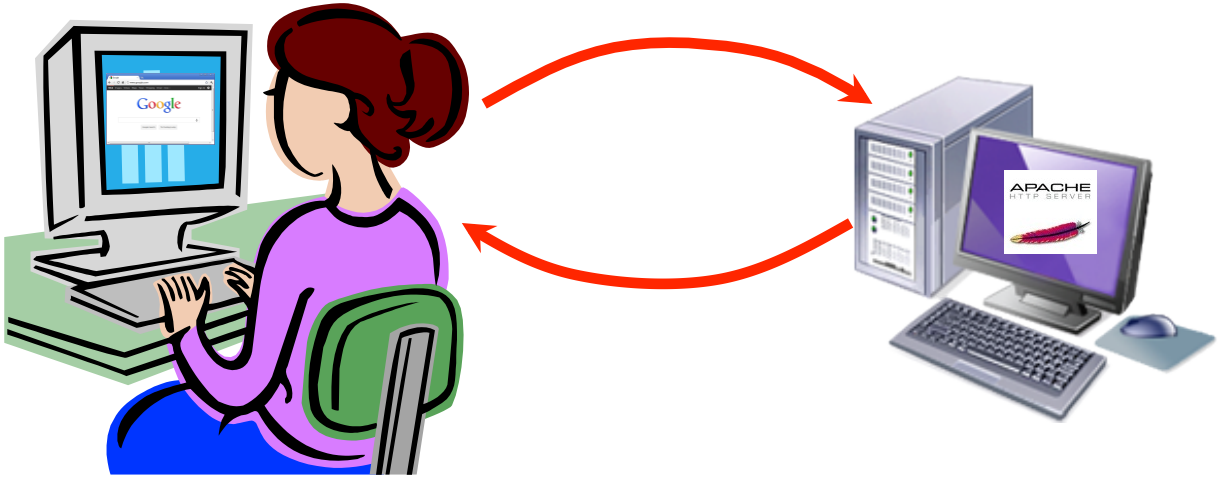
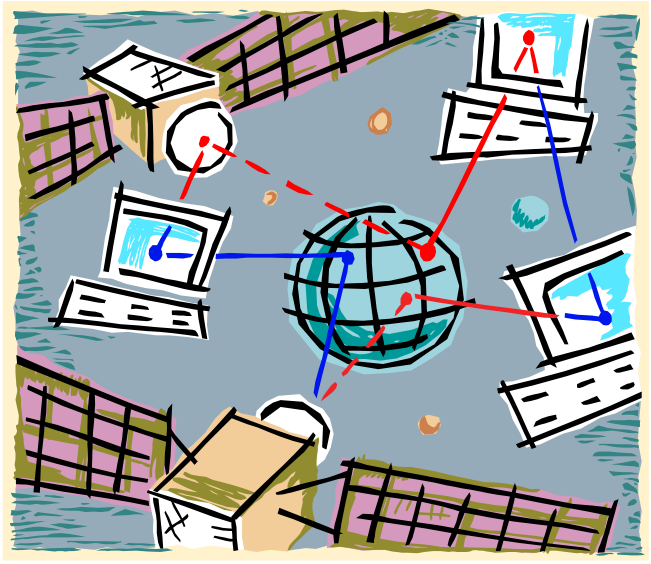
Networking and socket communication



