

Distributed Systems

CS425/ECE428

Radhika Mittal

Today's agenda

- Course overview
- Logistics
- Distributed System Model (if time)
 - Chapter 2.4 (except 2.4.3), parts of Chapter 2.3

Today's agenda

- **Course overview**
- Logistics
- Distributed System Model (if time)
 - Chapter 2.4 (except 2.4.3), parts of Chapter 2.3

Examples of distributed systems

- World Wide Web
- A cluster of nodes on the cloud (AWS, Azure, GCP)
- Multi-player games
- BitTorrent
- Online banking
-

What is a distributed system?

Hardware or software **components** located at **networked** computers communicate or **coordinate** their actions only by **passing messages**.

- *Your textbook*
(Coulouris, Dollimore, Kindberg, Blair)

What is a distributed system?

A collection of **autonomous computing elements**, connected by a **network**, which appear to its users as a **single coherent system**.

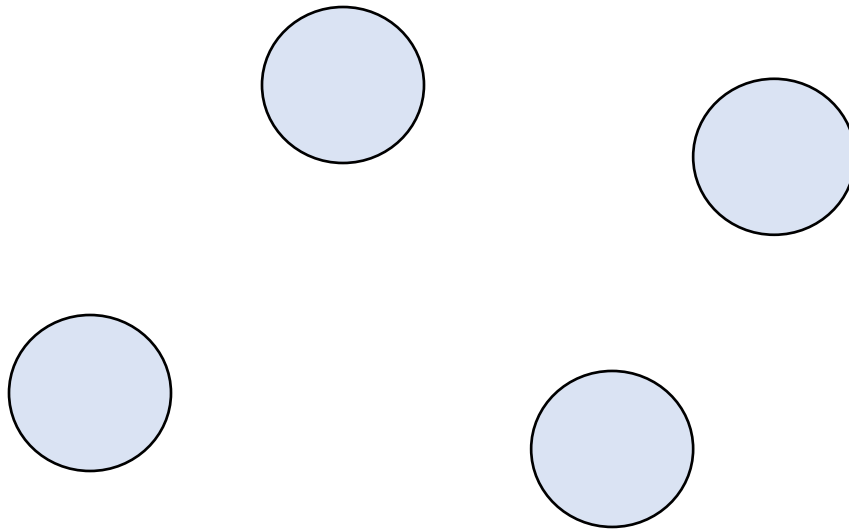
- *Steen and Tanenbaum*

What is a distributed system?

A system in which **components** located on **networked** computers communicate and **coordinate** their actions by **passing messages**. The components interact with each other in order to achieve a **common goal**.

- *Wikipedia*

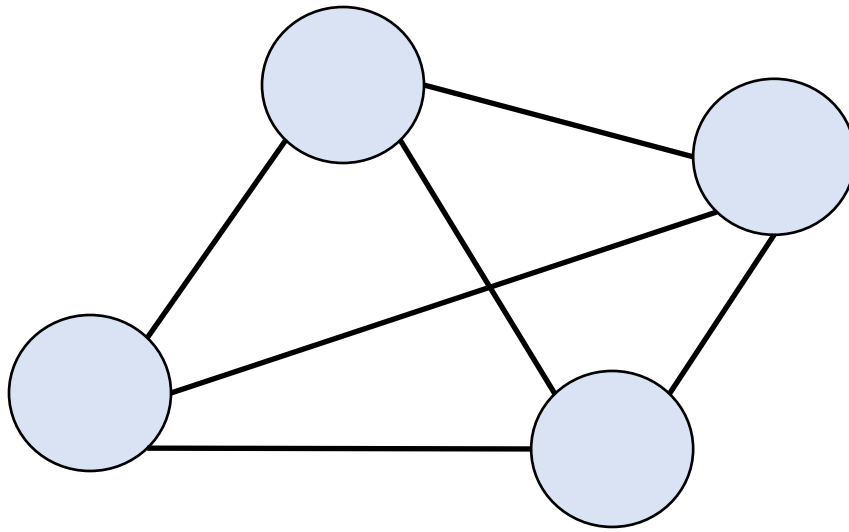
What is a distributed system?



