Welcome to CS 168

•••

Intro to the Internet

Agenda

- Introductions
- Poking the Internet
- Sockets
    - Establishing connections
    - Send and receive

Introductions

Poking the Internet

Ping, Traceroute, Dig

- These tools can give some insight into what is going on inside and across the internet.
- We’re going to play around with them a little bit

Ping, Traceroute, Dig

- Ping is a simple utility that basically lets you poke a website and see if it moves (spoiler: most do!)
- You say hi and see if the server says hi back
  - This by itself is not super interesting
- Ping also tells you how long the reply took to come back
  - This is more interesting!
- Let’s try out a few websites.

Ping: A prediction

- We've pinged a couple websites and seen pretty significant differences in *latency*.
    - Latency is the time between when a request is sent and when the response is heard.
- What about differences in latency for the same website, but in different regions?
- We've pinged google.com and seen its latency.
    - How many times longer will it take for a ping to google.co.uk to come back?

Ping, Traceroute, Dig

- Does what its name implies: it allows you to trace the route that packets take from your computer to the destination.
  - Specifically lets you see the routers/switches that are forwarding your packets.
- Packets have a limit to how many times they can be forwarded
  - Dropped when the limit is exceeded
    - Prevents a packet from looping and other issues. We will come back to this later.
  - Most routers will notify the sender when the packet is dropped
- Traceroute sets the limit to 1,2,3,… for the same destination
  - Receive drop notifications from each router
  - Allows sender to trace the path of the packet
- Demo

Traceroute: Notice anything?

- Traceroute gives us a lot more interesting feedback than ping.
    - Latency to *every* step along the way.
        - Can see a breakdown of latencies!
    - Router names.
        - Often have locations in them (i.e. city name)
        - Can roughly trace packet path on a map!
    - Multiple routers at a given step!?
        - Traffic engineering, load balancing, route changes!
    - Weird stars
        - Some routers just don’t respond ¯_(ツ)_/¯

Ping, Traceroute, **Dig**

- When humans want to go to a website, we think in terms of names
    - i.e. google.com
- The internet does not think this way, it thinks in terms of *addresses*
    - i.e. “1.2.3.4”
- It’s like the postal service
    - You wouldn’t just write “To: Alice” on a letter
    - You would look up Alice’s address in some directory
        - Then mail the letter to her address
- Dig lets you lookup the address of a website by its name
    - Command line interface to the Domain Name Service (DNS)
- Demo

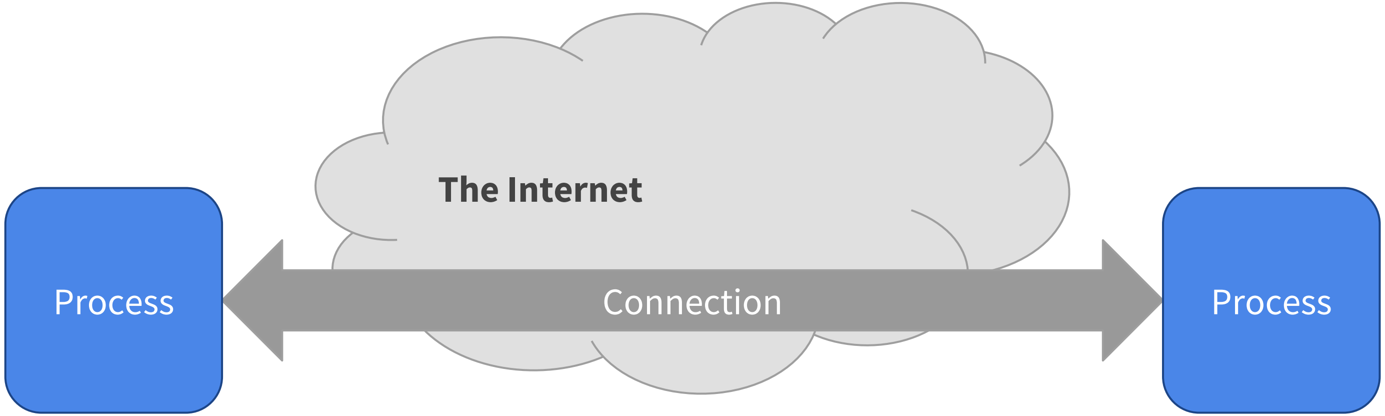
Sockets

Sockets

- The Internet’s user API
- Developed here, at UC Berkeley!