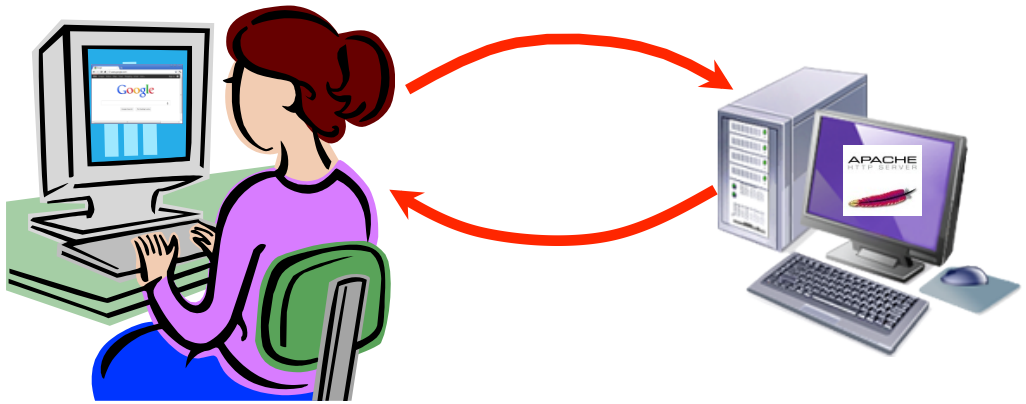


Overview

- Networking basics
  - Different between clients and servers
  - Addressing
    - IP addresses, hostnames, DNS
    - Private addresses, localhost
  - Port numbers
- Socket communication
  - Byte-level communication between two hosts on the network
  - Java client: reading /writing text
  - Java server: accepting clients, reading/writing text

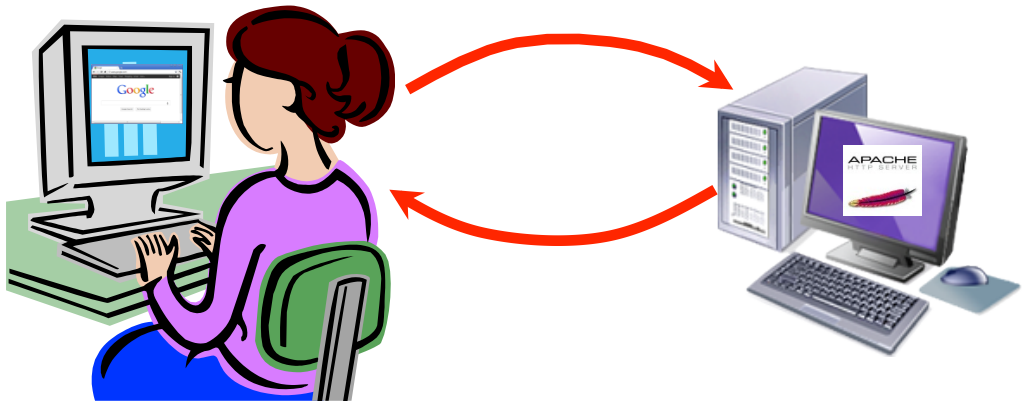
Clients and servers

- Client program
  - Requests service
  - E.g. web browser, audio player, Twitter client
- Server program
  - Provides service
  - E.g. web server, audio server at streaming station, server at Twitter



Clients and servers

- Client program
  - "sometimes on"
  - Doesn't talk to other clients
  - Needs to know server's address
- Server program
  - "always on"
  - Serves requests from many clients
  - Needs fixed address



Communication components

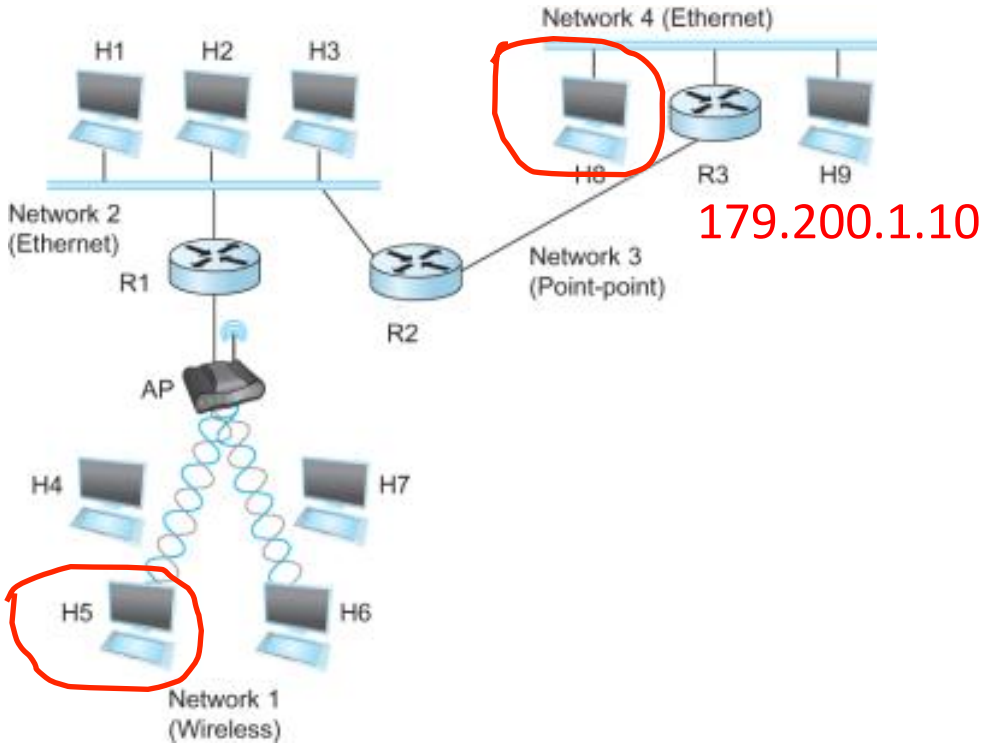
- Network
  - Gets data to the destination host
  - Uses destination IP address
- Operating system
  - Forwards data to a given "silo" based on port #
  - E.g. All port 80 request go the web server
- Application
  - Actually reads and writes to socket
  - Implement the application specific magic
    - e.g. Implementing a mail reading/writing protocol
    - e.g. Implementing a file retrieval (FTP) protocol

