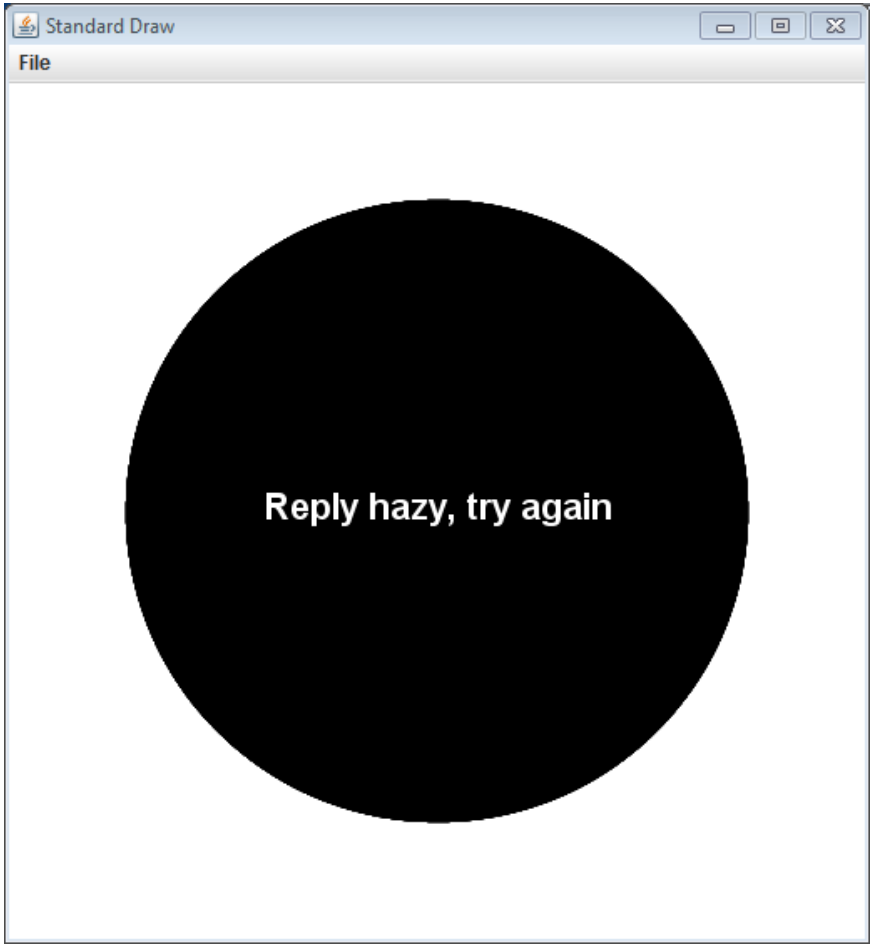
Socket clients and servers



Overview

- Networking basics
  - IP Addresses (review)
  - Port numbers (review)
  - Reliability, connecting, latency, firewalls
- Single threaded examples
  - Magic-8 ball
  - Magic-8 ball persistent
- Multi-threaded servers
  - Magic-8 ball multi-threaded server
  - Shared key/value server

Client1 @ 192.168.1.100

```
% java Magic8Client
150.131.202.152 5000
```

Server @ 150.131.202.152
katie.mtech.edu

```
% java Magic8Server
5000
```

Client2 @ 192.168.1.110

```
% java ValueClient
katie.mtech.edu 6000
```

Client3 @ 192.168.1.120

```
% java ValueClient
150.131.202.152 6000
```

```
% java ValueServer 6000
```