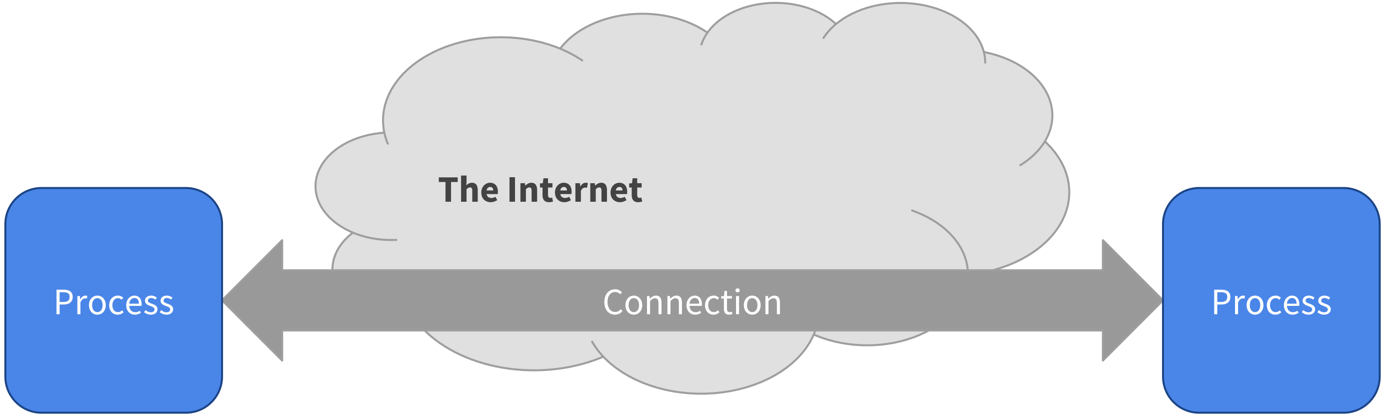
Connection (the basic abstraction)

- Think of this as a simple pipe between two processes
  - A process is just a program running on a host
- Data goes in one end, and comes out the other
- Data flows both ways!



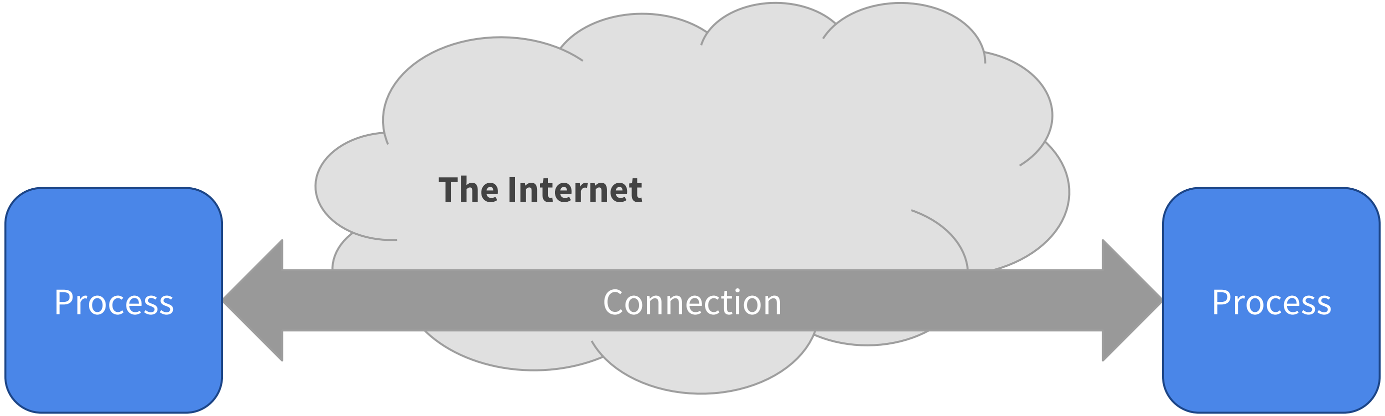
Connection (the basic abstraction)

- Data is sent simply as a stream of bits
- Reconstruction of what the bits mean done entirely at the endpoints
- Means the Internet knows nothing about what it's transmitting!