Naming computers

- Goal: Establish communication between A to B
  - How do computer A and B refer to each other?
  - The network needs an addressing system

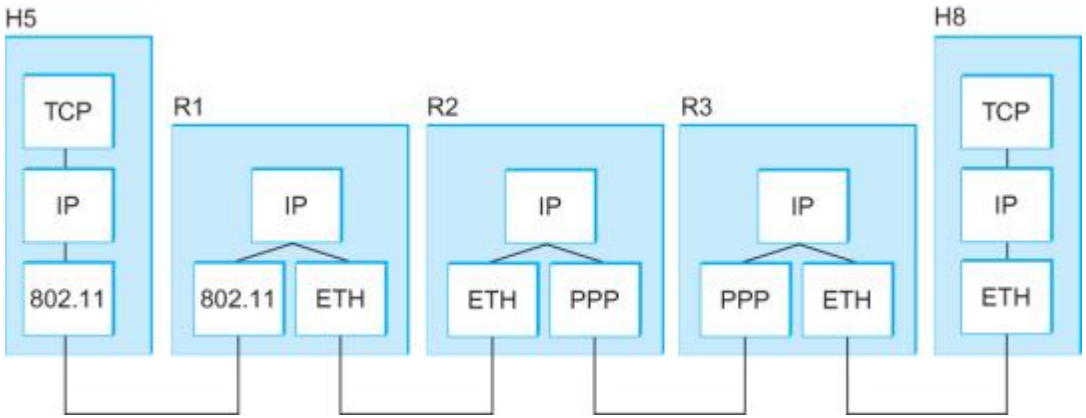
- IP (Internet Protocol) address
  - IPv4 address, 32 bits = about 16 million hosts
  - Usually expressed as four numbers 0-255 (8 bits)
    - e.g. 173.194.79.106
  - IP address uniquely identifies a network endpoint
  - Devices in the network (e.g. switches, routers) use the IP address to get data to its destination

Communication from H5 to H8



179.200.1.10

173.194.79.106



DNS

- Problem 1: Humans can't remember all the numbers in an IP address

- Domain Name System (DNS)
  - Given a readable web site name, translate to an IP address
  - e.g. www.google.com -> 173.194.79.106

http://xkcd.com/302/

Private IP addresses

- Problem 2: IPv4 only has 16 million addresses
  - 7 billion people , all want a laptop, Xbox & iPhone
- Private IP addresses
  - Allow construction of a private network
    - You can route data between any two endpoints on the private network
  - But address isn't valid outside that network
  - 192.168.x.x, 10.x.x.x, 172.16/31.x.x
  - Typically what you'll have:
    - On your home network
    - Desktops at Tech
    - Wireless at Tech
  - 127.0.0.1 (localhost)

http://xkcd.com/742/

Port numbers

- Problem 3: Many applications on the same computer want to talk at the same time
  - Web browser tab 1 wants to talk to google.com/
  - Web browser tab 2 wants to talk to google.com/gmail
  - Web browser tab 3 wants to talk to facebook.com
  - Email client wants to talk to popmail.mtech.edu.