Thread 1

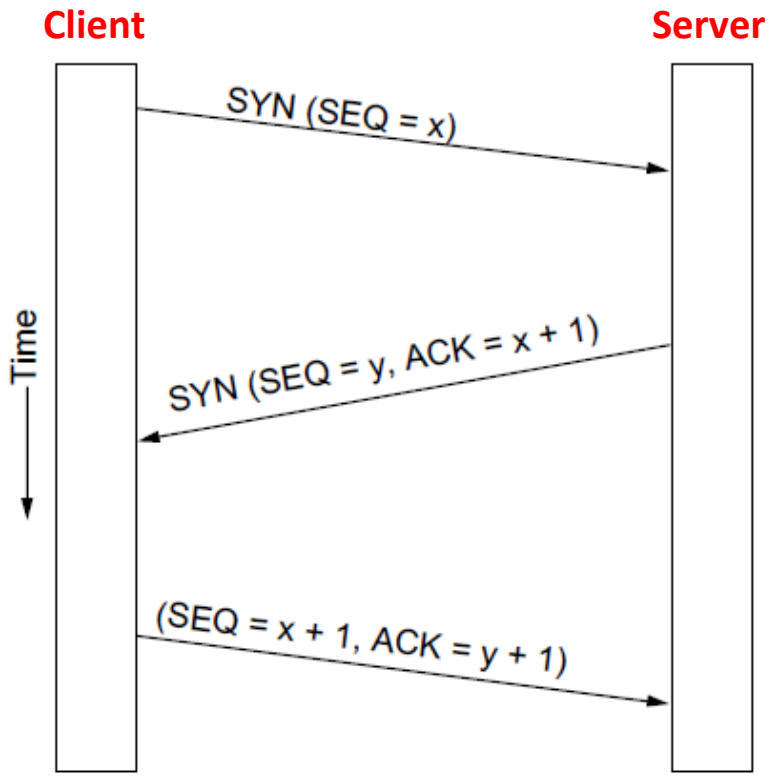
Thread 2

Reliability

- Socket communication
  - We'll be using TCP (Transmission Control Protocol)
  - TCP/IP, TCP over IP (Internet Protocol)
  - IP is the de facto standard for messages on the Internet
  - IP is "best-effort" delivery
    - Messages may or may not get there
    - Messages may get reordered in transit
  - Lucky for us, TCP provides reliable and in-order delivery
    - You can be sure what you read/write will get there

Establishing a connection

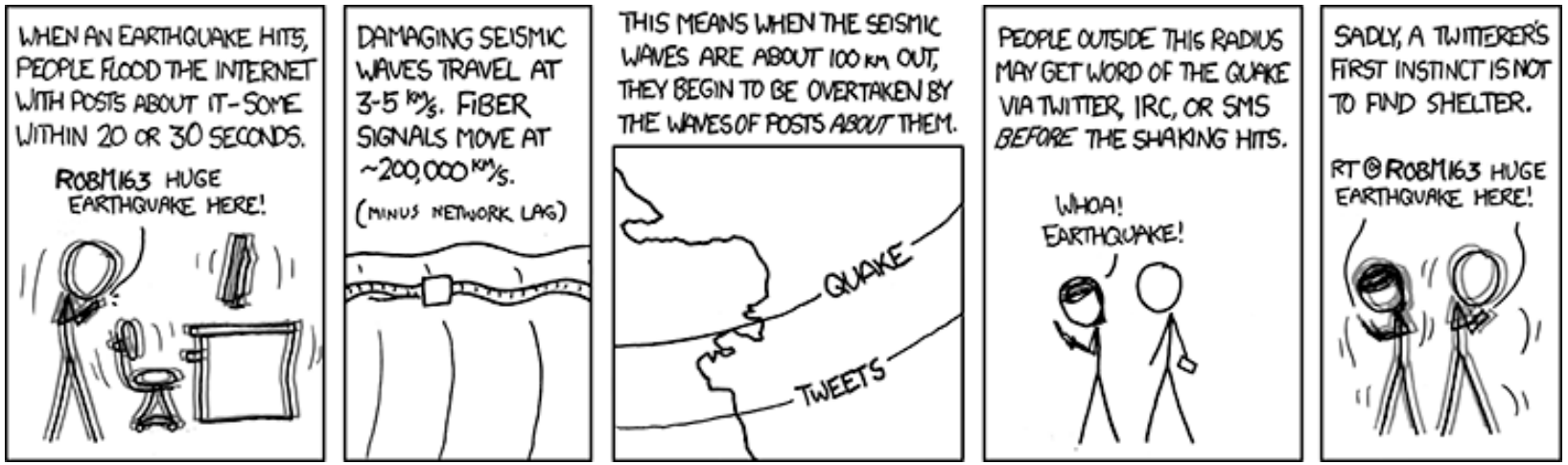
- Starting a socket connection:
  - 3 way handshake
  - Connection takes a bit to startup
  - Keep around if you have an ongoing conversation