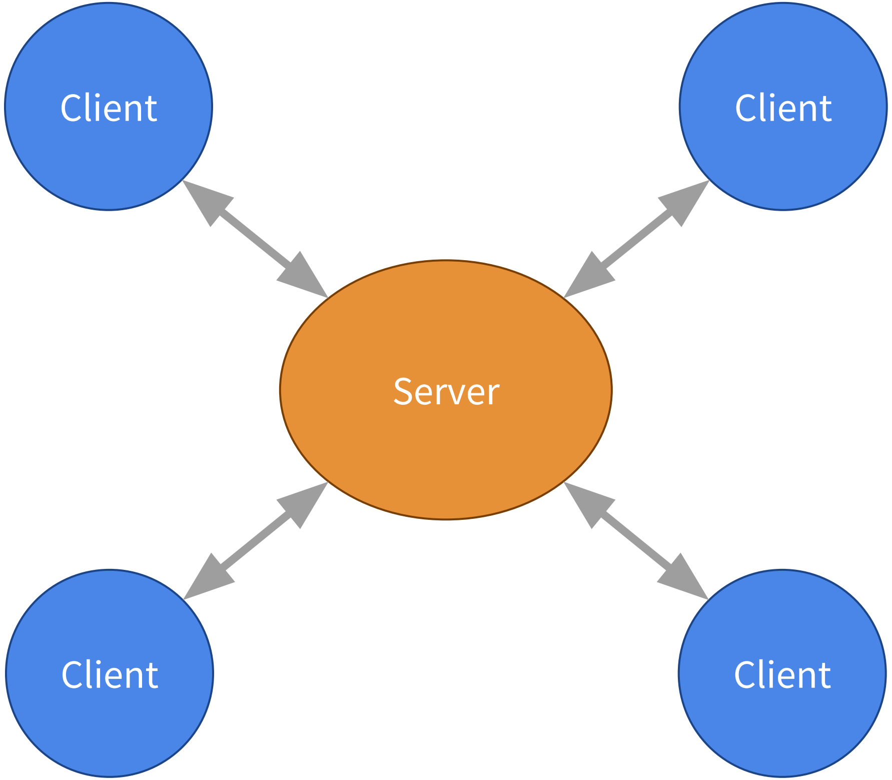
Socket API

- Establish Connection
- Sending
- Receiving



Connections

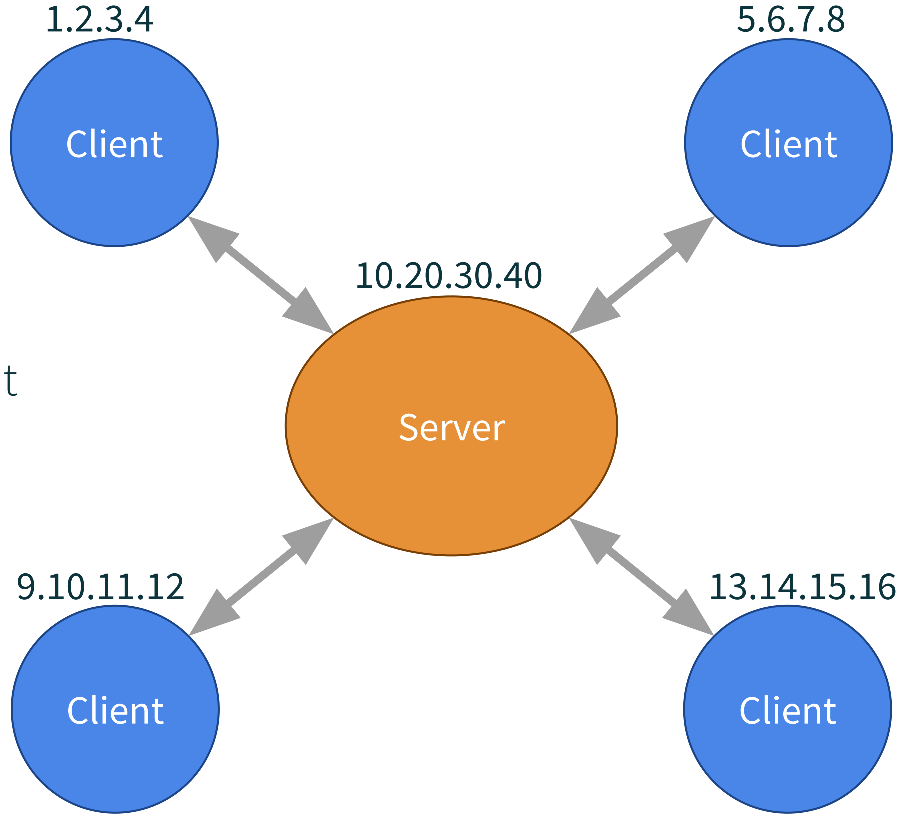
- Two types of sockets
  - Server and Client
- Servers *listen* for clients to connect to them
  - Wait until a connection is attempted
    - Accept and dispatch connection
  - Usually serving many clients at once
- Clients *initiate* new connections to servers
- Example
  - Server: berkeley.edu
  - Client: Your internet browser



