CMPS 4760/6760
Distributed Systems
Prof. Zizhan Zheng
Spring 2021
Agenda

- Administrative trivia’s
- An Overview of Distributed Systems
Personal

- Instructor: Prof. Zizhan Zheng
  - Office: Stanley Thomas 307B
  - Email: zzheng3@tulane.edu
  - Office hours (virtual): Wed 2-3 pm and by appointment
Meeting Plan

- Hybrid class (60% on ground, 40% online)
  - Jan & Feb: Tu – on-ground (ST 302), Th – online
  - Will meet more in person in Mar and Apr

- Class meeting time: Tu & Th 2:05pm-3:15pm
  - All classes will be recorded
Attendance Policy

- All students should follow the hybrid class meeting schedule unless
  - you don’t feel well or have COVID symptoms
  - you have been approved for remote learning

- If you cannot attend class for any reason, you are responsible for communicating with me to make up any work you may miss

- If you are sick or told to quarantine, you can contact your case manager, and have your case management contact me directly
Course Overview

- **Objective:** understand the *key concepts* and *fundamental principles* in the design and analysis of distributed *systems* and distributed *algorithms*

- **Course Webpage**
  - Used to post weekly schedule, assignments, lecture slides, reading assignment, etc.

- **Prerequisites**
  - Knowledge of systems, networking, and algorithms
  - Comfortable with rigorous mathematical reasoning
  - Programming skills
Textbook and References


- References

- Research papers, tutorials, etc.
Grading Policy

- Homework - 25%
- Labs - 15%
- Presentation - 5%
- Midterm - 20%
- Final Exam - 25%
- Class participation - 10%

- All grades will be posted on Canvas

A >= 93%  A- >= 90%
B+ >= 87%, B >= 83%, B- >= 80%
C+ >= 77%, C >= 73%, C- >= 70%
D >= 60%
F < 60%
Homework and Labs

- Homework assignments (4-5)
  - Written problem sets
  - Graduate level students will be given extra questions that require advanced algorithmic and/or analytic techniques.

- Labs: programming assignments (2-3)
  - Java, Python

- Requests for a homework/lab assignment extension (with a valid reason) must be given to the instructor before the assignment is due
Late Policy

- 6 grace days that may be applied to homework/lab assignments
- No more than 2 grace days on any single assignment
  - assignment submitted > 2 days past the deadline (or no late day credit left) will get zero credit
- No late days for presentations and exams
Paper Presentation

- Each student will give a presentation
  - At the end of the semester
  - Two undergraduate students can work together
  - A suggested paper list will be posted on the course webpage
  - You can also suggest paper(s) you want to present
Exams

- One midterm and one final exam
  - Online exam on Zoom
  - Open-book and open-notes

- Missing an exam will result in a grade of zero. A request for a make-up exam must be given to the instructor prior to the exam date (documentation may be required).
Class Participation

- Class attendance
- Q&A in class & during office hours
- Homework solution presentation
- Online discussions
- Course Webpage
  - Used to post weekly schedule, assignments, lecture slides, reading material, etc.

- Canvas
  - Used to post grades, discussions, and reading material, etc.
An Overview of Distributed Systems

CMPS 4760/6760: Distributed Systems
Outline

- What Is a Distributed System?
- Examples of distributed Systems
- Challenges
- Architectural models
- Fundamental models
What is a distributed system

- A system in which hardware or software components at networked computers communicate and coordinate their actions only by passing messages.

- A network of processes: the processes interact with one another to achieve a goal.
What is a distributed system

The **nodes** are processes, and the **edges** are communication channels.
What is a distributed system

- Multiple processes
  - Concurrency
  - No global clock
  - Fail independently
  - No global knowledge
  - Disjoint address space
  - Collective objective

- Interprocess communication
  - Message passing vs. shared memory
  - Communication channels: delays, packet error/loss, reordering
  - Dynamic network topology
Why Distributed Systems

- Geographically distributed environment
- Speed up
- Resource sharing
- Fault tolerance
Examples of Distributed Systems

- The Internet
- World Wide Web
- Social networks: Facebook, Twitter, ...
- Peer-to-peer networks: BitTorrent, Skype, ...
- Cloud computing
- Internet of Things
- Smart Grid
- Blockchains
- ...

