Lecture 1: Introduction

CS 539 / ECE 526

Distributed Algorithms

Outline

- Basic course info
- Scope of class
- Brief intro to distributed computing
- Models of distributed computing

Basic Course Info

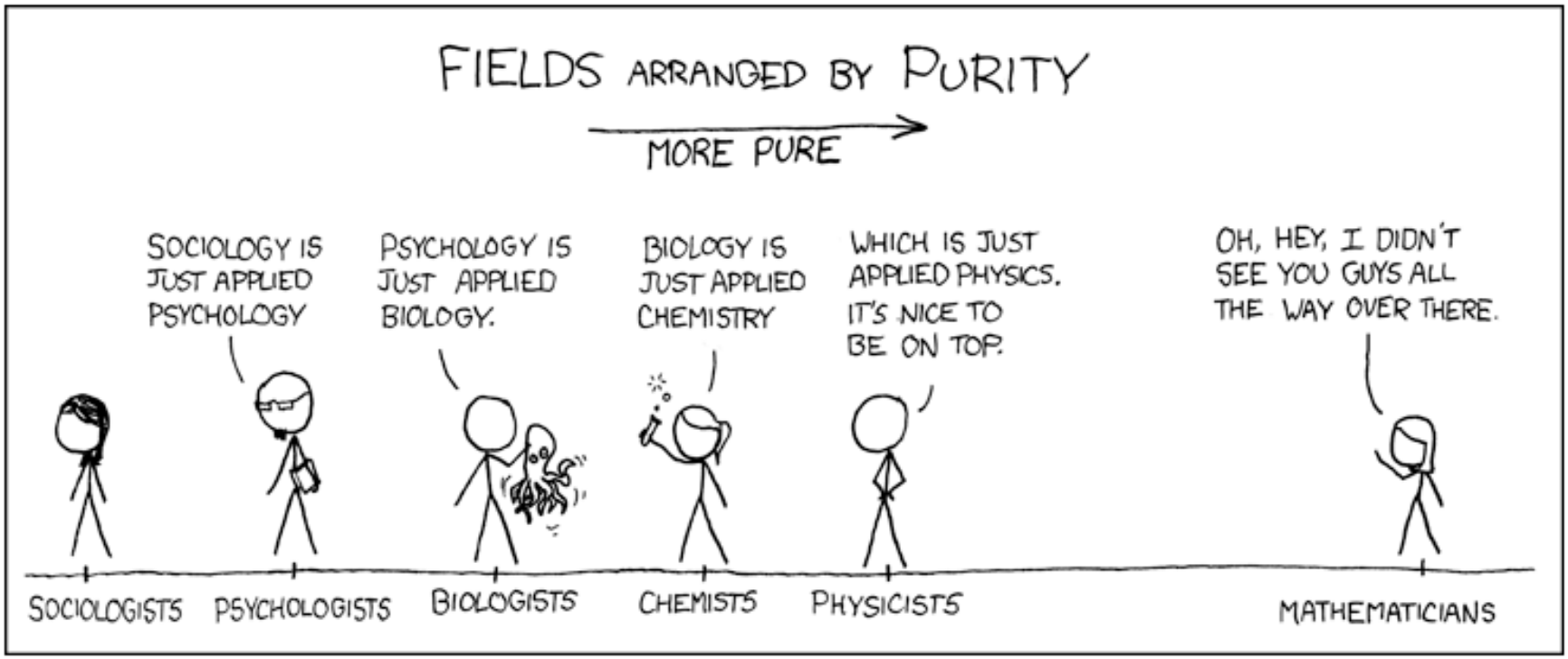
- Website: https://canvas.illinois.edu/courses/33367
- Lecture: Mon / Wed 9:30 -- 10:45 am
  Transportation building 114
- Office Hour: after Monday class Siebel 4312
- No required textbook. Useful resources on course website. Suggested reading optional.