Independent components or elements

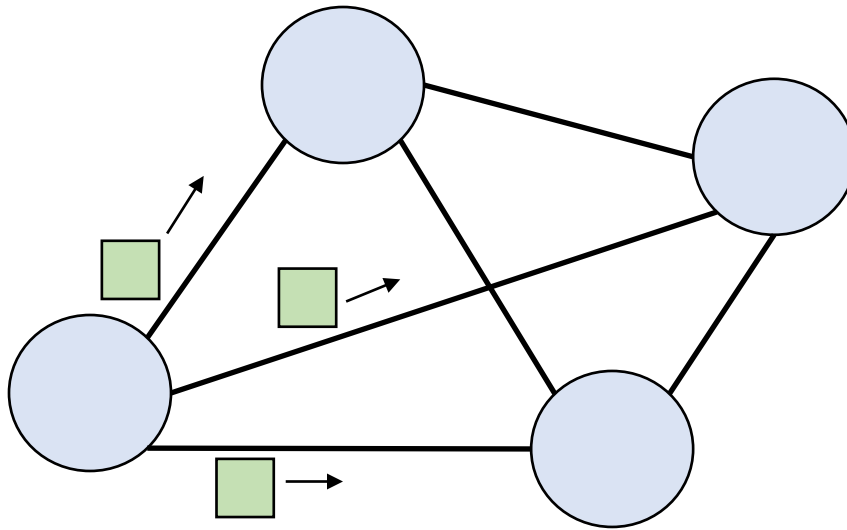
(software processes or any piece of hardware used to run a process, store data, etc)

What is a distributed system?



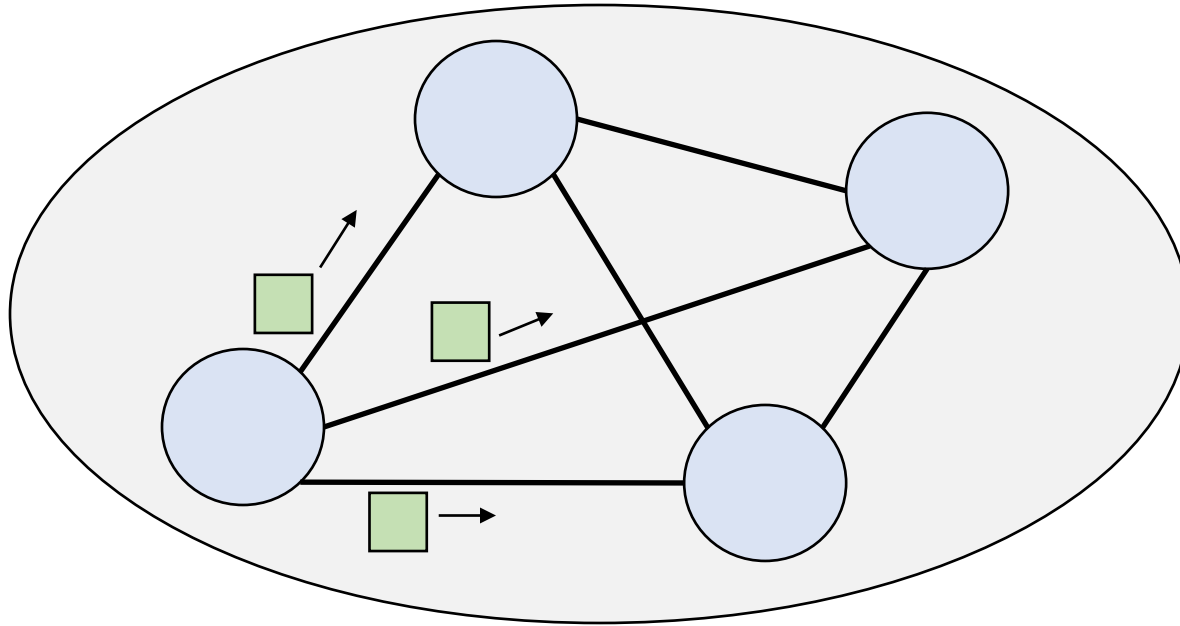
Independent components or elements that are **connected** by a network.

What is a distributed system?



Independent components or elements that are **connected by a network** and communicate by **passing messages**.

What is a distributed system?



Independent components or elements that are **connected by a network** and communicate by **passing messages** to achieve a common goal, appearing as a single coherent system.

What is a distributed system?

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

- *Leslie Lamport*

Why distributed systems?

- Nature of the application
 - *Multiplayer games, P2P file sharing, client requesting a service.*
- Availability despite unreliable components
 - *A service shouldn't fail when one computer does.*
- Conquer geographic separation
 - *A web request in India is faster served by a server in India than by a server in US.*
- Scale up capacity
 - *More CPU cycles, more memory, more storage, etc.*
- Customize computers for specific tasks
 - *E.g. for storage, email, backup.*

Example: scaling up Facebook

- 2004: Facebook started on a single server
 - Web server front end to assemble each user's page.
 - Database to store posts, friend lists, etc.
- 2008: 100M users
- 2010: 500M users
- 2012: 1B users
- 2019: 2.5B users

How do we scale up?

