DISTRIBUTED SYSTEMS

FALL 2022

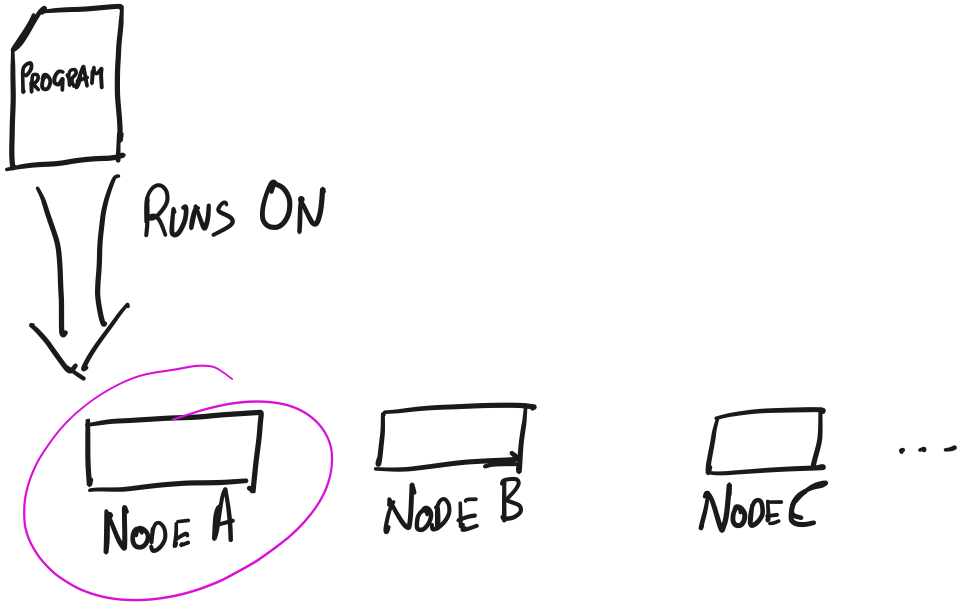
LECTURE 1

hello!

TODAY

- What & Why
- Mechanics (How)
- Expectations

What Are Distributed Systems



What Is A Node?

- Something That Can Execute Logic & Maintain State

↳ Thread ?

↳ Process ?

↳ Process/Thread On A Server?

↳

- Can Communicate With Other Nodes

Functionally Equivalent:
- ↳ Send Or Receive Messages (Focus Of This Class)
- ↳ Read Or Write To Shared Memory

Isn't This Just Concurrent/Parallel/Multi-Threaded Programming?

- Nodes Can Fail
  - ↳ Crash: Stop Making Progress
  - → Crash-Recover: Miss Some Steps, But Then Make Progress
  - → Byzantine: Exhibit Arbitrary Behavior
- Other Nodes Cannot Decide Whether A Node Has Failed Or Is Slow

→ Asynchronous Model

Asynchronous Model

- n Processes $(\Pi = \{\pi_0, \pi_1, \ldots \pi_n\})$
- Connected By A Communication Medium That

↳ CAN ARBITRARILY DELAY MESSAGES

[READS & WRITES FROM SHARED MEMORY TAKE ARBITRARY TIME]

→ CAN DROP MESSAGES

→ MUST BE FAIR

↳ RESENDING A MESSAGE ENOUGH TIME IS SUFFICIENT.

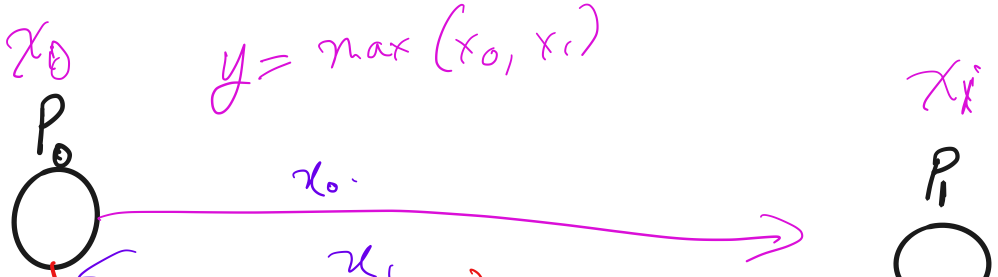
PROBLEM:

