CIS 455/555: Internet and Web Systems

Introduction

September 1, 2021
Agenda

- Introduce the Course
- Course logistics
- Let's get started...
Internet and the Web Systems!

Two different definitions of “Internet”:

1. The networking infrastructure that links all connected devices

2. The entire ecosystem of networked applications that uses the basic connectivity provided by the definition above

In this class, we’ll be using the second.
What this course is about
Wait a minute...

- What is actually going on behind the curtains?
  - How is the machine responsible for 'www.google.com' found?
  - What systems and protocols are involved in getting the page from California to my laptop?
  - How do services like Amazon or Facebook scale to millions of requests every day?
  - How come they are almost always available?
  - How would you program a system of this scale?
  - How does Google download 'the entire web'? What kind of system can hold that much data?
  - How do you answer a search in 0.69 seconds, and find the most relevant pages among 151,000,000 candidates?
  - Why do the best results always seem to be on the first page?
  - How did Google know I meant the city center in Philadelphia?
  - ...

Main themes

- CIS455/555 focuses on services like Google, Akamai, iTunes, Facebook, eBay, ...

- Main goals:
  - Understand the **technology** (DNS, HTTP, TCP/IP, cookies, ...)
  - Underlying **principles** (consistency models, scalable algorithms, ...)
  - Basics of system **design** / how to make the right choices
  - How to handle massive amounts of **data**
    - Find it, gather it, encode it, store it, process it, integrate it, keep it consistent, extract useful answers from it, rank it, deliver it quickly, prevent its loss, ...
  - **Understanding risks**, e.g., security/privacy, and countermeasures
  - **Case studies** of concrete systems (Google, AWS, BitTorrent, ...)
  - Technical **skills** (using cloud platform, programming a data center)
  - Hands-on **experience** with a concrete system (web search)
    - Working in teams, evaluating your solution, ...

- This is **not** a course on building web sites!
Relationship to other courses at Penn

- NETS 212 and CIS 545
  - Cloud service layers; key/value stores
  - Basics of MapReduce, Spark, and data-parallel programming

- CIS 450/550
  - Data representation and management; DBMS-backed web sites
  - Relational querying with SQL; XML querying with XQuery

- CIS 553
  - TCP/IP, HTTP, DNS; the protocols that underpin the Internet
  - 455/555 will teach you what you build on top of that infrastructure

- CIS 505
  - About 25% overlap with this class
  - 505 focuses more on fundamentals and basic building blocks
Agenda

- Introduce the Course
- Course logistics
- Let's get started...
Your team this semester

Instructor: Vincent Liu (Me)

Daniel Bessonov
bessonov@seas

Peter Baile Chen
cbaile@seas

Ameya Gadkari
ameyaga@seas

Matt Lebermann
mleb@seas

Wesley Yee
wesyee@seas

Karen Zheng
karenzxy@seas

University of Pennsylvania
Course Format

- 3 (5) substantial programming assignments
  - HW1 (MS1&2), HW2 (MS1&2), HW3

- Major final project
  - In a team of 4, build Google (circa late 1990s)

- Two midterms
  - Covering the two halves of the course

- Participation
  - Attendance, in-class contribution, Piazza, office hours
  - This portion will only affect extreme cases
Tools

- **Website** ([https://www.cis.upenn.edu/~cis455/](https://www.cis.upenn.edu/~cis455/))
  - Schedule, homework assignments, readings, lecture slides

- **Canvas** ([https://canvas.upenn.edu/courses/1606917](https://canvas.upenn.edu/courses/1606917))
  - Lecture recordings

- **Gradescope (through canvas)**
  - Homework turn-in

- **Piazza** ([https://piazza.com/upenn/fall2021/cis455555](https://piazza.com/upenn/fall2021/cis455555))
  - Discussion, announcements
Piazza

Piazza will also be used for
- Announcements, e.g., cancelled classes (if necessary)
- Supplemental materials, e.g., links to relevant papers or articles
- Corrections and clarifications, e.g., bugs in the homework handouts
- Please check the group frequently!

You may ask questions or answer them
- Incentive: top 3 students by answer count will get 2 points on their final grade!