Connections

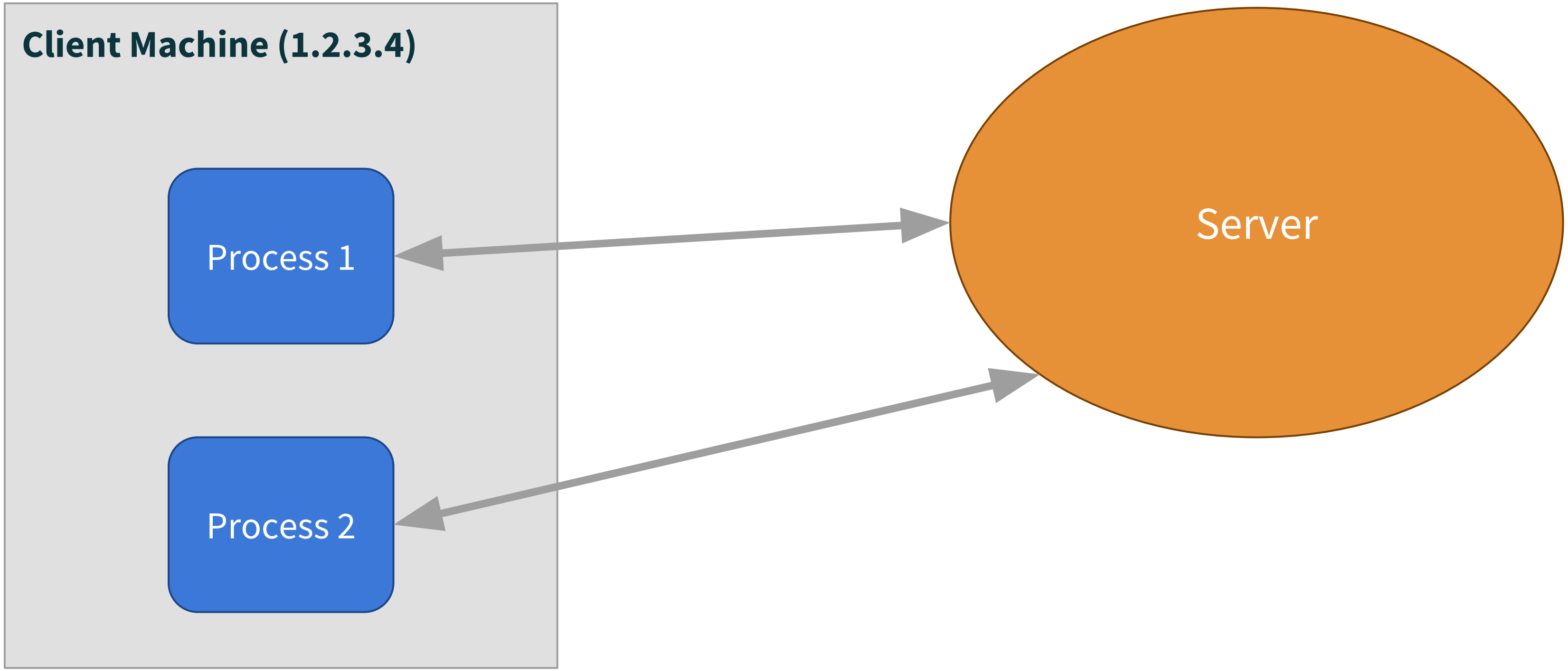
- Hosts have addresses
  - Unique identifier (just like a street address)
- Clients find servers with their addresses
  - Servers send data back with the client address
- Example addresses ➔

Are addresses enough to make this work?