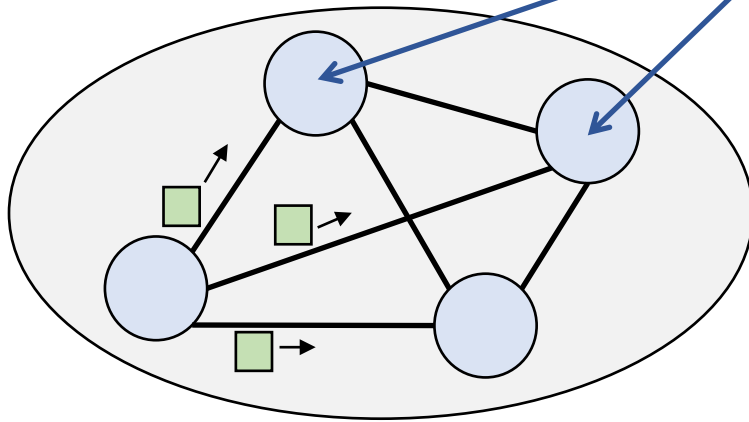
Example: scaling up Facebook

- One server running both webserver and DB
- Two servers: one for webserver, and one for DB
 - *System is offline 2x as often!*
- Server pair for each social community
 - *E.g., school or college*
 - *What if server fails?*
 - *What if friends cross servers?*

Example: scaling up Facebook

- Scalable number of front-end web servers.
 - Stateless: if crash can reconnect user to another server.
 - Use various policies to map users to front-ends.
- Scalable number of back-end database servers.
 - Run carefully designed distributed systems code.
 - If crash, system remains available.

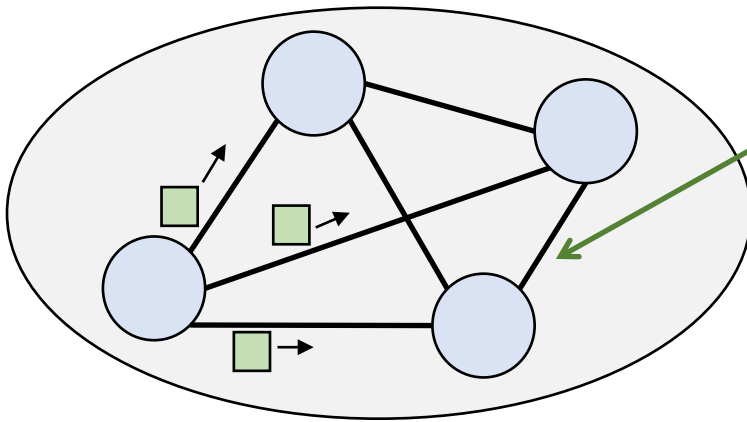
Challenging properties



Multiple computers

- Concurrent execution.
- Independent failure.
- Autonomous administration.
- Heterogeneous.
- Large numbers.