Please sign up at
- https://piazza.com/upenn/fall2021/cis455555
Textbook

  - We'll read from the book ~50% of the time
  - Electronic edition available for free from the author's web page (hardcopy $33 on Amazon)

- Frequent supplementary reading
  - Excerpts from several books; many recent research papers
  - There is a 'reading list' on the web page as well; this contains the assigned readings plus some additional background papers

- Your first one was assigned for today:
  - Hints for Computer Systems Design
  - we will be discussing this next time!
Prerequisites and workload

You will need:

- Ability to code in Java (substantial implementation project!)
- Good debugging skills -- this will be the biggest time sink!
- Problem-solving skills (not all the answers will be on Piazza)
- Ability to work in a team with classmates (towards the end)
- Willingness to learn how to read 'real' API documentation
- Some exposure to threads / concurrent programming
WARNING!!!

- This course is a lot of work! Likely one of the hardest projects in the CIS department

- Should be considered 1.5-2 CU

- Many drop the class after attempting HW 1
But...

- Typical feedback: "The course was hard, but we learned a lot"

- Payoff:
  - Practical development, debugging, SW engineering experience
    - One of the courses our alumni cite as "most useful for passing my interview"
  - Good knowledge of the fundamentals behind scalable systems
  - A working "academic clone of Google,"

- **Homework 0**: Install VM, write 'Hello World'
  - Opportunity to familiarize yourself with this environment
  - Gives us a chance to iron out technical problems before HW1
  - Handout will be available on the course webpage
    - http://www.cis.upenn.edu/~cis455/
Policies: Collaboration

- All assignments must be done **individually**
  - Only exception is the final project (teams of four)
  - All the code you submit has to be your own
    - Only exception: Code we have provided or explicitly authorized
    - **NO** code you have found on the web. **NO** sharing with others.
- Penn's Code of Academic Integrity applies
  - No cheating, plagiarism, fabrication, multiple submissions, gaining an unfair advantage, or facilitating (!) academic dishonesty
  - **It's not worth it!!** Penalties can be severe: http://www.upenn.edu/academicintegrity/ai_violations.html

- **Zero tolerance policy** to ensure fairness
  - We will use various tools to actively look for cheating
    - These tools work: We have caught a number of cases in the past

- Yes, we're serious!
  - I bring several cases to OSC every year
Policies: Extensions

- Everyone gets 5 late days
  - Deducted in increments of 1 day
  - No questions asked

- Absolutely no extensions without late days

- Some recommendations:
  - Please start the homeworks early!!!
  - Save your late days for unforeseen events
    - Interview calls, deadlines in other courses, tricky bugs in your code, ...
Policies: In-person class and COVID

- Assumption is that you will make an effort to come to class

- If you feel sick or get a red pass:
  - Do not come to class!
  - Email me to let me know
  - Lectures will be recorded and participation will be excused

- If you don’t feel comfortable coming to class
  - That’s fine, just let me know now
  - I.e., don’t come to me at the end of the semester asking for a retroactive change to your participation grade
Policies: Other medical emergencies

- What if you can’t submit a homework, or attend a midterm or project demo, because of an emergency?
  - Example: Emergency room visit, severe illness, death in the family

- In this case, you should:
  - let me know as soon as possible (if at all possible and practical)
  - save the relevant documentation
  - once you’re better, speak with Sonya Gwak from RAS
  - If and when Sonya confirms that there was a real emergency, we can talk about accommodations (extension, make-up exam, …)

- Some examples of things that are not emergencies:
  - Interviews, other classes, travel, conferences, cooking club, etc.
  - If is your responsibility to schedule these to avoid a conflict
Policies: Waitlist

- In typical semesters, we start with a long waitlist

- Many, many spots open once folks start HW1...
  - We can usually accommodate everyone who actually wants to take the course

- Students who attend class and turn in assignments will be given priority
Disclaimer

- This has always been a "bleeding edge" course
  - The technology in this field changes quickly, and the course staff have been trying to include new and emerging trends
  - This means that we'll be using some immature technology
  - Not everything has been tested and validated ahead of time; we'll do the best we can to smooth over the bugs

- Focus is on hands-on experience
  - Substantial implementation project; goal is a level of comfort in managing large, complex software development with 3rd party code
  - As in the real world: learning APIs, dealing with inadequate tools, working with incomplete and/or ambiguous specifications (RFCs etc.)!
  - Most of you will find this a struggle!
  - You’ll spend many hours debugging!