Basic Course Info

- Prerequisite: None
  - A course on algorithms (374) is recommended
  - Distributed thinking can be unnatural at first
  - A course on distributed systems will be helpful

Scope of CS 539 / ECE 526

- A theory and algorithm class
  - Abstract/formal models and rigorous proofs



Scope of CS 539 / ECE 526

- A theory and algorithm class
  - Abstract/formal models and rigorous proofs
  - Study fundamental algorithms
    - Some algorithms have wide applications. But we will not discuss actual applications.
    - Others are of theoretical interests only at present. But no one knows the future.
  - Also study fundamental limits (negative results)

Scope of CS 539 / ECE 526

- A theory and algorithm class
  - Abstract/formal models and rigorous proofs
  - Study fundamental algorithms and their limits
- CS 425 / ECE 428 Distributed Systems
  - Deployed systems + relevant basic theory
  - Poll

Grading

- 60% five to six problem sets
  - Gradescope entry code: WV4GP5
  - Algorithm = pseudocode + correctness proof + efficiency analysis
- 15% midterm + 20% final, open book
- 5% class participation
  - Help with grading, scribing, asking/answering Qs

Disability-Related Accommodations

To obtain disability-related academic adjustments and/or auxiliary aids, students with disabilities must contact the course instructor and the as soon as possible. To ensure that disability-related concerns are properly addressed from the beginning, students with disabilities who require assistance to participate in this class should contact Disability Resources and Educational Services (DRES) and see the instructor as soon as possible. If you need accommodations for any sort of disability, please speak to me after class, or make an appointment to see me, or see me during my office hours. DRES provides students with academic accommodations, access, and support services. To contact DRES you may visit 1207 S. Oak St., Champaign, call 333-4603 (V/TDD), or e-mail a message to disability@uiuc.edu. http://www.disability.illinois.edu/.

Statement on Mental Health

Diminished mental health, including significant stress, mood changes, excessive worry, substance/alcohol abuse, or problems with eating and/or sleeping can interfere with optimal academic performance, social development, and emotional wellbeing. The University of Illinois offers a variety of confidential services including individual and group counseling, crisis intervention, psychiatric services, and specialized screenings at no additional cost. If you or someone you know experiences any of the above mental health concerns above, it is strongly encouraged to contact or visit any of the University's resources provided below. Getting help is a smart and courageous thing to do -- for yourself and for those who care about you.

- Counseling Center: 217-333-3704, 610 East John Street Champaign, IL 61820
- McKinley Health Center:217-333-2700, 1109 South Lincoln Avenue, Urbana, Illinois 61801

Academic Integrity Policy

Every student is expected to review and abide by the Academic Integrity Policy (Article 1, Part 4) of the University of Illinois at Urbana-Champaign Student Code http://studentcode.illinois.edu/. Ignorance is not an excuse. It is your responsibility to read the policy to avoid any misunderstanding. Do not hesitate to ask the instructor(s) if you are ever in doubt about what constitutes plagiarism, cheating, facilitating infractions, or any other breach of academic integrity.

We will scrutinize all exams and homework submissions for evidence of violations. All academic integrity violations will be reported to the university. Violations may result in a failing grade and probation.

Outline

- Basic course info
- Scope of class
- Brief intro to distributed computing
- Models of distributed computing

Distributed Computing

- What is distributed computing?
  - Computation carried out by multiple computers (processes/threads/nodes/parties/participants) that coordinate their actions
- Examples
  - Internet
  - Data centers
  - Multicore machines