- Solution: Use IP address + port number
  - A 16-bit number, 0 - 65535
  - Port number determines app that traffic is routed to
  - Just a "virtual" port, exists in the operating system

Port numbers

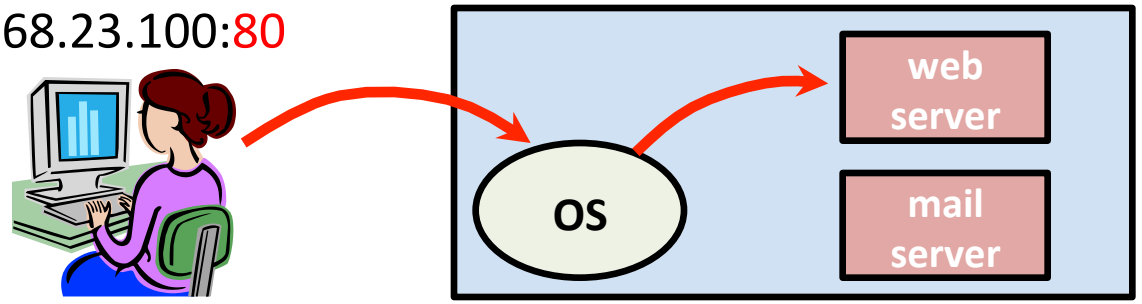
- Popular applications have known ports
  - Ports 0 - 1023: reserved for well-known services
  - Ports 1024 - 65535: available to any user-level application

Port	Service
21	File transfer protocol (FTP)
22	Secure shell (SSH)
23	Telenet
25	Simple mail transfer protocol (SMTP)
53	Domain name system (DNS)
80	Hypertext transfer protocol (HTTP)
110	Post office protocol (POP)
143	Internet message access protocol (IMAP)
443	HTTP secure (HTTPS)