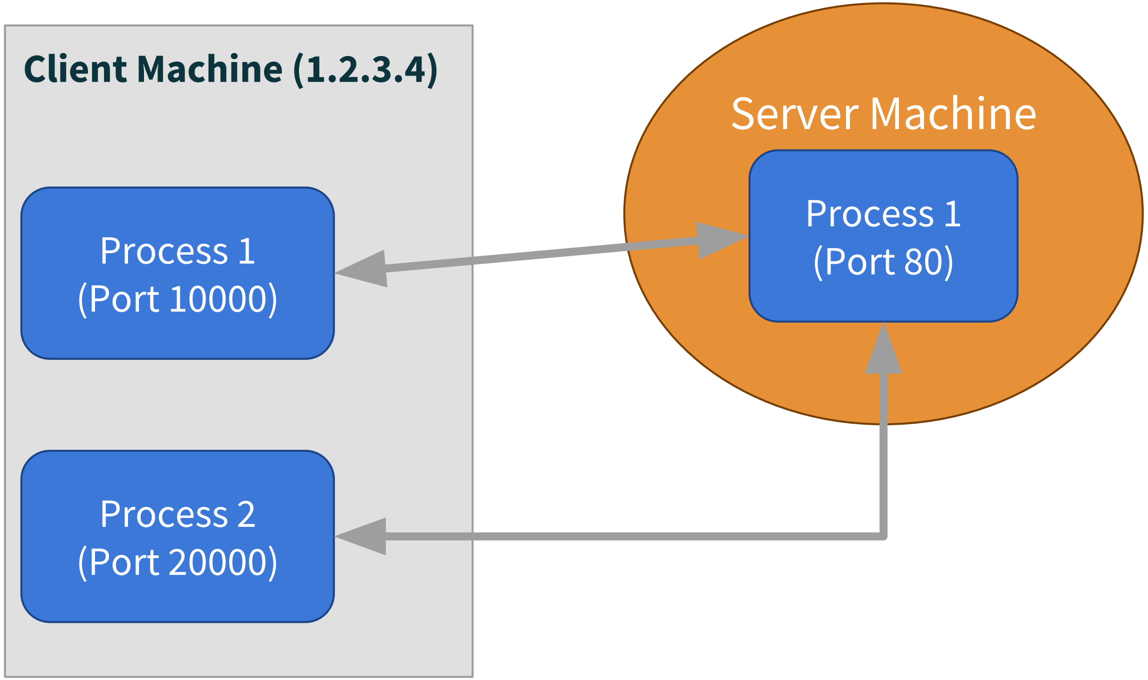
Address aren’t enough

How does the client host know which process to deliver data to?