- We hope it will be a fun course, though...
  - ... and an interesting one!
Agenda

- Introduce the Course
- Course logistics
- Let's get started...
What exactly is the Internet?

- Founded on tens of thousands of interconnected networks
  - Examples: AT&T, Comcast, Level 3, Penn, Verizon, ...
  - Provides global connectivity

"Backbone" of some large networks

My laptop (in PHL)

Many access networks

Google's servers (far, far away)
The Internet is built in layers

<table>
<thead>
<tr>
<th>L1/2: Physical/Link</th>
<th>L3: IP</th>
<th>L4: Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFi, UMTS, ZigBee, Ethernet, WiMax, ...</td>
<td>IPv4, IPv6 Unicast, (multicast)</td>
<td>SSH, FTP, HTTP, IM, P2P, …</td>
</tr>
</tbody>
</table>

| | | Lightweight streaming, etc. |
|---------------------|---------------------------|
| TCP (session-based) | UDP (sessionless) | Web Services, distrib. transactions, … |

Your Application
What exactly is the Web?

- A collection of documents and other resources that are interconnected (via URLs)

```html
<html>
<body>
<img src="cat.gif">
<a href="otherdoc.html">link</a>
</body>
</html>
```
What exactly is the Web?

- A collection of documents and other resources that are interconnected (via URLs)
  - Implemented by HTTP servers that publish HTML, XML, and a few other content types, plus a huge number of web clients
  - Builds on a number of Internet protocols: DNS, TCP, IP, ... (other Internet services exist and use other protocols)
  - Web services and custom applications sometimes use HTTP in ways it was not designed for
What is an Internet system?

- Not just a something that can serve a web page...
- An application built over the Internet, whose functionality is distributed across multiple machines
  - Typically, at least in a client-server or server-to-server fashion, but may have many more participants
  - Typically, data and/or code must be exchanged in distributed fashion for the functioning of the application
  - Often, the data must be partitioned, replicated, translated, etc. ("shards" in Google-speak)
  - Often, there is heterogeneity, e.g., code runs in multiple different environments, is written in different languages, etc.
  - Often, there are concerns about handling failures, firewalls, attacks, ...
Example: Web search

A system: Many different components working together!

Uses a model of document/word similarity to rank matches

Search Interface Servers

Crawlers

Index Servers

Web pages

Content

queries

HTML forms; results

query

results

keywords + locations
A closer look

URL server

Crawler

Anchors

Repository

Indexer

Lexicon

Barrels

Sorter

Doc index

PageRank

Searcher

Web frontend

URL resolver

© 2021 A. Haeberlen, Z. Ives, V. Liu

University of Pennsylvania
Data centers

Data centers (size of a football field)

Google data center in The Dalles, Oregon

Source: 1&1
What other systems work like this?

- Social networking sites: Facebook, LinkedIn, ...
- e-commerce sites: Amazon, eBay, Expedia, ...
- Streaming video services: Netflix, YouTube, ...
- VoIP services: Zoom, Skype, Google Meet, ...
- Multiplayer games: World of Warcraft, ...
- Cloud platforms: EC2, Azure, ...
- Content distribution networks: Akamai, ...
- Collaborative sites: Wikipedia, ...
- Volunteer computing: SETI@home, BOINC, ...
- ... and LOTS of other systems!
  - These systems are running a large part of our daily lives by now
  - It is important that they are reliable/secure/available etc.!
What are the hard problems?

- Some of the biggest challenges are:
  - Scalability
  - Availability / reliability
  - Consistency
  - Interoperability
  - Location and resource discovery
  - Security

- **Disclaimer**: Most of the hard problems are not solved!
  - Many cannot be solved perfectly

- Often there is not any single 'best' solution
  - Typically there is a *tradeoff*; for example, we can't have consistency, availability, and partition tolerance together
  - We need to find the right compromise for each specific problem
    - "Is A or B better?" - Typical answer: "It depends!"
Scalability

Question: How can we build services that can support a large number of clients and/or requests?