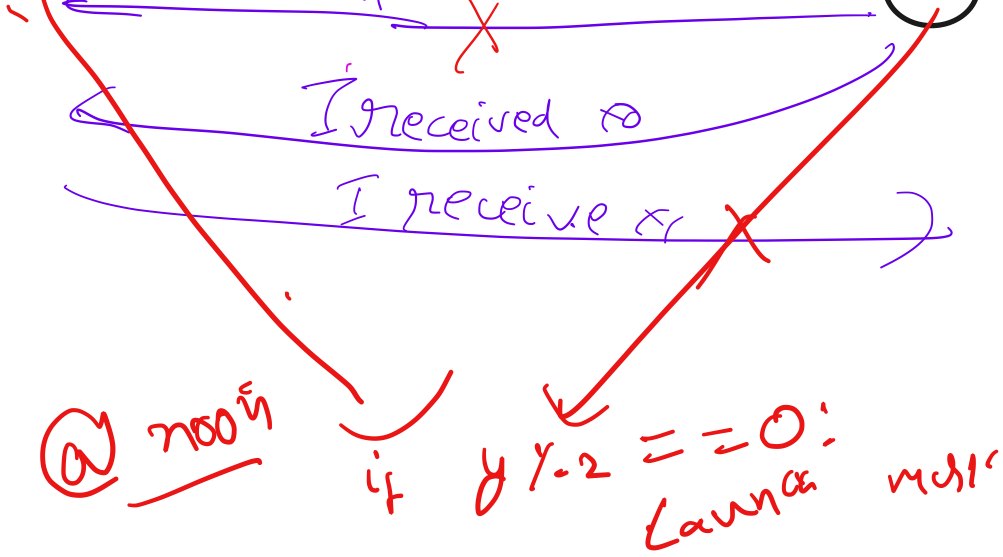
HOW TO DESIGN ALGORITHMS & SYSTEMS THAT

- WORK EVEN WHEN SOME PROCESSES FAIL
- IN THE ASYNCHRONOUS MODEL

↳ MANY BASIC TASKS ARE IMPOSSIBLE.

TWO GENERALS





WHY? (BEYOND INTELLECTUALLY INTERESTING)

- FAULT TOLERANCE
- SCALING RESOURCES & USERS

COURSE MECHANICS

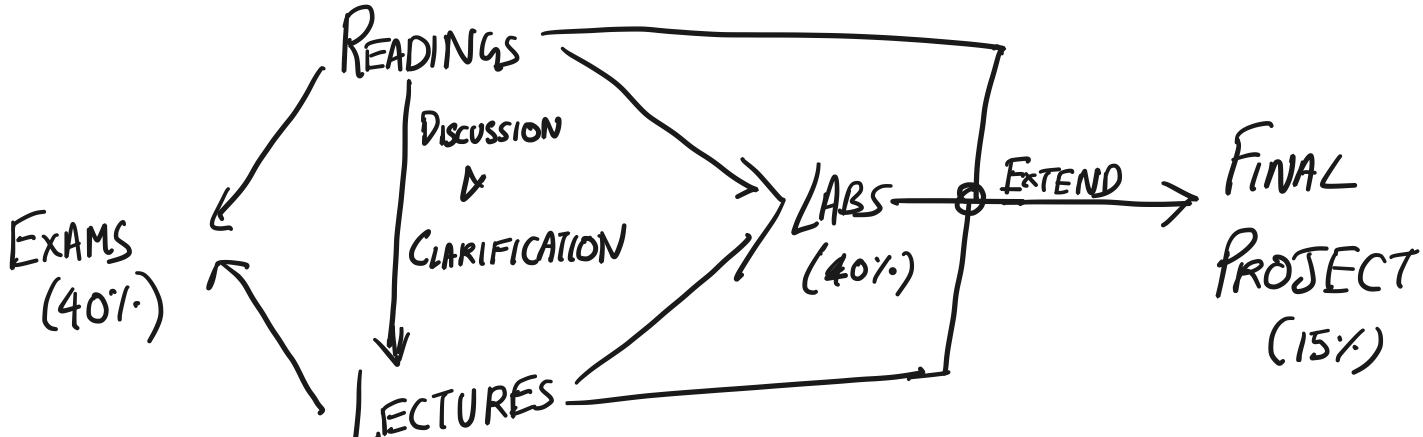
- PAPERS

↳ WHY?