Distributed Computing

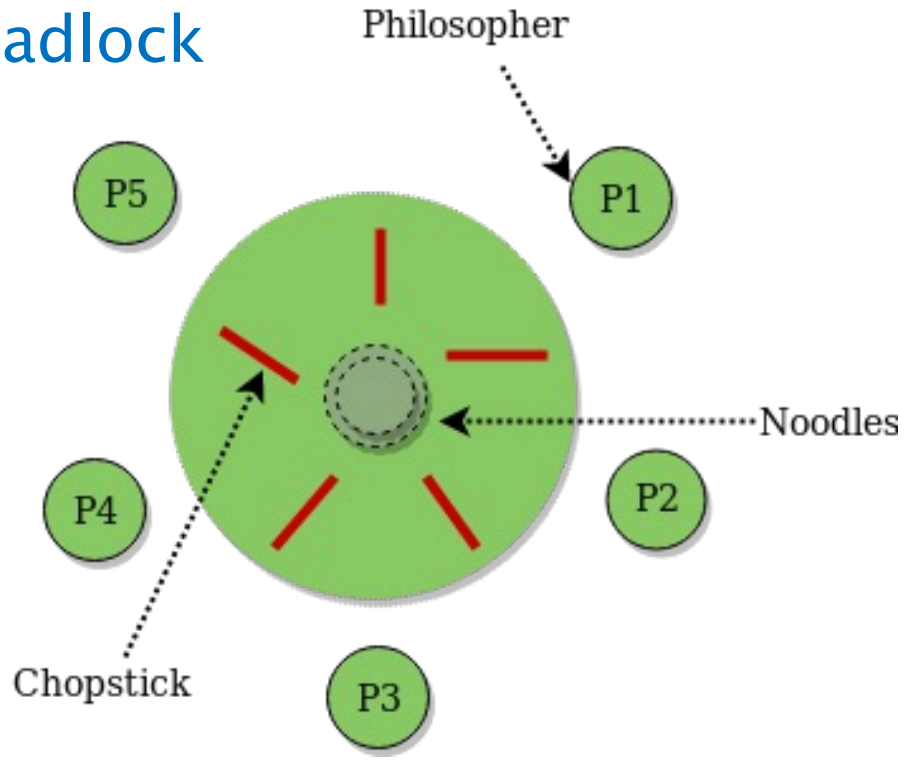
- What is distributed computing?
  - Computation carried out by multiple computers (processes/threads/nodes/parties/participants) that coordinate their actions
- Why distributed computing?
  1. Efficiency / scalability
  2. Reliability / trust

Distributed Computing

- What is distributed computing?
  - Computation carried out by multiple computers (processes/threads/nodes/parties/participants) that coordinate their actions
- Motivation: efficiency or fault tolerance
- Challenges: hard to reason about
  - Uncertainty and non-determinism
  - Often due to concurrency, delays, and failures

Example: Dining Philosophers

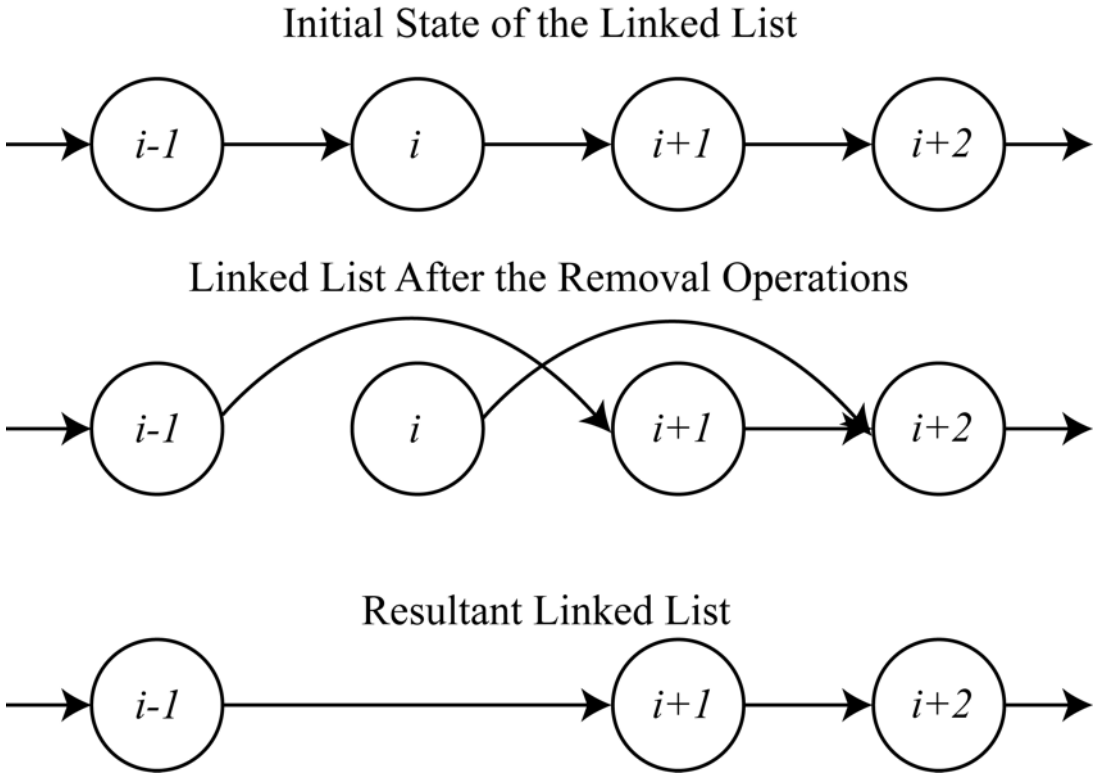
- Each philosopher: think → hungry → pick up chopsticks (how?) → eat → put down → think
  - Naïve solution can deadlock