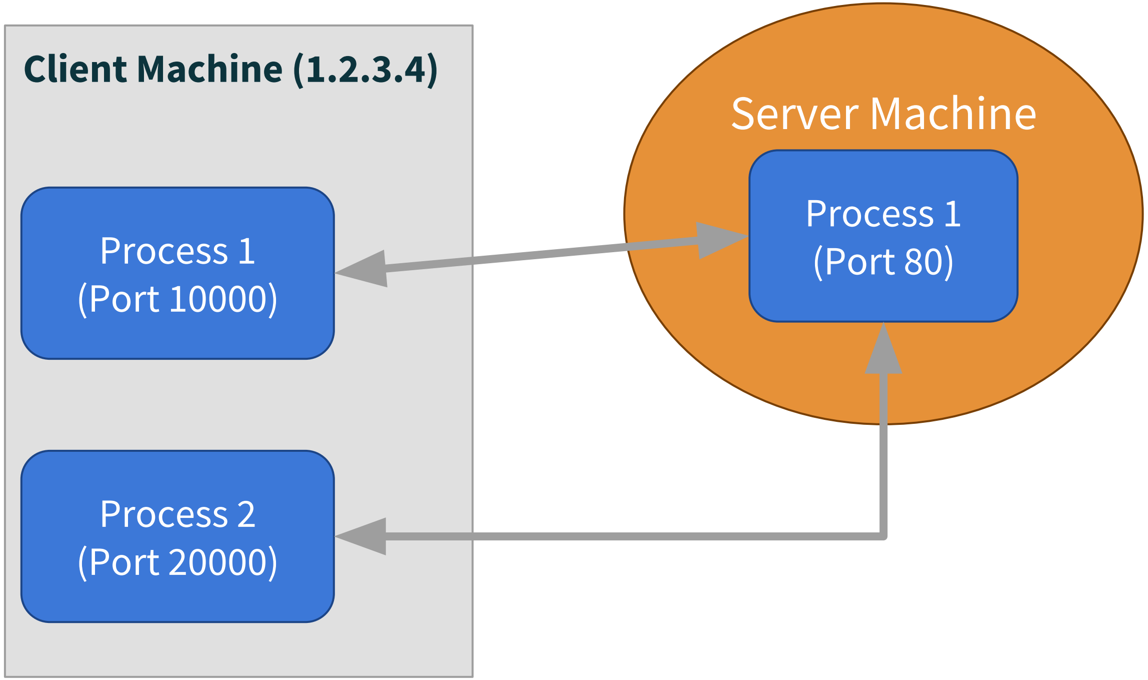
Ports

- Sockets are identified by unique IP:port pairs
- A port is a number that the OS associates with a process when it is created
    - Used in the address to tell which port socket is listening on
    - i.e. sending to address “1.2.3.4:10000” would send data to the socket owned by Process 1



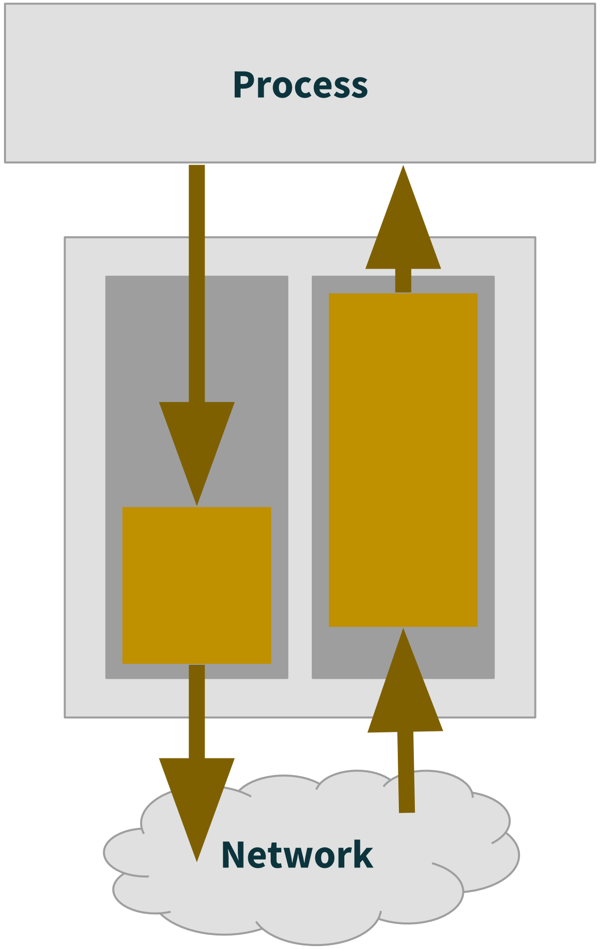
Ports

- Packets carry port number
- Servers listen on a port
  - Which one depends on application
  - HTTP: 80
  - SSH: 22
- Client process connects to well known port
- Client also has a port
  - Randomly assigned by OS
  - Used by OS to send data to correct process