Use of port number

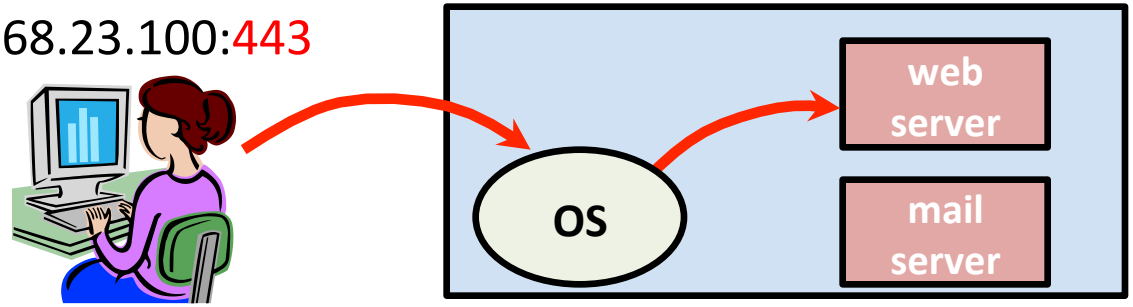
Requesting a non-secure web page

192.168.23.100:80



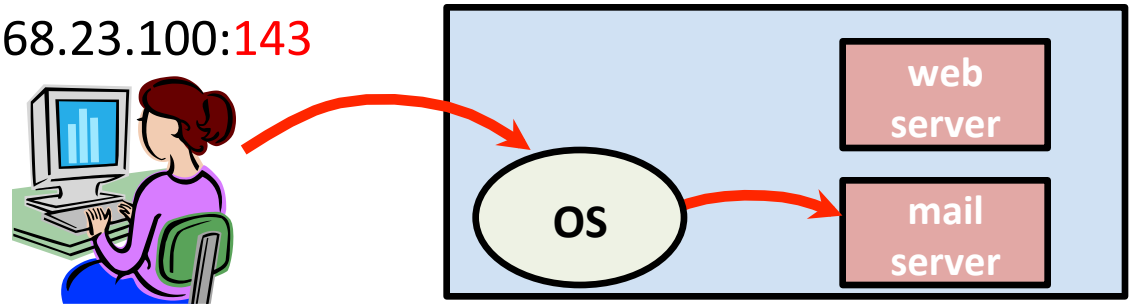
Requesting a secure web page

192.168.23.100:443



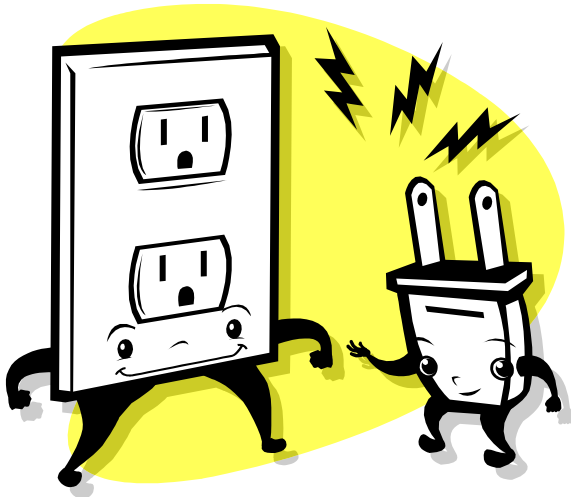
Requesting new email messages

192.168.23.100:143



Sockets

- Socket API (applications programming interface)
  - Allows communication over IP (Internet Protocol)
  - Originally in Berkeley Unix
    - Thus: Berkeley sockets, BSD sockets
  - De facto standard in all operating systems
  - API in many programming languages:
    - C/C++
    - Java
    - C#
    - …



Java client: reading from a socket

- Step 1: Create a new `Socket` object
  - Needs to know the IP address of server + port number

```
Socket socket = new Socket("127.0.0.1", 5000);
```

- Step 2: Create an `InputStreamReader`
  - Convert low-level socket data into a character stream

```
InputStreamReader stream = new InputStreamReader(socket.getInputStream());
```

- Step 3: Create a `BufferedReader`
  - Provides buffered reading from character stream

```
BufferedReader reader = new BufferedReader(stream);
```

- Step 4: Read some text

```
String message = reader.readLine();
```

BufferedReader

Method Summary	
void	close() Close the stream.
void	mark(int readAheadLimit) Mark the present position in the stream.
boolean	markSupported() Tell whether this stream supports the mark() operation, which it does.
int	read() Read a single character.
int	read(char[] cbuf, int off, int len) Read characters into a portion of an array.
String	readLine() Read a line of text.
boolean	ready() Tell whether this stream is ready to be read.
void	reset() Reset the stream to the most recent mark.
long	skip(long n) Skip characters.

Java client: writing to a socket

- Step 1: Create a new `Socket` object
  - Or use an existing one
  - You can combine reads and writes to same socket

```
Socket socket = new Socket("127.0.0.1", 5000);
```

- Step 2: Create an `PrintWriter`
  - Seen previously when writing to a file

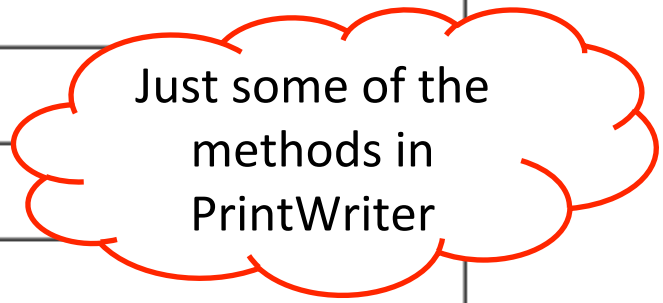
```
PrintWriter writer = new PrintWriter(socket.getOutputStream());
```

- Step 3: Write something

```
writer.println("Hello over there!");
```

PrintWriter

void	print(double d) Prints a double-precision floating-point number.
void	print(float f) Prints a floating-point number.
void	print(int i) Prints an integer.
void	print(long l) Prints a long integer.
void	print(Object obj) Prints an object.
void	print(String s) Prints a string.
PrintWriter	printf(Locale l, String format, Object... args) A convenience method to write a formatted string to this writer using the specified format string and arguments.
PrintWriter	printf(String format, Object... args) A convenience method to write a formatted string to this writer using the specified format string and arguments.
void	println() Terminates the current line by writing the line separator string.
void	println(boolean x) Prints a boolean value and then terminates the line.
void	println(char x) Prints a character and then terminates the line.
void	println(char[] x) Prints an array of characters and then terminates the line.