https://www.geeksforgeeks.org/dining-philosopher-problem-using-semaphores/

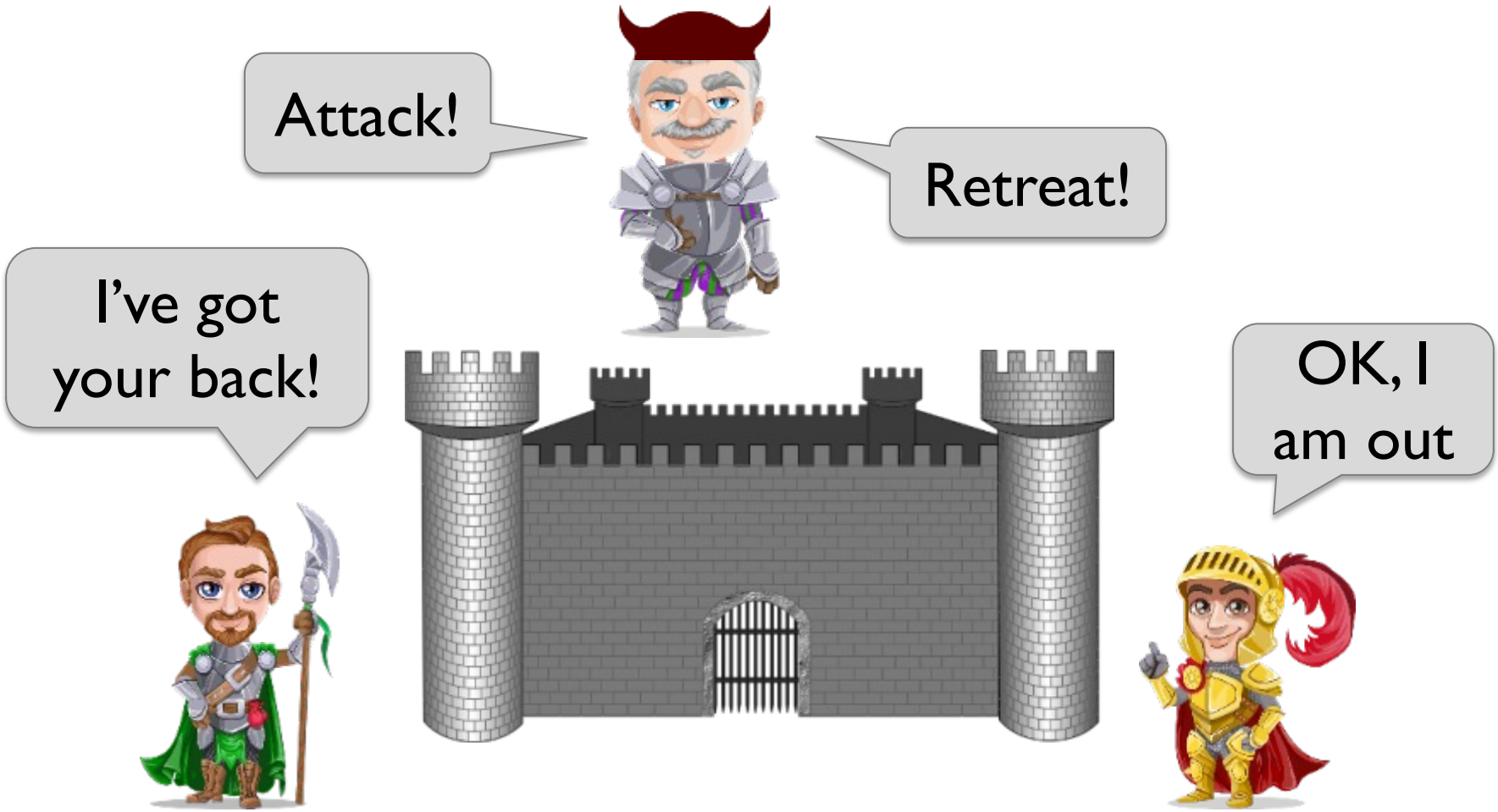
Example: Mutual Exclusion

- Two processes remove i and i+1 around same time. Result could be wrong if not careful.



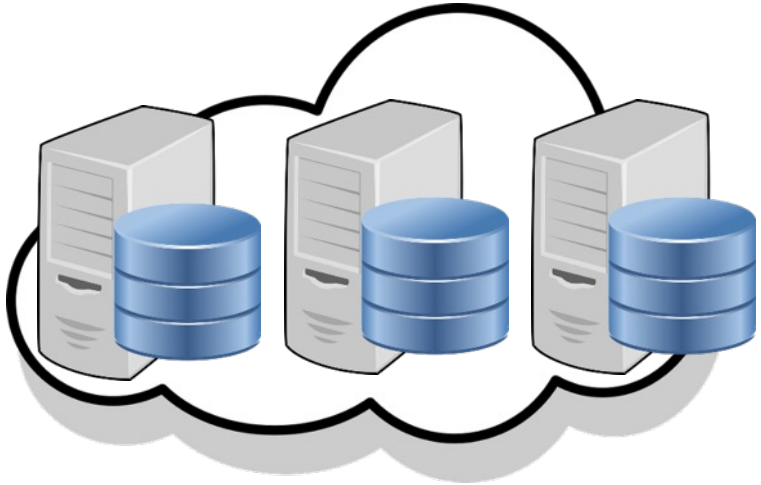
https://en.wikipedia.org/wiki/Mutual_exclusion

Example: Byzantine Generals



Example: Replication

- Consider any service
  - The server may fail
- Replicate the service
  - Need consensus