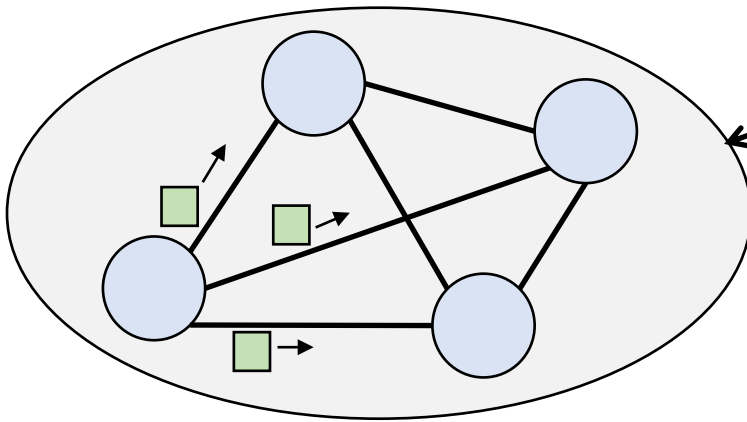
Challenging properties



Networked communication

- Asynchronous
- Unreliable
- Insecure

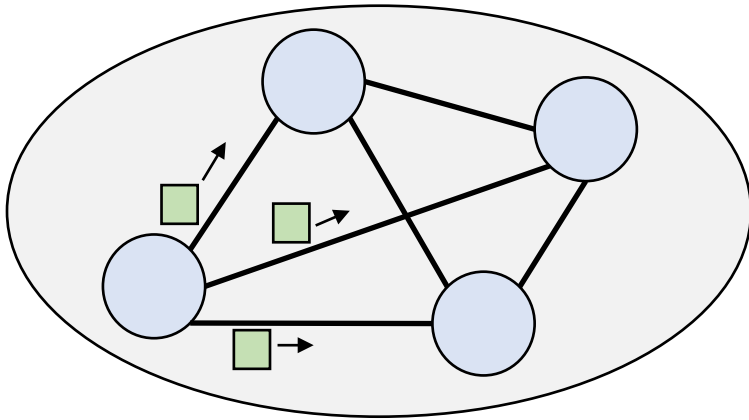
Challenging properties



Common goal

- Consistency
- Transparency

Challenging properties



Multiple computers

- Concurrent execution.
- Independent failure.
- Autonomous administration.
- Heterogeneous.
- Large numbers.

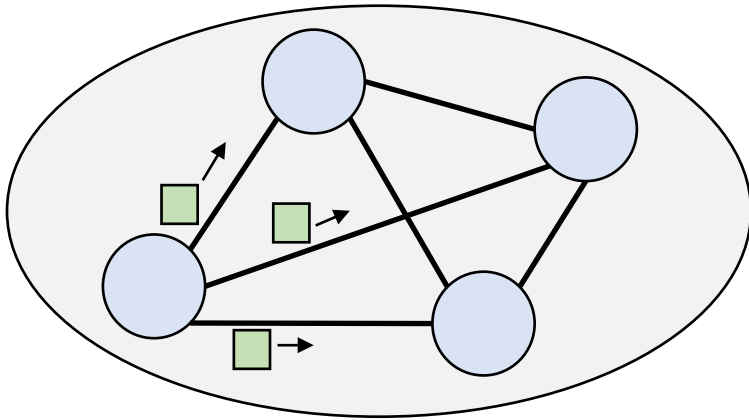
Networked communication

- Asynchronous
- Unreliable
- Insecure

Common goal

- Consistency
- Transparency

Challenging properties



Multiple computers

- Concurrent execution.
- Independent failure.
- Autonomous administration.
- Heterogeneous.
- Large numbers.

Networked communication

- Asynchronous
- Unreliable
- Insecure

Common goal

- Consistency
- Transparency

Rest of the course

- **Distributed system concepts and algorithms**
 - How can failures be detected?
 - How do we reason about timing and event ordering?
 - How do concurrent processes share a common resource?
 - How do they elect a “leader” process to do a special task?
 - How do they agree on a value? Can we always get them to agree?
 - How to handle distributed concurrent transactions?
 -
- **Real-world case studies**
 - Distributed key-value stores
 - Distributed file servers
 - Blockchains
 - ...

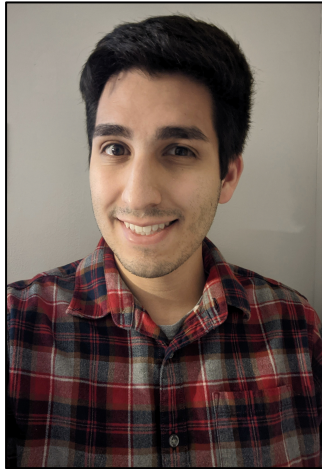
Today's agenda

- Course overview
- **Logistics**
- Distributed System Model (if time)
 - Chapter 2.4 (except 2.4.3), parts of Chapter 2.3

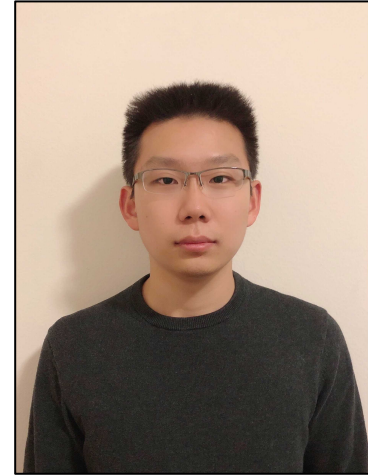
Course Staff



Radhika Mittal



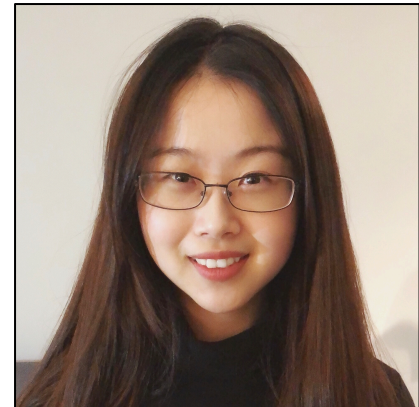
Federico
Cifuentes-Urtubey



Dayue Bai



Sanchit Vohra



Yitan Ze

Sources of information

- **Course website**

- <https://courses.grainger.illinois.edu/ece428/sp2021/>
 - <https://courses.grainger.illinois.edu/cs425/sp2021/> also works.
- Course Calendar
 - Time slots and Zoom links for office hours
- Homeworks, MPs
- Lecture schedule, readings, and slides

- **CampusWire**

- Announcements, questions, clarifications
- Link with access code has been emailed to you.
- Please reach out if you do not have access.

