CSCI 339
Distributed Systems

Jeannie Albrecht

Feb 1, 2023
Administrative Details

- Class roster
- Class syllabus (see webpage!)
- Class meeting time: MTh 1:10 – 2:25
- Meeting location: Schow 030A
- Lab code: #392781
- Unix accounts? Gitlab?
- Course Webpage:
Today's Outline

• Course Mechanics

• Course Topics/Outline
  • Distributed Systems problem statement
Syllabus

• How to contact me
  • Office: TCL 305
  • Office hours (tentative): Tue 1-2:30pm, Wed 1-2:30pm, by appt
  • Webpage: http://www.cs.williams.edu/~jeannie
  • Email: jeannie@cs.williams.edu

• Textbook
  • Recommended: Distributed Systems, by Van Steen and Tanenbaum, 4th ed.

• TA: it’s complicated….
Honor Code and Ethics

- The student handbook describes the Honor Code and Computer Ethics guidelines.
- You should also know the CS Dept computer usage policy.
- If you are not familiar with these items, please review them ASAP.
- No public Git repositories!
- No searching for solutions on the web
- No ChatGPT blah blah blah
- **We take these things very seriously!**
Audience

• Why should you take this course:
  • Interested in learning about large-scale dist sys
    • We will learn how systems work
    • We will learn “by doing”
  • Interested in graduate school
  • Interested in industry positions at popular places
    • Google, Microsoft, Meta, Cisco, Akamai, IBM, Apple, startups, etc.
General Goals

• Gain background in networking & distributed systems
  • Lecture
  • Textbook (provides background and context for class discussions and projects)

• Gain understanding of research issues
  • Study of relevant research papers (“written” homework)

• Learn how to build large-scale distributed systems
  • Lectures, programming projects

• Overall goal: Emphasize “why” and “how” over “what”
Non-Goals

- Teach the basics of systems programming
  - Assume some general familiarity with operating systems, networks, C, C++, Python, Java
  - I realize that some of you have not taken OS or networks…that's OK, we'll review the basics!
  - BUT if you feel like you need more background info on a particular topic, come see me (or read your textbook)
- 75 minute soliloquies!
  - Lectures should be interactive
    - EVERYONE should participate (seriously!)
Grading

- 5% Class Participation
- 10% Homework
  - Written paper evaluations
- 30% Midterm exam (format TBD)
- 35% Programming projects (3 in total)
- 20% Final Project
  - Including paper and presentation
  - Start thinking about possible topics
Written Homework: Paper Evaluations

• Will read ~10 research papers on constructing high-performance distributed systems
• ~1 page evaluation of reading for each paper
• Evaluations completed in advance of class (mostly Thur)
  • Submit on GLOW
  • Grade entirely based on effort
• Include 3 paragraphs:
  • (1) Summary that highlights biggest contributions of the paper
  • (2) Most glaring problem(s) with the work (think about this!)
  • (3) What conclusions you draw for building robust, scalable distributed systems (this will get easier with practice…)
• Learn to be critical without being cynical!
Programming Projects

• 4 projects spanning the semester (*including* final project)
  • Hands on construction of interesting distributed services
  • You’ll have ~2 weeks per project (~1 month for final project)
  • Should work in teams of 2 (or 3 with approval)
  • *Start early!*

• Assignment 1 will be officially assigned on Monday:
  • Build an HTTP Server in C/C++
  • Support HTTP/1.0 and subset of HTTP/1.1 functionality
  • Submit basic evaluation and write-up of system design
  • Details are posted on web page (and we'll discuss in class)
  • Look for an email/google form about partners!
Project 1 Preview: Sockets API

- Creating a socket in C – see “man socket” on lab machines
  ```c
  int socket(int domain, int type, int protocol)
  domain = AF_INET (IP)
  type = SOCK_STREAM (tcp), SOCK_DGRAM (udp)
  protocol = IPPROTO_IP (IP...just use 0)
  ```

- Passive Open (on server)
  ```c
  int bind(int socket, struct sockaddr *addr, int addr_len)
  int listen(int socket, int backlog)
  int accept(int socket, struct sockaddr *addr, int addr_len)
  int select(int n, fd_set *readfds, fd_set *writefds,
             fd_set *exceptfds, struct timeval *timeout);
  ```

- Bind Example:
  ```c
  struct sockaddr_in my_addr;
  my_addr.sin_family = AF_INET;
  my_addr.sin_port = htons(8888);
  my_addr.sin_addr.s_addr = INADDR_ANY;
  int num = bind(sockfd,
                  (struct sockaddr *)&my_addr,
                  sizeof(struct sockaddr))
  ```
Sockets API

• Active Open (on client)
  int connect(int socket, struct sockaddr *addr, int addr_len)

• Sending/Receiving Messages
  int send(int socket, char *msg, int mlen, int flags)
  int recv(int socket, char *buf, int blen, int flags)
Server/HTTP Protocol

• HTTP Server
  • Creates a socket
  • Binds to an address
  • Listens to setup accept backlog (backlog = # pending connections)
  • Can call accept to block waiting for connections
  • Can call select to check for data on multiple sockets

• Handling requests
  • Hand off to separate thread? Separate process? Event driven?

GET /index.html HTTP/1.0 
<optional body, multiple lines> 

(Don't type \n, just use carriage return)
Outline

• Course Mechanics

• Course Topics/Outline
  • Distributed Systems “problem statement”
Internet History
Vannevar Bush

- **Summary**: Vannevar Bush established the U.S. military/university research partnership that later developed the **ARPANET**.