  - Despite some faulty servers



- Goal: provides an illusion of a single non-faulty server despite that some servers are faulty
  - Many like to call it blockchains these days

Outline

- Basic course info
- Scope of class
- Brief intro to distributed computing
- Models of distributed computing

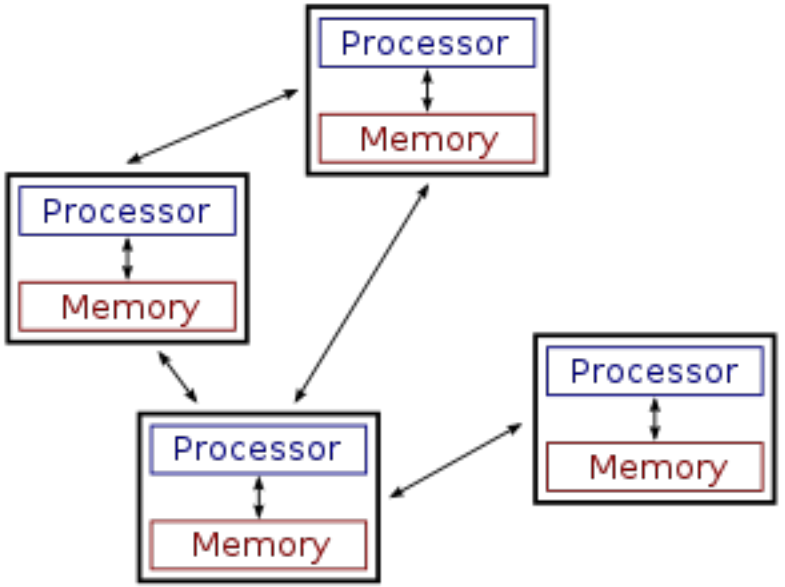
Model of Distributed Algorithms

- Communication model
- Timing model
- Fault model
- ……

Communication Model

- How do processes communicate?