Books

- *Distributed Systems: Concepts and Design*, Coulouris et al., 5th edition.
 - Earlier editions may be acceptable.
 - Your responsibility to find correct reading sections.
- Other texts
 - *Distributed Systems: An Algorithmic Approach*, Ghosh
 - *Distributed Systems: Principles and Paradigms*, Tanenbaum & Steen
 - *Distributed Algorithms*, Lynch

Relevant Online Platforms

- CampusWire
 - Link with access code has been emailed to all registered students.
 - Reach out to Federico (netID: fc8) if you need access.
- Gradescope
 - Enrolled students have been added.
 - Reach out to Federico if you are currently enrolled and have not yet been added to Gradescope.
- CBTF for exam proctoring
 - More instructions to follow.

For students in different timezones

- Lecture videos will be uploaded to Echo360.
 - Please make sure you view them timely and regularly.
 - Ask clarifying questions on CampusWire or during office hours.

Grade components

- **Homeworks**

- 6 homeworks in total.
- Approx every 2 weeks.
- Will be submitted using Gradescope.
- Must be **typed** (hand-written diagrams are fine).
- Must be done **individually**.

Grade components

- Homeworks
- **MPs (only for 4 credit version)**
 - 4 mini projects.
 - First (warm-up) MP0 will be released on Friday!
 - Groups of up to 2
 - Need to fill up a form to activate VM clusters.
 - MP0, MP1, and MP3 can be in any language
 - Supported languages: Python, Go, C/C++, Java
 - MP2 must be implemented in Go.

Grade components

- Homeworks
- MPs (only for 4 credit version)
- Exams
 - Two midterms
 - Tentative dates and times:
 - March 8, Mon, 7-8:50pm
 - April 5, Mon, 7-8:50pm
 - Comprehensive final.
 - Exams will be scheduled and proctored via CBTF.

Grade components

- Homeworks
- MPs (only for 4 credit version)
- Exams
- CampusWire participation

Grade distribution

	3-credit	4-credit
Homework	33%	16% (drop 2 worst HWs)
Midterms	33%	25%
Final	33%	25%
MPs	N/A	33%
Participation	1%	1%

Switching between credits

- Multiple sections:
 - *ECE428/CS425 T3*
 - *ECE428/CS425 TU4*
 - *ECE428/CS425 T4*
- If you'd like to switch between 3 and 4 credits, try to get on the wait list for the desired section.
- If you are unable to make the switch, reach out to Heather Mihaly (hmihal2) after the drop deadline.