- Example: Serve >2 billion videos per day (YouTube), serve >1.2 billion queries per day on more than 27 billion items (Google)
What happens when you do it wrong

We have a lot of visitors on that.
Please stay on the

Here are a few key numbers about what the system can and cannot handle:

- The site's **response rate was originally eight seconds** — "which is totally unacceptable," U.S. Chief Technology Officer Todd Park told a House oversight committee.

- Programmers have improved that to one second, but efforts are underway to shorten the response time to a fraction of a second.

- Park said the system was designed to handle 50,000 to 60,000 concurrent users.

- Today, with fixes, it can handle about half the intended capacity with **20,000 to 25,000 concurrent users at most**, Park said.
Scalability

- **Key idea:** Distribute work!
  - Challenges: Coordination, load balancing, efficiency

- **Parts of the solution:**
  - Client-server, multi-tier, P2P architectures
  - Restricted programming models, e.g., MapReduce
  - Data partitioning, replication, remote procedure calls, ...
Availability / reliability

- **Question:** How do we ensure the system is "up" when we want it to be, and doing the "right" thing?
  - Things break all the time, and in creative ways
  - But: When was the last time you saw Google down?

- **Key ideas:**
  - Replication and redundancy; security measures against attacks; ability to undo/redo

- **Challenges:**
  - Keeping things consistent
  - Performance vs. security
  - Acknowledgments

- **Parts of the solution:**
  - Data partitioning, replication, ...
  - Logging, transactions, ...
  - Redundant hardware, multiple sites, ...
  - Quorum and consensus algorithms
Consistency

What can happen if this code runs concurrently?

1) B = Balance(Bob)
2) A = Balance(Alice)
3) SetBalance(Bob, B+100)
4) SetBalance(Alice, A-100)

1) A = Balance(Alice)
2) B = Balance(Bob)
3) SetBalance(Alice, A+500)
4) SetBalance(Bob, B-500)

Alice's balance: $200 $200 $700 $700 $100
Bob's balance: $800 $900 $900 $300 $300

© 2021 A. Haeberlen, Z. Ives, V. Liu

University of Pennsylvania
Consistency / consensus

- Question: Replication, distribution, and failures make it difficult to keep a unified, consistent view of the world – what can we do about this?

- Key ideas:
  - Locking, concurrency control, and invalidation schemes
  - Clock synchronization

- Challenges:
  - Locking has huge performance overhead
  - Network partitions, disconnected operation

- Parts of the solution:
  - Optimistic concurrency control, 2-phase locking
  - Distributed clock synchronization
  - Conflict resolvers
Interoperability

- **Question:** How do we coordinate the efforts of components that have different data formats and/or source languages, and are on different machines?

- **Key idea:** Standardization!

- **Challenges:**
  - Everything has a different semantics!

- **Parts of the solution:**
  - Standard data formats: XML, XML schemas, JSON
  - “Schema mediation” and data translation
  - Remote procedure calls: CORBA, XML-RPC, ...
If you put a system on the Internet, it will get attacked
  - Criminals, script kiddies, activists, foreign governments, the NSA (?)...
  - You need to be ready!
Our first focus: Single machines ('servers')

- How do you handle large numbers of concurrent users?
  - Processes
  - Threads
  - Events
  - Hybrids (e.g., thread pools)
  - Staged architectures

- Next week, we'll look under the covers of an HTTP server
  - Key ideas in building scalable systems
  - Principles of HTTP and web servers
  - Management of concurrent sessions
To do: For next time

- Please read:
  - If you haven’t: B. Lampson, "Hints for Computer Systems Design", ACM SOSP 1983
  - Tanenbaum Chapter 3.1
  - If necessary: Review Tanenbaum “Modern OS,” Ch. 2.3 or a similar OS book on interprocess communication

- Please sign up for the Piazza discussion group!
  - https://piazza.com/upenn/fall2021/cis455555

- Start on HW0!
Stay tuned

Next time you will learn:
How to build a web server