Latency

- Signals only go so fast

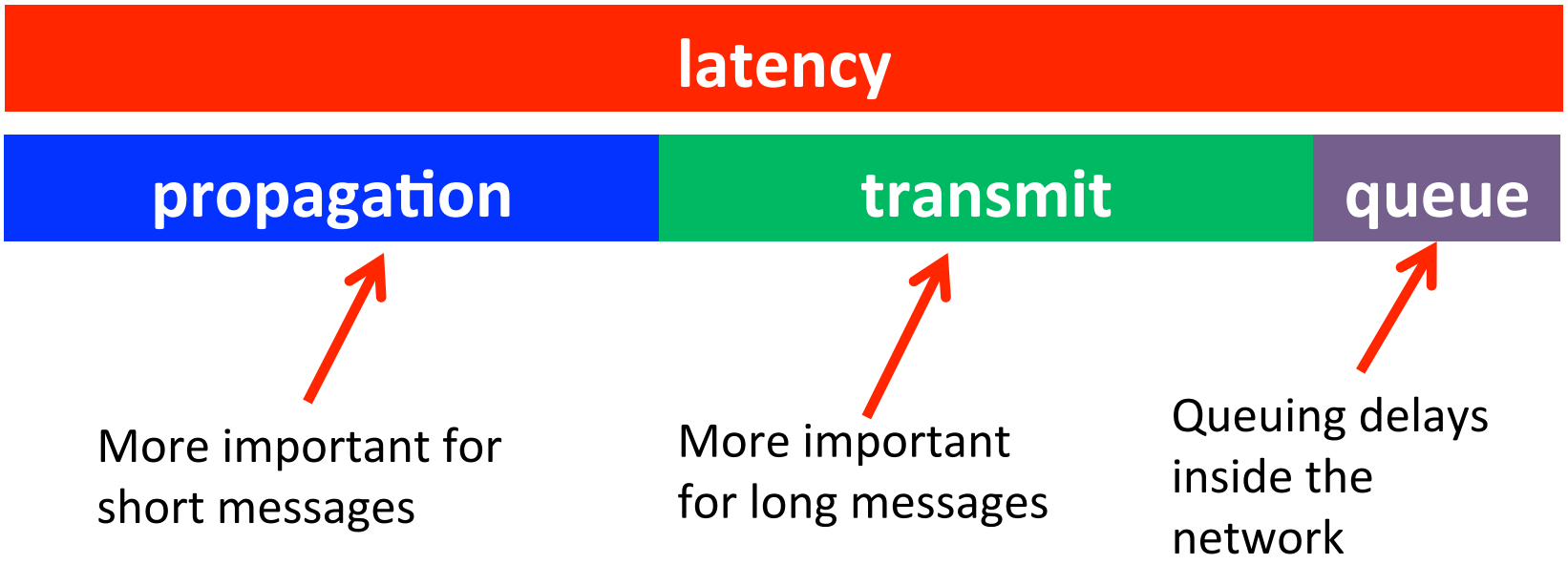
Medium	Speed of light
Vacuum	3.0 x 10^8 m/s
Copper cable	2.3 x 10^8 m/s
Optical fiber	2.0 x 10^8 m/s