Integrity

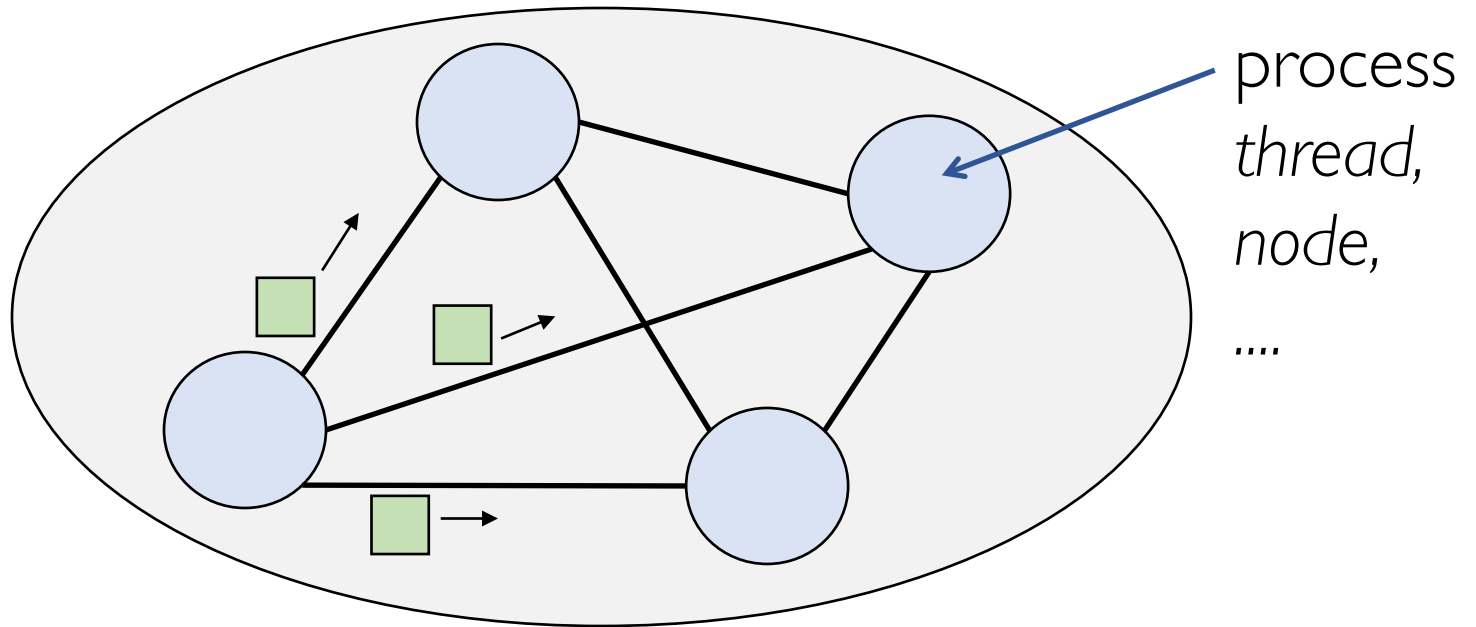
- Academic integrity violations have serious consequences.
 - Min: 0% on assignment
 - Max: expulsion
 - All cases are reported to CS, your college, and senate committee.
- **As students, it is your responsibility to uphold academic integrity.**
- Example of violations:
 - Sharing of code outside group.
 - Copying homework solutions (from colleagues, from previous years', from the web).
 - Collaborating in exams.
 -

Questions?

Today's agenda

- Course overview
- Logistics
- **Distributed System Model**
 - Chapter 2.4 (except 2.4.3), parts of Chapter 2.3

What is a distributed system?



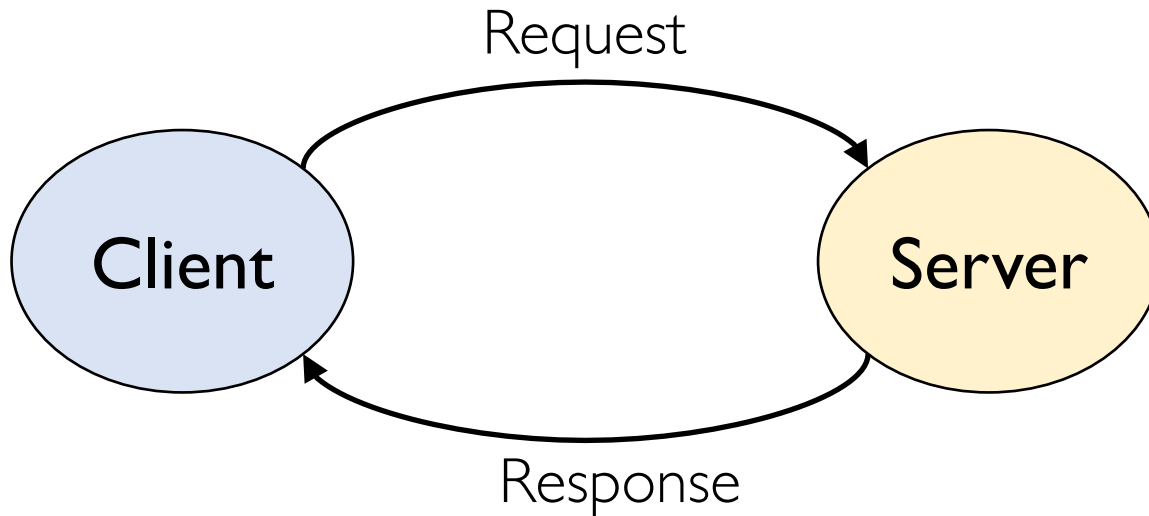
Independent components that are **connected by a network** and communicate by **passing messages** to achieve a common goal, appearing as **a single coherent system**.