Java socket server

- Client needs somebody to talk to!
- Server slightly different than Client:
  - Must be running before client tries to connect
  - Server decides what port number to listen on
    - But doesn't specify IP address
    - Doesn't know who is going to connect
  - Blocks waiting to "accept" a client connection
  - Then reading/writing similar to client

Java socket server

- Step 1: Create a `ServerSocket` object
  - Declares what port you are listening on
  - Nobody else on the computer better be using it!

```
ServerSocket serverSock = new ServerSocket(5000);
```

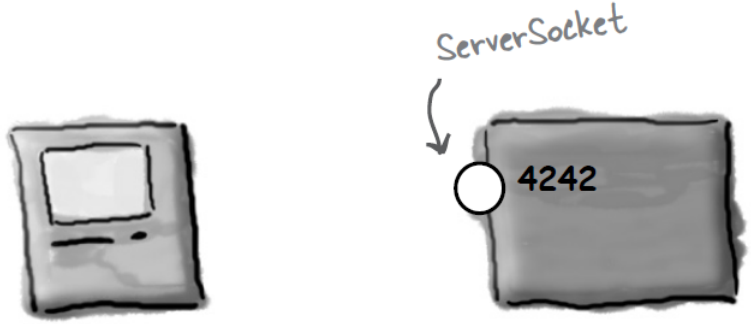
- Step 2: Wait for a client to connect
  - `accept()` method blocks until client arrives

```
Socket sock = serverSock.accept();
```

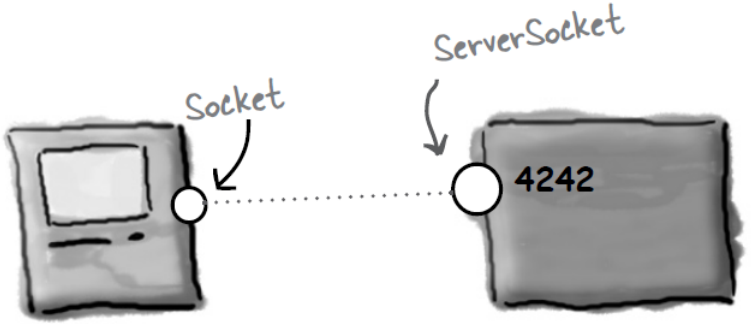
- Step 3: Read/write same way as a client
  - Create BufferedReader for reading strings
  - Create PrintWriter for writing strings

Connection process

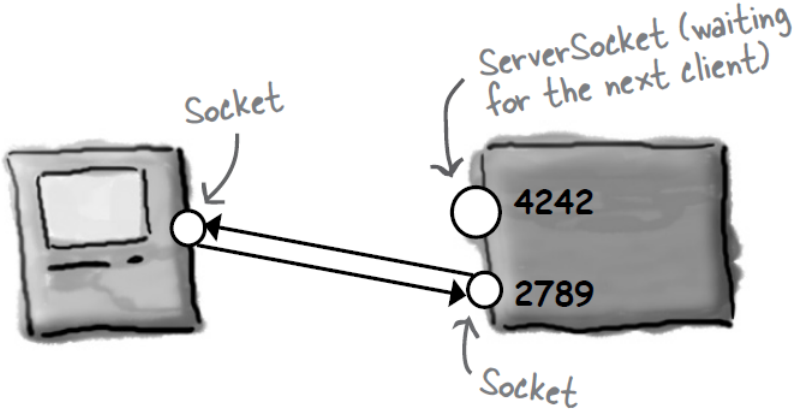
1. Server program starts up.
2. Starts listening on port 4242.
3. OS sends all inbound connection requests to 4242 to the server program.



4. Client program starts up
5. Requests connection to server IP address on port 4242.



6. Server establishes a socket connection to client, but on different port (2789)
7. Server can listen for new clients on the 4242 port number.



Magic 8 ball: Internet Edition

- Server:
  - My laptop on the wireless network
  - Private IP address
  - Running on port 5000
  - Delivering 1 of 20 messages
- Client(s):
  - My laptop in a different shell window
  - Your laptop on the wireless network
  - Private IP address
  - Displays message from the server

Summary

- Basics of networking
  - Computer all have a numeric IP address
  - Some computers have a friendly name (e.g. google.com)
  - Port numbers identify which program to send request to
- Java socket communication
  - Clients create: `Socket` object
  - Servers create: `ServerSocket`, then a `Socket` per client
  - Reading via `BufferedReader`
  - Writing via `PrintWriter`