CIS 505
Software Systems

Linh Thi Xuan Phan

Department of Computer and Information Science
University of Pennsylvania

Lecture 1: Introduction
August 31, 2016
About me

• My name: Linh Thi Xuan Phan

• Research interests:
  – Real-time, embedded, cyber-physical systems
  – Formal methods
  – Cloud computing

• Member of the PRECISE center and Distributed Systems group
Your TA team

Bipeen Acharya

Hitali Sheth

Chirag Shah

Siri Anil
Can you hear me now?

- If you are sitting in the back, your job is to wave your hand if you can't hear me!
Why CIS505?

• What do all these systems have in common?
  – They all have a large distributed system at their core
What distributed systems look like

- Often tens to hundreds of thousands of machines
- Can be globally distributed & highly heterogeneous
Are these systems hard to build?

• Yes! A couple of typical challenges:
  – Concurrency: Managing 100,000s of machines working in parallel!
  – Robustness: Keeping the system up, e.g., 99.999% of the time
  – Scalability: Growing the system by 2x/10x/1000x without breaking it
  – Efficiency: How to get the most out of the available resources
  – Abstraction: How to help system designers build better systems
  – Coordination: Getting the machines to work with each other
  – Consistency: Getting the system's responses to 'make sense'
  – Fault tolerance: System needs to work despite component failures
  – Security: Preventing the bad guys from breaking everything
  – Availability: Making sure that data isn't lost, service remains up, etc.
  – Heterogeneity: Working with lots of different kinds of components
  – Timeliness: Getting the data when you need it; getting good QoS
  – Laws of physics: Speed of light is finite, etc.
  – ...
What happens if you get it wrong?

We have a lot of visitors on the site right now. Please stay on this page.

We’re working to make the experience better, and we don’t want you to lose your place in line. We’ll send you to the login page as soon as we can. Thanks for your patience!

In a hurry? You might be able to apply faster at our Marketplace call center. Call 1-800-318-2596 to talk with one of our trained representatives about applying over the phone.
How this course helps

• This course will teach you some of the fundamental building blocks of distributed systems
  – How to properly coordinate parallel processes, how to synchronize activities, how to pick a consistency model and how to enforce it, ...
  – These building blocks are at the heart of pretty much any large distributed system (whether it is Google or the NY stock exchange)

• We will look at some case studies of real systems
  – Apache Spark, Apache Kafka, Amazon Dynamo, ...

• And you will gain some practical experience with building these systems
  – 3-4 programming assignments and a final team project
Plan for today

• Introduction
• Course logistics
• Overview
# Course staff

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Email</th>
<th>Office hour</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linh Thi Xuan Phan</td>
<td>Instructor</td>
<td>linhphan@cis</td>
<td>Mondays 1:00-2:00pm</td>
<td>Levine 464</td>
</tr>
<tr>
<td>Bipeen Acharya</td>
<td>TA</td>
<td>acharyab@seas</td>
<td>Tuesdays 5:00-6:00pm</td>
<td>Levine 612</td>
</tr>
<tr>
<td>Hitali Sheth</td>
<td>TA</td>
<td>hitalis@seas</td>
<td>Wednesdays 1:00-2:00pm</td>
<td>Levine 6th floor bump space</td>
</tr>
<tr>
<td>Chirag Shah</td>
<td>TA</td>
<td>chirags@seas</td>
<td>Thursdays 12:30-1:30pm</td>
<td></td>
</tr>
<tr>
<td>Siri Anil</td>
<td>TA</td>
<td>anilsiri@seas</td>
<td>Fridays 1:00-2:00pm</td>
<td></td>
</tr>
</tbody>
</table>
Course discussion group

• We are going to use Piazza for:
  – Questions about the lecture, or about assignments
  – Announcements (office hours moved, etc.)
  – Supplemental materials
  – Corrections and clarifications (e.g., for handouts)

• You can sign up here:
  – https://piazza.com/upenn/fall2016/cis505

• Please check Piazza regularly!
Prerequisites

• To succeed in this course, you'll need
  – An undergraduate OS and/or networking course (e.g., CIS380)
  – Programming skills in some language

• We will be using C/C++ for the assignments
  – You don't have to have experience in C/C++ specifically
  – However, if you don't, I expect that you are willing to invest the time to catch up (e.g., by reading a good textbook)
  – I won't be able to spend class time on programming basics
Workload

- Note: The class is changing a bit this year
  - Based on feedback from previous years' students
  - No more written homework assignments, more programming assignments (and more difficult ones)
  - Please do send me feedback and suggestions on the assignments! (Example: Too hard / too easy / not interesting)

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4 substantial programming assignments</td>
<td>30%</td>
</tr>
<tr>
<td>Final group project</td>
<td>25%</td>
</tr>
<tr>
<td>Midterm exam</td>
<td>15%</td>
</tr>
<tr>
<td>Final exam</td>
<td>25%</td>
</tr>
<tr>
<td>Participation (in-class, Piazza, group project)</td>
<td>5%</td>
</tr>
</tbody>
</table>
Other related courses