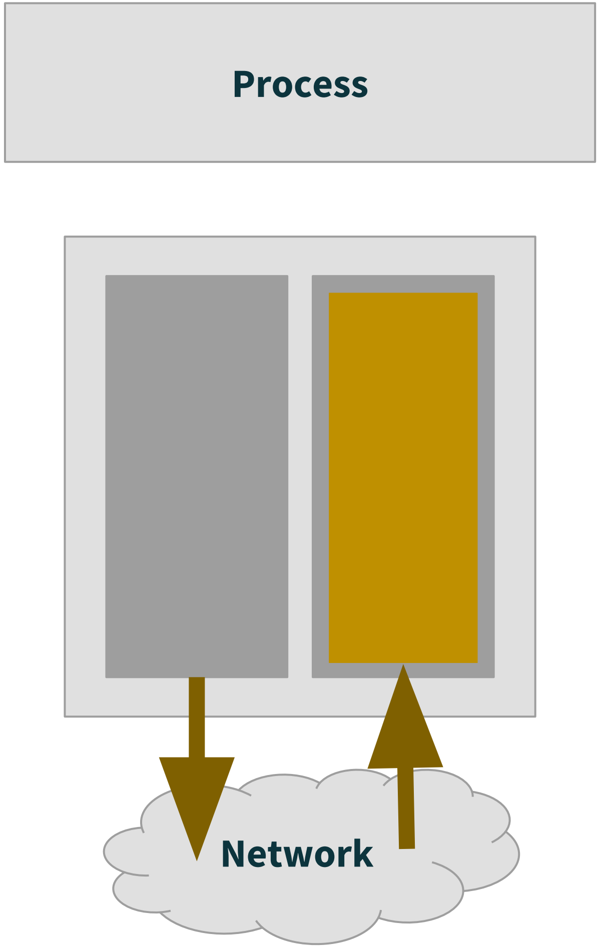
Socket Mechanics

- Send buffer
  - Filled by process
  - Drained by network
  - Bits wait to be transmitted by network
- Receive buffer
  - Filled by network
  - Drained by process
  - Bits wait to be read by process
- Why two buffers?



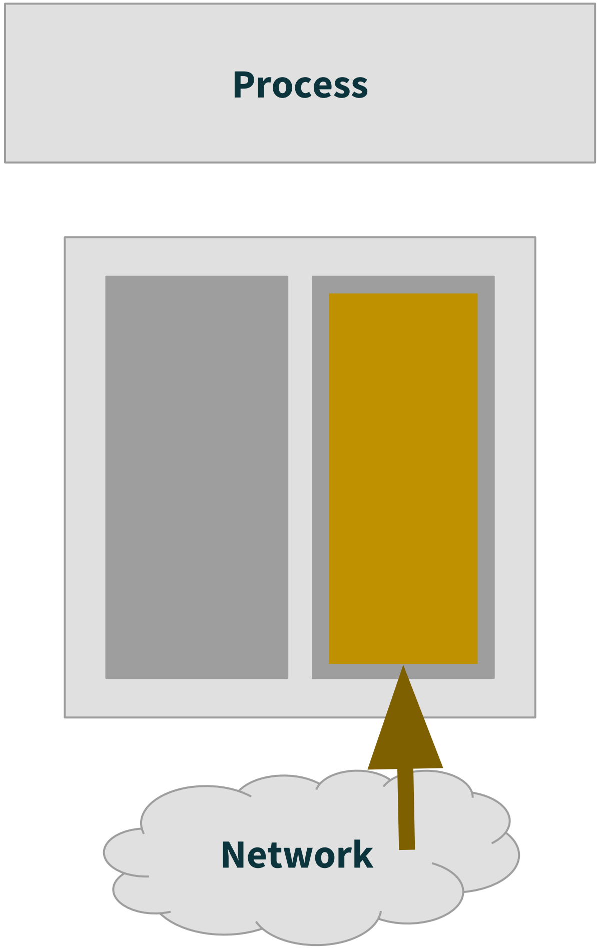
Socket Mechanics: Full or Empty

- What if you write to a full socket buffer or read from an empty one?
- Two solutions:
  - Blocking (wait)
  - Non-blocking (return error)
- We'll talk about blocking briefly.



