMTP

- You have to type lines marked with **<--**

```
$ telnet smtp.usfca.edu 25
Trying 138.202.192.18...
Connected to smtp.usfca.edu.
Escape character is '^]'.
220 smtp.usfca.edu ESMTP Postfix
HELO cs.usfca.edu                                 <--
250 smtp.usfca.edu
MAIL FROM: <parrt@cs.usfca.edu>                   <--
250 Ok
RCPT TO: <support@antlr.org>                      <--
250 Ok
DATA                                              <--
354 End data with <CR><LF>.<CR><LF>
From: parrt@cs.usfca.edu                          <--  Needs this header
                                                  <--
This is a test                                    <--
so nothing really                                 <--
.                                                 <--
250 Ok: queued as 1A0C183F
QUIT                                              <--
221 Bye
Connection closed by foreign host.
```