• **CIS 455/555: Internet and Web Systems**
  – Some overlap with CIS505 (e.g., naming, fault tolerance, ...)
  – Centered around web search as a case study (information retrieval, crawling, indexing, data formats, ...)
  – Counts as a MSE core course, but not towards the WPE-I

• **NETS 212: Scalable and Cloud Computing**
  – Sophomore-level class; doesn't go into as much depth as CIS505
  – Focus is more on scalability, whereas CIS505 focuses more on distributed systems fundamentals (coordination, consistency, ...)
  – Doesn't count as a WPE-I or MSE core course
The CIS505 Virtual Machine

• I am going to provide a VM with all the tools you'll need
  – Use with VMware Player, VirtualBox, etc.
  – Safe to experiment
  – Standardized environment → easier for you to get help

• Homework 0: Install the VM, write 'Hello world'
  – Idea is to try out the environment & iron out any bugs
  – Handout will be available on the course webpage soon
    • http://www.cis.upenn.edu/~cis505/
  – You will need a SEAS account
    • If you do not have one yet, please get one now!
Textbook

• We are going to use the classic Tanenbaum / van Steen book
  – Electronic version available for free: http://www.distributed-systems.net/
  – Hardcopies available from the campus bookstore

• I may assign some chapters from other books, some research papers, etc.
  – Readings will be posted on the course web page (in the schedule at the bottom): http://www.cis.upenn.edu/~cis505
Policies: Collaboration

• All assignments must be done individually
  – All code you submit has to be your own
    • Only exception: Code we have provided or explicitly authorized
  – No code from the Internet, no sharing with classmates
  – Penn's Code of Academic Integrity applies
    • Includes not only cheating but also "gaining an unfair advantage" or "facilitating academic dishonesty" (!)
    • Penalties can be severe! Some students get notes on their transcript (bad for job hunting!!) or have even had to leave Penn.
    • It's not worth it!!

• Zero tolerance policy
  – We are going to actively look for cheating in various ways
  – If you are caught, we will refer you to OSC
Quiz

• Can we work on assignments together?
• Can I discuss the assignment with others (in general terms)?
• Can I use code from the Internet?
• Can I ask questions about assignments on Piazza?
• I just happened to leave my git key on the table, and my classmate happened to find it. Will I be penalized for this?
Policies: Extensions

• You should submit all assignments by the deadline
  – Late submission cost: 20% per day (or partial day)
  – 1 minute late: -20%. 2 days late: -40%. 5 days late: zero

• However, I understand that things can happen, so everyone gets three late days for free
  – You will each get three 'jokers', which you can use to extend a deadline by 24 hours. No questions asked!
    • You may use jokers at any time before the deadline.
    • You can use more than one joker (or all three jokers) on a given assignment
    • You can apply the jokers one by one (e.g., one before deadline, one the next day)
    • No fractional jokers! No refunds! Absolutely no additional jokers!

• Please start working on the homeworks early!
Disclaimer

• I made some substantial changes to the course this semester
  – New programming assignments
  – New infrastructure (VM, web submission system)
  – Some new topics

• Hopefully the new course will be more fun (and more interesting) for you!

• But: Not everything will work perfectly right away
  – If something doesn't work, please let one of us know immediately (Piazza!), and give us a chance to fix it
  – Please check Piazza for updates regularly
Potential payoff

• What can you expect to get out of this course?
  – Insights about how large distributed systems work 'under the hood'
  – Toolbox of solutions for common challenges
  – Practical experience with distributed systems
  – Portfolio of systems you have built (e.g., for use in interviews)

• These skills are in high demand right now!
  – E.g., if you want to work for Google, Facebook & friends
Plan for today

• Introduction
• Course logistics
• Overview
What is a distributed system?

- A collection of independent computers that appears to its users as a single, coherent system
  - Typically connected by a network, so that the computers ("nodes") can interact by sending messages to each other
  - Often aims to achieve some degree of transparency: The user shouldn't have to know that there are multiple computers
Topic #1: Processes and threads

- Two key challenges in distributed systems are concurrency and communication
- These are important even on individual nodes!
  - Concurrency: Processes/threads, multiple cores, ...
  - Communication: Inter-process communication (IPC), file I/O, ...
  - But some things are much simpler on a single node. Which ones?

- We will start by studying individual nodes
  - Process & thread basics; fork(), exec(), wait()
  - Address spaces
  - How the kernel implements these abstractions (PCBs, context switching, system calls, user- vs. kernel-threads)
  - IPC mechanisms
Homework #1: Parallel sort

• The first homework is to build a program for sorting numbers in parallel
  – Nice opportunity to get familiar with C/C++ (if you aren't already)
  – Uses different kinds of parallelism (threads, processes) and coordination mechanisms
  – Experimental component (evaluating scalability)
Topic #2: Concurrency control

• What can go wrong when you have concurrency?
  – Concurrent activities can 'step on each others' toes'
  – Examples: Race conditions, deadlocks, starvation...
  – Classical problems: Bounded-buffer problem, producer-consumer problem, dining philosophers, ...
  – What can you do about these?