21
The Internet

- mobile network
- home network
- institutional network
- global ISP
- regional ISP
P2P file distribution: BitTorrent

- file divided into 256KB chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file

Alice arrives ...
... obtains list of peers from tracker
... and begins exchanging file chunks with peers in torrent
Cloud Computing
Challenges

- Heterogeneity
- Openness
- Security
- Scalability

- Failure handling
- Concurrency
- Transparency
- Quality of service
Heterogeneity

- Networks: masked by the Internet protocols
- Computer hardware: big-endian vs. small-endian
- Operating systems: programming interface
- Programming languages: data representation
- Implemented by different developers
Middleware
Scalability

- A system is **scalable** if it remains effective when there is a significant increase in the scale of the system (e.g., number of processes)

- For a system with \( n \) processes to be scalable
  - quantity of physical resources should be \( O(n) \) or lower
  - Space or time complexity should be \( O(\log n) \) or lower

- Preventing software resources running out: e.g., IP addresses
Common measures of complexity

Space complexity
How much space is needed per process to run an algorithm? (measured in terms of $n$, the size of the network)

Time complexity
What is the max. time (number of steps) needed to complete the execution of the algorithm?

Message complexity
How many messages are exchanged to complete the execution of the algorithm?
Transparency

- **Access transparency**: enables local and remote resources to be accessed using identical operations.

- **Location transparency**: enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).

- **Concurrency transparency**: enables several processes to operate concurrently using shared resources without interference between them.

- **Replication transparency**: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
Transparency

- **Failure transparency**: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

- **Mobility transparency**: allows the movement of resources and clients within a system without affecting the operation of users or programs.

- **Performance transparency**: allows the system to be reconfigured to improve performance as loads vary.

- **Scaling transparency**: allows the system and applications to expand in scale without change to the system structure or the application algorithms.
Outline

- What Is a Distributed System?
- Examples of distributed Systems
- Challenges
  - Architectural models
  - Fundamental models
Architectural elements

- Communication entities
  - Processes, threads, nodes
  - Objects: Java RMI (5.5), CORBA (8.3)
  - Components: EJB (8.5), CORBA component model
  - Web services: URI, XML-based message passing (chapter 9)

- Communication paradigms
  - Interprocess communication: message passing and multicast
  - RPC (5.3), RMI (5.4-5.5)
  - Group communication (6.2), publish-subscribe systems (6.3), ...
Client-server vs. peer-to-peer architectures

Client-server model
Client-server vs. peer-to-peer architectures
Outline

- What Is a Distributed System?
- Examples of distributed Systems
- Challenges
- Architectural models
- Fundamental models
Synchronous vs. Asynchronous Systems

- Two limiting factors
  - Communication performance
  - Lack of a single global notion of time

- A **synchronous system** is one with the following bounds defined:
  - Time to execute each step of a process has known upper and lower bounds
  - Each message transmitted over a channel is received within a known bounded time
  - Each process has a local clock whose drift rate from real time has a known bound

- An **asynchronous system** is one in which there are no bounds on processing execution speeds, message transmission delays, and clock drift rates.
Synchrony vs. Asynchrony

<table>
<thead>
<tr>
<th>Synchronous clocks</th>
<th>Physical clocks are synchronized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous processes</td>
<td>Lock-step synchrony</td>
</tr>
<tr>
<td>Synchronous channels</td>
<td>Bounded delay</td>
</tr>
<tr>
<td>Synchronous message-order</td>
<td>First-in first-out channels</td>
</tr>
<tr>
<td>Synchronous communication</td>
<td>Communication via <em>handshaking</em></td>
</tr>
</tbody>
</table>

- Any constraint defines some form of synchrony
Common Failure Types

- Crash failure = the process halts
  - a form of “nice” failure. In a synchronous system, it can be detected using timeout, but in an asynchronous system, crash detection becomes tricky.

- Omission failure: Message lost in transit
  - e.g., buffer overflow, software errors at the sender or the receiver, link errors, collisions at the MAC layer

- Byzantine failure
  - any type of erroneous behavior, including malicious ones
  - Most difficult to deal with