http://xkcd.com/723/

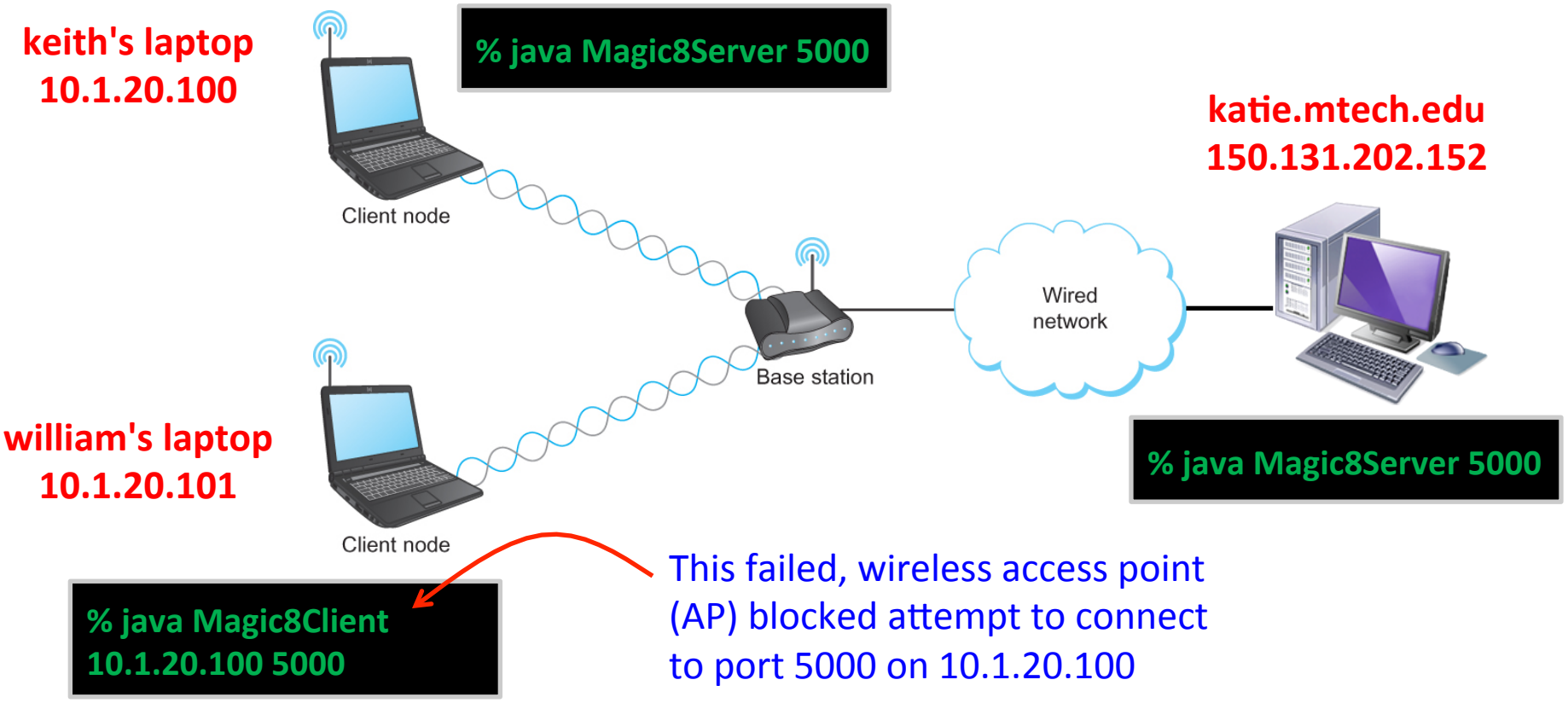
Latency

- latency = propagation + transmit + queue
- propagation = distance / speed of light
- transmit = size / bandwidth



Firewalls

- Network device/software may thwart connection
  - e.g. On the same network but device/software blocks traffic on certain port numbers



Handy network utilities

- ping <hostname, IP address>
  - See if you can reach the destination
    - Though some hosts may disable response
  - Time to get a tiny msg there and back (round trip time)

```
% ping keithv.com

Pinging keithv.com [69.164.194.211] with 32 bytes of data:
Reply from 69.164.194.211: bytes=32 time=123ms TTL=44
Reply from 69.164.194.211: bytes=32 time=123ms TTL=44
Reply
Reply

Ping
    P
Appro
    M
```

```
% ping katie.mtech.edu

Pinging katie.mtech.edu [150.131.202.152] with 32 bytes of
data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 150.131.202.152:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Handy network utilities

- ipconfig (Windows), ifconfig (Mac/unix)
  - Find out your wired/wireless IP address

```
c:\ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : passcall
   Link-local IPv6 Address . . . . . : fe80::615f:559:cfb6:8d35%10
   IPv4 Address. . . . . . . . . . . : 192.168.1.6
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Magic 8 ball: Internet Edition

- Server:
  - katie.mtech.edu wired network
  - Public IP address
  - Running on port 5000
  - Delivering 1 of 20 messages
  - Services only a single client at a time
- Client(s):
  - My laptop on the wireless network
  - Your laptop on the wireless network
  - Private IP address
  - Displays message from the server