→ EXPECTATION ABOUT DETAILS, PROOFS, ETC.

→ TIME COMMITMENT

EXPECTED STRUCTURE



FOCUS IN LECTURES

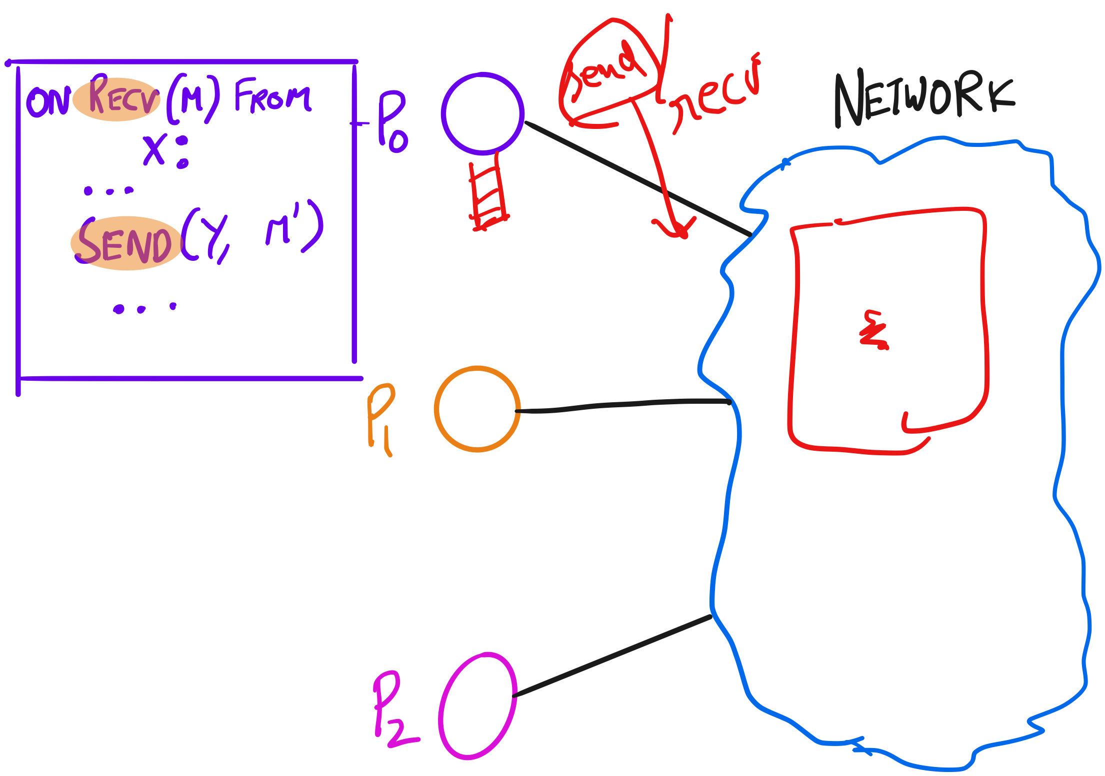
- DEFINE REQUIREMENTS & CORRECTNESS
- ASSUMPTIONS & INVARIANTS
- HOW PROTOCOLS MAINTAIN INVARIANTS
- THINGS NOT COVERED IN THE PAPER

LABS

ELIXIR?

Collaboration & Communication.

Message Passing & The Asynchronous Model



On Boot:
send(x_i, (i+1)%2)

On recv $x_{(i+1)\%2}$
from (i+1)%2
send(ack),
on timer ( )

FAIRNESS

If process P sends P' message M infinitely often
Then P' (if alive) receives M infinitely often.

$\therefore x \in \mathbb{N}$ _____

$\mathbb{N}^+$

_____

$P \xrightarrow[x]{\text{sends}} P'$

Why Assume & Why Realistic?