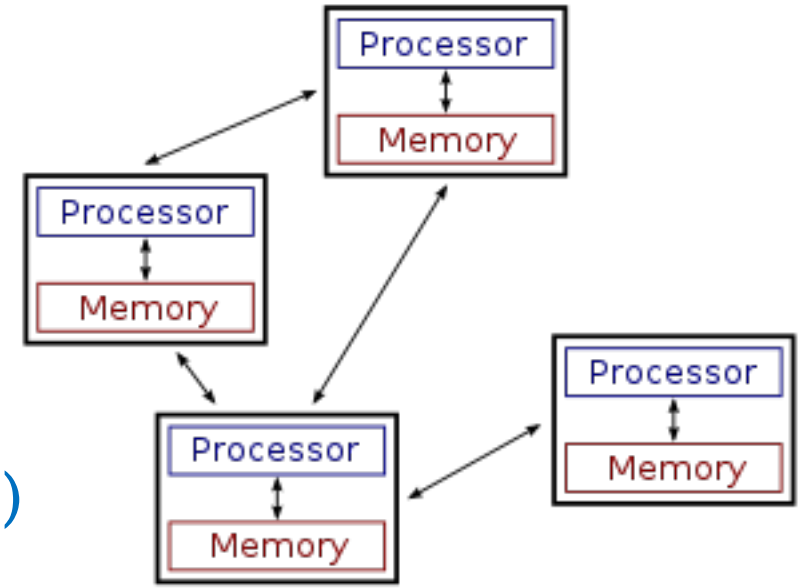
- Message passing
  - Proc i → out-buffer$_{i->j}$
    → in-buffer$_{i->j}$ → proc j



https://en.wikipedia.org/wiki/Distributed_computing

Message Passing

- Each link = outgoing buffer + incoming buffer
  - Not too hard to ensure in-order & reliable delivery
- One step:
  - Fetch from in-buffer
  - Process (local computation)
  - Put msgs in out-buffer (not always)

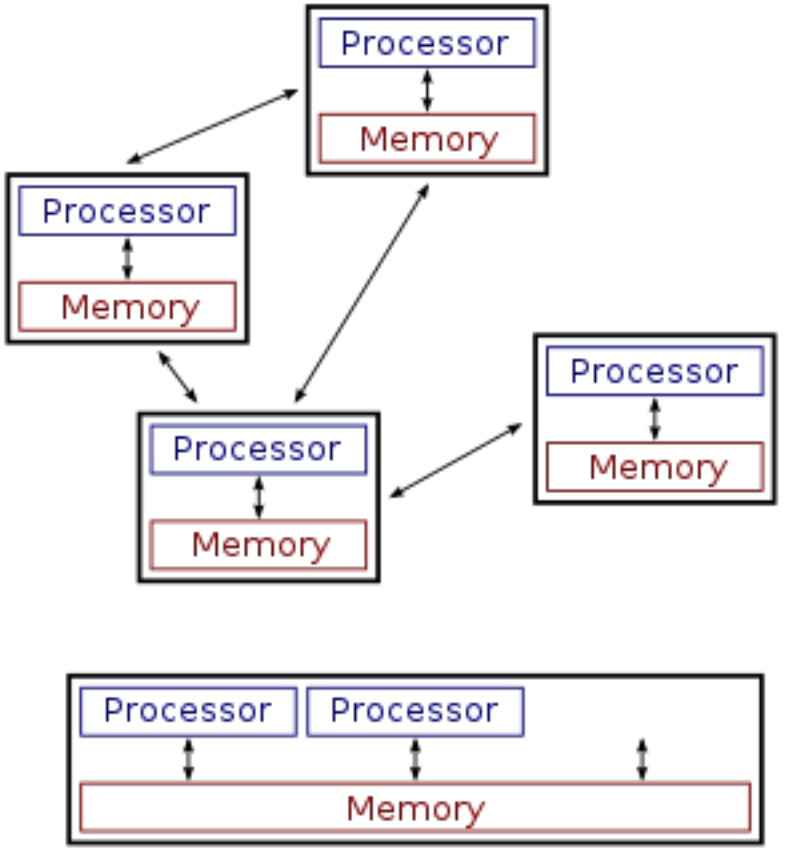


https://en.wikipedia.org/wiki/Distributed_computing

Communication Model

- How do processes communicate?