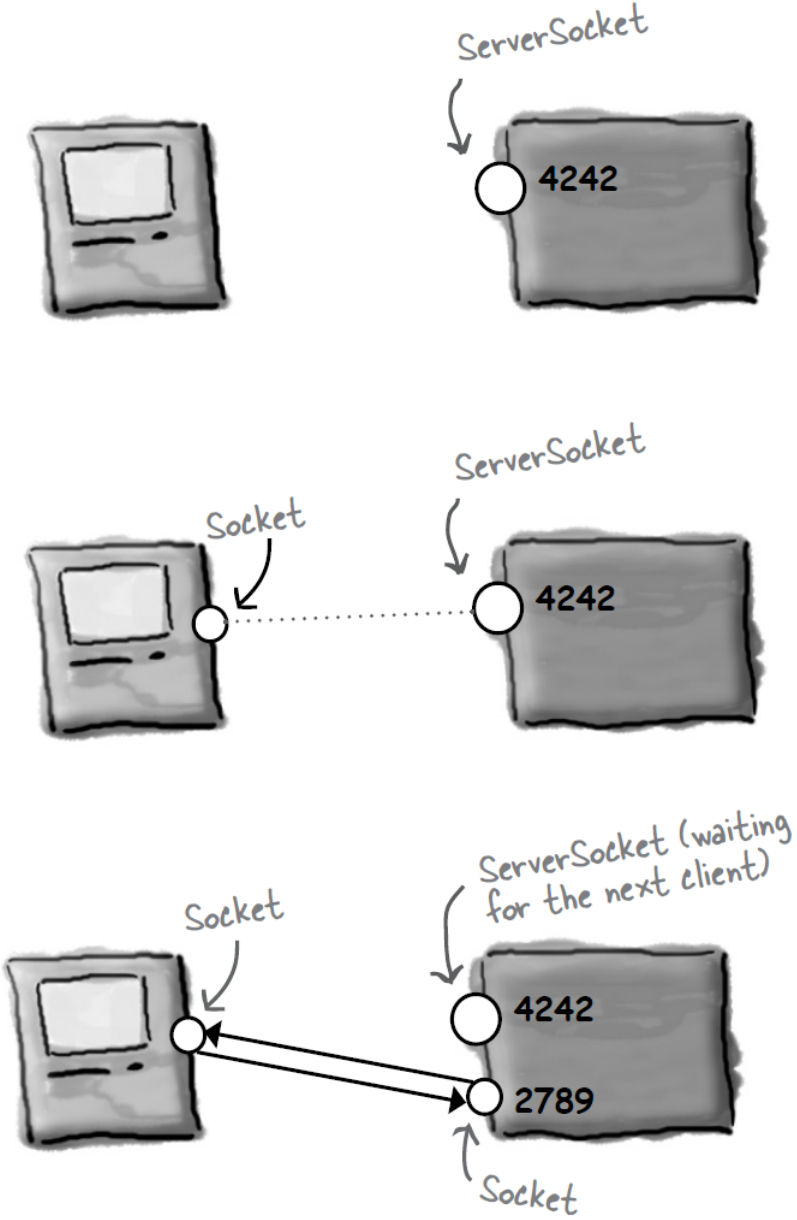
Magic 8 ball: Persistent connections

- Original version:
  - One prediction per connection setup/teardown
- Persistent version:
  - Establish a ***protocol*** between client and server

Client	Server
	Wait for client
Make connection to server	
	Send first fortune
Receive first fortune	
Send "MORE"	
	Receive command
	Send second fortune
Receive second fortune	
Send "QUIT"	
Close socket	Close socket

Magic 8 ball: Multi-threaded server

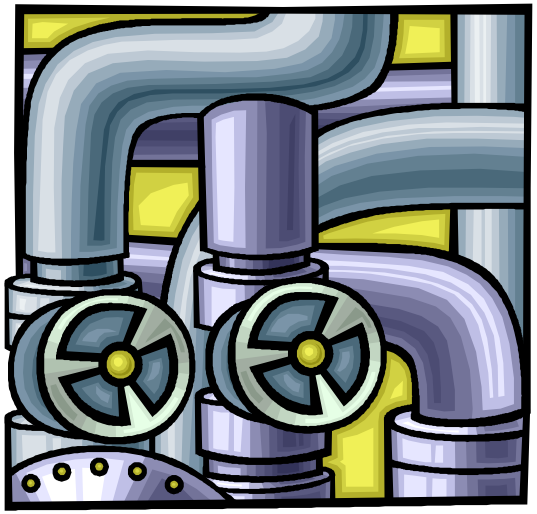
- Problem with persistent version:
  - One client can hog the 8-ball for a very long time
- Multi-threaded server:
  - Spawn a thread to handle each incoming client
  - Server's main thread can then go back to waiting for a new incoming client



Programming activity

- Create a client to GET/PUT values on server
  - Connect to my server: katie.mtech.edu, port 5000
  - Server stores a shared HashMap of (key, value) pairs
  - Client reads a line in from StdIn
  - Protocol:
    - Client starts, sends single line of text:
      - GET <key>
      - PUT <key> <value>
      - PRINT
      - QUIT
    - Server returns a single line of text, one of:
      - RESULT <key> <value>
      - OK
      - ERROR
      - List of key value pairs

Summary



- Networking and sockets
  - IP addresses, port numbers
  - Getting connected:
    - Setup expense
    - Firewall problems
  - Message delivery is:
    - Reliable and in order (at least how we're currently doing it)
    - Incurs some latency to travel over the interpipes
- Building socket client/servers
  - One-hit wonder versus persistent interaction
  - Single threaded versus multi-threaded server