Relationship between processes

- Two main categories:
 - Client-server
 - Peer-to-peer

Relationship between processes

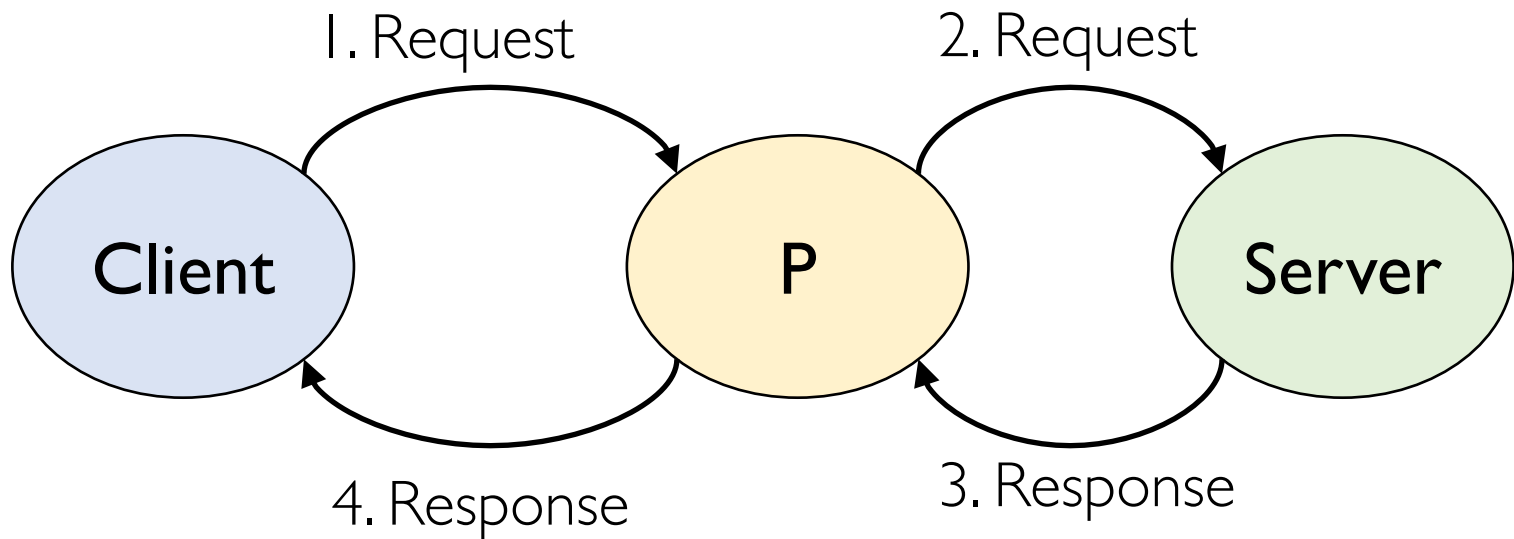
- Client-server



Clear difference in roles.