- Message passing
  - Proc i → out-buffer$_{i->j}$
    → in-buffer$_{i->j}$ → proc j
- Shared memory
  - Read/write shared var

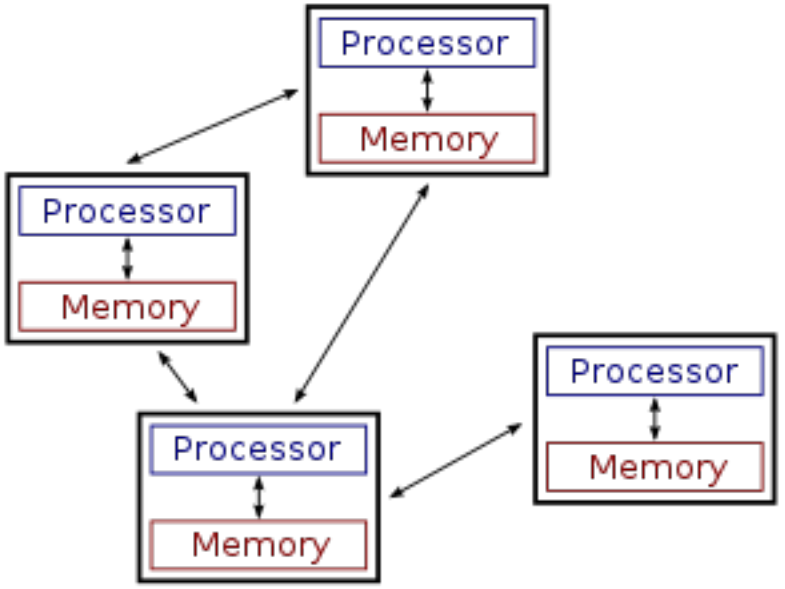


https://en.wikipedia.org/wiki/Distributed_computing

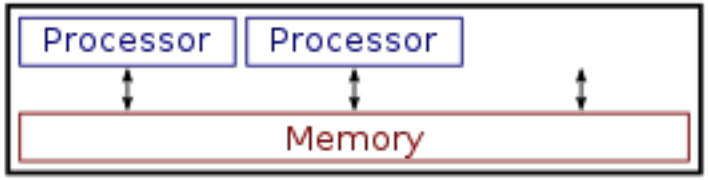
Communication Model

- How do processes communicate?

- Message passing
  - 1st half of semester



- Shared memory
  - 2nd half of semester



https://en.wikipedia.org/wiki/Distributed_computing

Communication Model

- Message passing is more fundamental than shared memory

- But shared memory is a more convenient (familiar) programming abstraction

Timing Model

- How long does an action (e.g., message arrival, memory read/write, computation step) take?
- Synchrony: each action completes in a predetermined bounded time
- Asynchrony: any action may take arbitrarily long time
- Crucial to note: no synchrony != asynchrony

Timing Model

- Lockstep synchrony
  - Perfectly synchronized rounds
  - A convenient ideal model in theory
  - Can be enforced under synchrony

- No lockstep synchrony != no synchrony != asynchrony

Fault Model

- How might a faulty process misbehave?

- Crash
- Arbitrarily, aka Byzantine or malicious

- Other faults: omission, crash-recovery, …

What is the “Right” Model?

- Depend on applications, goals, settings, …
- Exercise: “right” model for mutual exclusion?
- Exercise: “right” model for replication?

Efficiency Metrics

- Time complexity
  - For synchrony: # of rounds (aka round complexity)
  - Can be extended to asynchrony
- Communication complexity
  - Can be measured in # of msgs or bits
- Computation and space complexity
  - Not too different from non-distributed
  - Often ignored, will discuss when important

Summary

- Distributed computing motivations: efficiency or fault tolerance
- Problems: mutual exclusion, consensus, …
- Models: communication / timing / fault
- Efficiency metrics: round and communication, computation and space