- Quote: “Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, 'memex' will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.”

- “It consists of a desk, and while it can presumably be operated from a distance, it is primarily the piece of furniture at which he works. On the top are slanting translucent screens, on which material can be projected for convenient reading. There is a keyboard, and sets of buttons and levers. Otherwise it looks like an ordinary desk.”

  - Vannevar Bush; *As We May Think*; Atlantic Monthly; July 1945.

Source: Livinginternet.com
J. C. R. Licklider

- **Summary**: Joseph Carl Robnett “Lick” Licklider developed the idea of a universal network, spread his vision throughout the IPTO, and inspired his successors to realize his dream by creation of the ARPANET.

- **Quote**: “It seems reasonable to envision, for a time 10 or 15 years hence, a 'thinking center' that will incorporate the functions of present-day libraries together with anticipated advances in information storage and retrieval.”

- “The picture readily enlarges itself into a network of such centers, connected to one another by wide-band communication lines and to individual users by leased-wire services. In such a system, the speed of the computers would be balanced, and the cost of the gigantic memories and the sophisticated programs would be divided by the number of users.”
  

Source: Livinginternet.com
Background

- 1957: USSR launches Sputnik, first artificial earth satellite
  - U.S. responds by forming ARPA
- 1962: Licklider's *Galactic Network*
- 1966: Roberts (MIT) *Towards a Cooperative Network of Time-Shared Computers*
- 1967: ACM SOSP *Multiple Computer Networks and Intercomputer Communication*
The **Advanced Research Projects Agency Network (ARPANET)** was the first wide-area packet-switched network with distributed control and one of the first networks to implement the TCP/IP protocol suite.
Internet Timeline

• 1971: Tomlinson develops email program, big hit
• 1972: Telnet
• 1973: FTP
• 1974: TCP
• 1978: TCP split into TCP and IP
• 1979: USENET (newsgroup) established
• 1984: 1000 hosts connected to Internet, DNS debuts
• 1988: Internet worm brings down 10% of Internet
• 1991: WAIS, Gopher, WWW released
Internet Growth Trends

- 1977: 111 hosts on Internet
- 1981: 213 hosts
- 1983: 562 hosts
- 1984: 1,000 hosts
- 1986: 5,000 hosts
- 1987: 10,000 hosts
- 1989: 100,000 hosts
- 1992: 1,000,000 hosts
- 2001: 150 – 175 million hosts
- 2002: 200+ million hosts
- 2011: 850+ million hosts (2.1 billion users, 30% of world pop.)
- 2023: 5+ billion users
Challenges

• What are some of the principal challenges to building distributed systems and network services?
  • Repeated themes throughout the semester
Major Course Topics

• Internetworking Review
  • Not all computers are directly connected to each other
  • The Internet Protocol (IP)

• End-to-End Protocols
  • Provide the abstraction of a reliable byte-stream over error-prone, packet-switched network
  • Transmission Control Protocol (TCP), Congestion Control

• Modern network services
  • How is Google/Amazon/etc actually built?
Major Course Topics (cont.)

- Naming
  - Given a name for an Internet resource, how do you find it?
- Remote Procedure Call
  - Client/Server systems
  - Run commands on computers around the world
  - Make remote procedure calls look like local procedure calls
- Distributed operating systems/file systems
  - How do you break down a centralized service to run across multiple machines?
  - Client/server, cluster computing, peer-to-peer
Major Course Topics (cont.)

• Consensus/Consistency
  • How do we ensure that all servers have the same version of our data?
  • How do we provide *mutual exclusion* in a distributed environment?

• Replication/Fault Tolerance
  • Replicate service contents for increased availability, performance, etc
  • But the copies must remain in sync!

• Security
  • How to ensure authenticity and integrity of data transmitted across machines?
Networking Goal: Scalable, Arbitrary Communication
Networking Goal: Scalable, Arbitrary Communication
Networking Goal: Scalable, Arbitrary Communication
Challenges to Achieving Universal Communication

- How to connect computers
  - Cannot have all-to-all connections

- How to *name* and *locate* computers
  - Billions of computers/devices: translate name into physical location

- Routing
  - Transmitting messages from one computer to another

- Software/Protocols
  - Don't just send random messages, must agree on format and interpretation

- Reliability
  - Networks drop, corrupt, and reorder messages
  - Hosts fail and are compromised

- Common challenge is *scalability*
Distributed Systems Goals

Provide the illusion of a single large "virtual" machine
Distributed Systems Applications

• Why would you want to build a distributed application?

• What are some compelling distributed applications?
Distributed Systems Applications

• Why would you want to build a distributed application?
  • Access remote resources not available locally
  • Transmit information to users of remote machines
  • Bring to bear additional processing power on a single task

• What are some compelling distributed applications?
  • Telnet/FTP
  • Email
  • WWW
  • Games
  • Social networking
  • …
Distributed Systems Challenges

- **Fault tolerance**
  - "Distributed computing is where a computer I have never heard of keeps me from getting my work done."
    - Leslie Lamport
  - Strive to maintain the illusion that remote resources are available locally, but it is very hard to mask failures

- **Performance**
  - Speed of light constraints, network congestion, unpredictable server load, network variability

- **Data consistency**
  - Data available at multiple sites, how to keep data consistent
  - E.g., Web caching