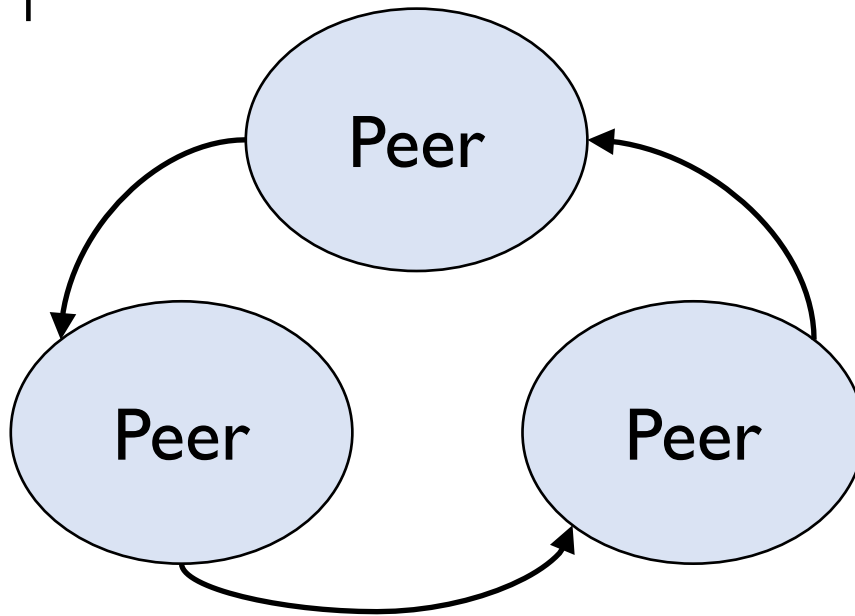
Relationship between processes

- Client-server



Relationship between processes

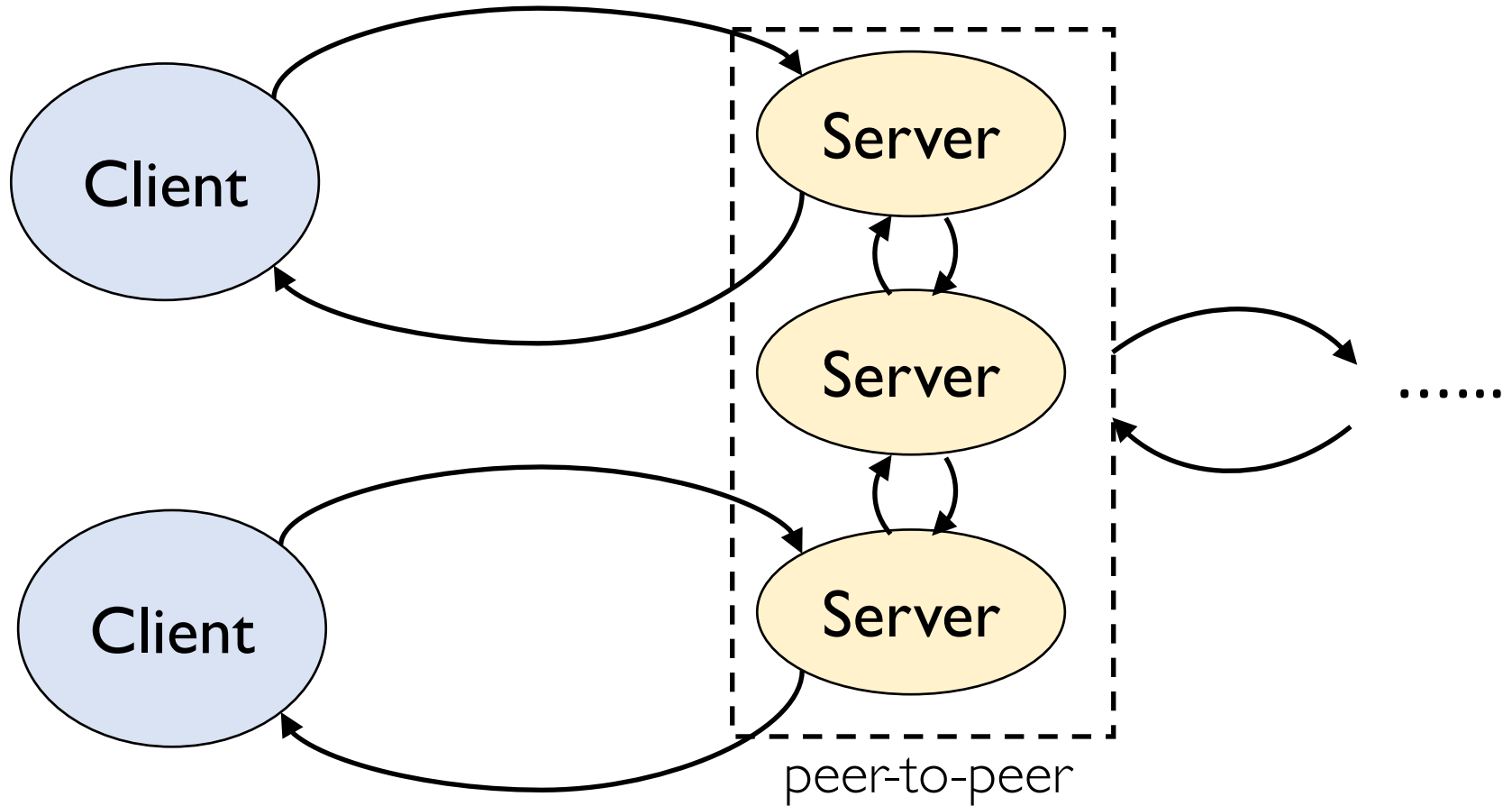
- Peer-to-peer



Similar roles.

Run the same program/algorithm.

Relationship between processes



Relationship between processes

- Two broad categories:
 - Client-server
 - Peer-to-peer

Distributed algorithm

- Algorithm on a single process
 - Sequence of steps taken to perform a computation.
 - *Steps are strictly sequential.*
- Distributed algorithm
 - Steps taken by each of the processes in the system (including transmission of messages).
 - *Different processes may execute their steps concurrently.*

Key aspects of a *distributed* system

- Processes must communicate with one another to coordinate actions. Communication time is variable.
- Different processes (on different computers) have different clocks!
- Processes and communication channels may fail.

Lecture Summary

- Distributed System
 - Multiple computers (or processes)
 - Networked communication
 - Common goal
- Distributed systems are fundamentally needed.
- They are challenging to build.
 - Variable communication time, clock drifts, failures.
- Course goals: concepts, designs, case studies

Acknowledgements

- Arvind Krishnamurthy
- Nikita Borisov

Questions?