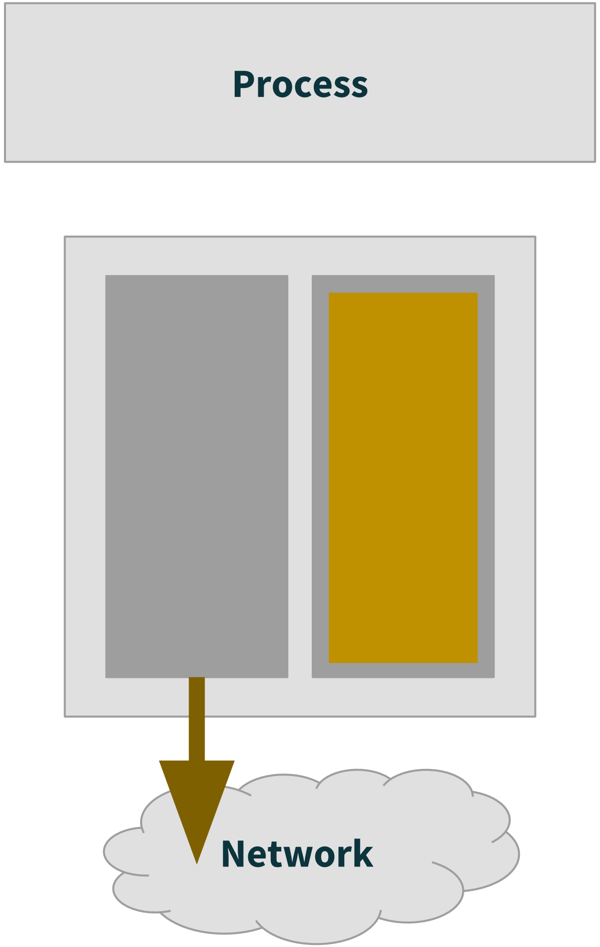
Socket Mechanics: Blocking Write

- Try to write, but buffer is full
- Block (wait) until there is enough room in the buffer
- Write the data and return



Socket Mechanics: Blocking Read

- Try to read, but buffer is empty
- Block (wait) until some bits appear in the buffer
- Read the bits and return them



Socket Mechanics: Non Blocking

- Non blocking reads and writes behave differently
- If the buffer isn’t ready for the operation
    - Just return an error
- The user must try again later

Discussion: Which is better, blocking or non-blocking?

Sockets: Python API

https://docs.python.org/2/howto/sockets.html

Connection Establishment: Server

- Create a socket:
  - `server_socket = socket.socket()`
- Bind the socket to listen on an IP:port address
  - `server_socket.bind((“0.0.0.0”, 80))`
  - Usually use “127.0.0.1” (“loopback”: only on local machine) or “0.0.0.0” (all host addresses)
- Listen for new connections
  - `server_socket.listen(5)`; up to 5 connections may wait in queue
- Process incoming connection request
  - `(new_socket, address) = server_socket.accept()`
  - `new_socket` now can send to and receive from the client

Connection Establishment: Client

- Create a socket:
  - `client_socket = socket.socket()`
- Connect to server IP “1.2.3.4” on port 80
  - `client_socket.connect((“1.2.3.4”, 80))`
- Socket is now ready to send and receive.
- You can find more socket methods here: https://docs.python.org/2/library/socket.html

Send and Receive

- `send(str)`
  - Take some bits in as an argument, and add them to the send buffer
- `recv(size)`
  - Read at most size bytes from the receive buffer

Bonus Slides

Ping, **Traceroute**, Dig

- Traceroute gives you the path of routers and switches your packets take.
- How?
  - Takes advantage of something called a **TTL** in the packet IP header.
  - TTL denotes how many times a packet should be forwarded before it is discarded.
    - Why does this exist?
      - To stop the internet from collapsing! (We’ll cover this when we get to routing)
  - Sets the TTL to 1, 2, 3, etc
  - When packets are dropped because of TTL expiring, most routers send back a message telling us.
  - Use the source of this notification to identify the routers along the packet’s path.

Old slides

Dig: A breakdown

- When using the +trace option, there was a lot more information
- We could see the steps that were taken when resolving the names
  - First, the ‘root’ servers were queried
  - Then, the TLD (top level domain) server was queried
  - After that, successive servers were asked until the IP was found
- More on how this works when we discuss DNS