• Synchronization can help!
  – We will study various synchronization primitives
    • Still focusing on individual nodes for now
  – Examples: Semaphores (and their variants), monitors, condition variables
  – We will also look at how these can be implemented!
    • CLI/STI, test&set, etc.
Topic #3: Communication

- Next, we'll look at ways in which different nodes can talk to each other
- We'll initially focus on UNIX sockets
  - Basics: OSI model, endianness, port numbers, stream vs. datagram sockets, ...
  - Socket programming: Thread pools, polling, select(), event-driven programming, ...

- Then we'll look at ways to increase convenience
  - Direct message passing is tedious & error-prone
  - Wouldn't it be nice to have tools to help us with this?
  - RPC, stub-code generators, pass-by-value vs. pass-by-reference, RPC binding, synchronous vs. asynchronous RPC
Homework #2: Email servers

- Homework #2: Build simple SMTP & POP3 servers
  - Relatively simple protocols (feasible to build from scratch)
  - Your solution will work with standard email clients, e.g., Thunderbird
  - Result: Experience with socket programming, client/server
Topic #4: Naming

• Finding things in a distributed system can be hard
  – Which node currently holds the content I am looking for?
  – Which node is hosting the user I am trying to reach?

• Naming systems can help with that!
  – Many different ways to define and organize namespaces
  – Many different ways to build systems that resolve names

• We will look at:
  – Naming basics, name spaces, iterative vs. recursive resolution
  – Case study: Domain Name System (DNS)
  – Case study: Lightweight Directory Access Protocol (LDAP)
Topic #5: Clocks

• Clocks matter in distributed systems
  – Many algorithms depend on it! Examples: Transaction serialization, network fault diagnostics, authentication protocols (Kerberos), etc.

• But there is no single "global" clock!
  – Each node has its own clock; clocks are not necessarily precisely in sync and can drift.
  – Also, the laws of physics are in the way!
  – If we can't have "real" synchronized clocks, can we at least have a clock that has properties that "matter"?
    – Which are these?

• We will look at:
  – Ways to synchronize clocks between nodes (Example: NTP)
  – Logical clocks; happened-before relations
  – Lamport clocks; vector clocks
Topic #6: Coordination

• Sometimes we want a single entity to coordinate an activity
  – Examples: Serialize updates, allocate resources, ...
  – If multiple entities do this concurrently, the result is utter confusion!

• How can we do this?
  – On a single node, we could use mutexes, semaphores, etc.
  – What is different in a distributed system?
    • On a single node, the kernel and the hardware can help us!
    • But in a distributed system, there is no single 'kernel'!

• We will look at different ways to coordinate
  – Distributed mutex: Centralized and decentralized solutions
  – Token passing; leader election; bully algorithm
  – Group communication; ordered multicast
Homework #3

• The third homework assignment will be about a system that uses group communication
  – I haven't decided yet which one
  – Stay tuned!
Topic #7: Fault tolerance

• Nodes in a distributed system sometimes fail
  – Causes: Hardware faults, software bugs, operator error...
  – Can we make sure that the system remains operational?

• Idea: Replicate function on multiple nodes!
  – Example: Keep multiple copies of the same file
  – File is available as long as at least one copy survives
  – But this creates new challenges! Name some of them?

• We will look at:
  – Replication protocols: Primary/backup, quorum-based, etc.
  – Logging and recovery; 2PC/3PC; Chandy/Lamport algorithm
  – Consistency models, and techniques for enforcing them
  – Non-crash (Byzantine) faults and how to tolerate them
Many distributed systems need to store some state
- Nice case study that ties together several of the things we've talked about so far (naming, replication, consistency, ...)
- This is typically done with a file system

What would you expect of a good file system?
- Not so clear! There are many 'standard' properties (e.g., durability), but others depend on what the application needs.

We will look at:
- Single-node file systems; disk layout; space allocation; free space management; directories (Case studies: FAT and UNIX file systems)
- Distributed file systems (Case studies: NFS and Coda)
Second half of the course

• After covering the fundamental building blocks, we will look at a number of real systems

• Likely topics:
  – GFS: Google's datacenter file system
  – Bigtable: A structured storage system on top of GFS
  – MapReduce: Google's framework for Big Data processing
  – Spark: A more recent Big Data framework, based on RDDs
  – Apache Kafka: A high-throughput publish-subscribe system
  – Chord/CFS: A DHT and a highly scalable file system
  – Dynamo: Amazon's scalable key-value storage
  – Zookeeper: A configuration management system

• This will keep us busy during the second half of the semester
Final group project

• The final project will be to build a real distributed system, in teams of three
  – The project will build on some of the homework assignments

• Maybe a multiplayer game?
  – Group communication for coordination
  – Servers could be replicated for fault tolerance
  – We would provide some example code, e.g., for the GUI

• (Details TBA)
That's it for today!

• I hope you will enjoy CIS505!

• To-dos for next time:
  – Join the Piazza discussion group (if you haven't already)
  – Get your copy of the Tanenbaum book
  – Watch out for "HW0" (the 'Hello World' assignment)
    • If you find any bugs or problems with the VM, please let us know right away!

• Any questions?