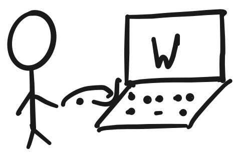
Hello! This is

Distributed Systems
Spring 2024

Today's Plan

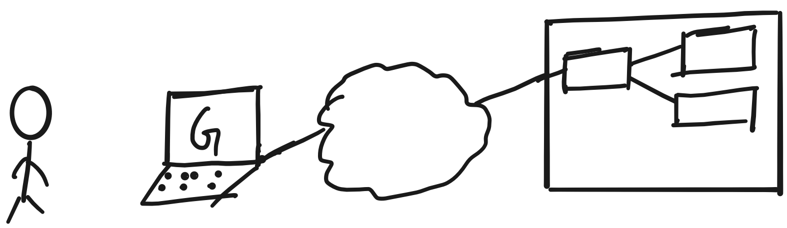
- WHAT are distributed systems
- Why
  - Useful
  - Study
- How
  - General
  - Course specific

What



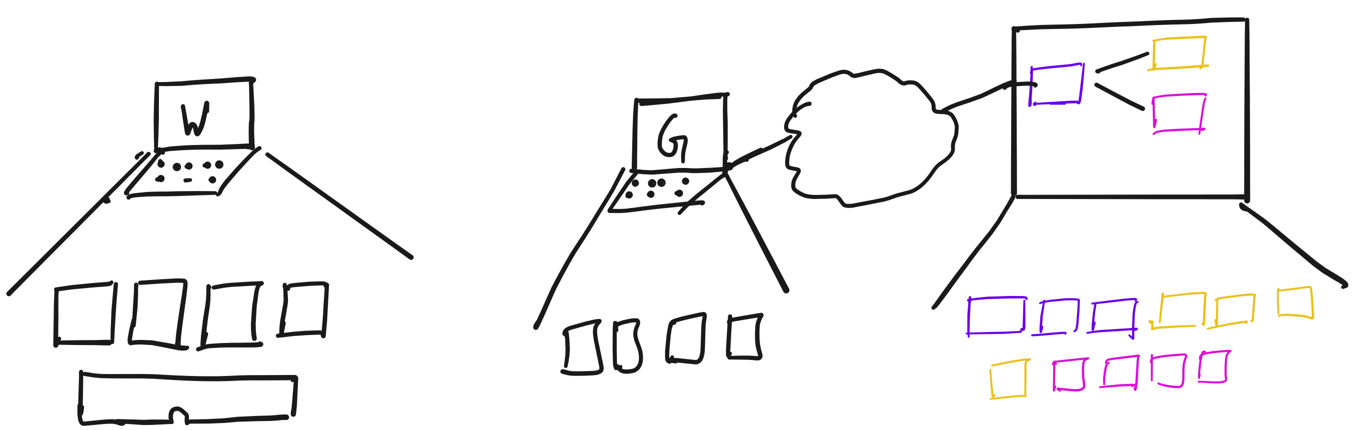
Contents of the document?

Contents of the document?



What changes

- Logic is concurrently executing on many machines



- Each of which can FAIL INDEPENDENTLY

Almost all of distributed systems is about reasoning about failures

What does failure mean

- CRASH (software or hardware)
- CRASH - RECOVER

  ...

- BYZANTINE (ARBITRARY BEHAVIOR)

WHY

Failures appear rare.

Why do all of this?

ORIGINALLY: Systems that needed to remain available

- ATC, Defence

INTERNET: ~1970 - to

Interactions between machines spread out GEOGRAPHICALLY & OWNED BY DIFFRENT ENTITIES

disconnect or fail without

↳ Likely to disconnect or fail without coordination

Large Clusters / Datacenters:

Failures for a single server are rare

MTBF 2 – 10 years

But at scale

Google reports

- ~1 net failure every 5 minutes
- ~1000 machine failures per DC per year
- ~1000+ disk failures
- ~ . . . .

More recently

Cloudflare

↳ 3x redundant power

Failed for ~3 days

Azure

↳ Several hours

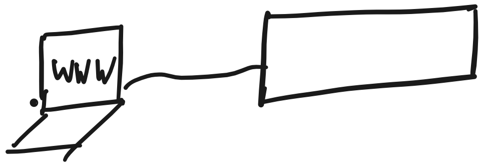
Bottom line: Consider enough machines,

something will fail

Our Focus:

- What problems can distributed systems solve (given failure & other ASSUMPTIONS)
- How? What algorithms (protocols) do they use.
- What happens when ASSUMPTIONS are violated.

Why study



- REACT
- Javascript
- ...

Deep Learning
- Training
- Inference

MICROSERVICE APPLICATION

CLOUD COMPUTING

Distributed systems

- Note, ABSTRACTIONS HIDE MOST OF THE COMPLEXITY

BUT

- Debug BAD PERFORMANCE
- Debug CORRECTNESS
- REASON ABOUT COST/PERF

} PEEK INSIDE ABSTRACTION

USEFUL TO UNDERSTAND HOW ABSTRACTIONS WORK & TRADE-OFFS THEY MAKE!

ALSO, (I THINK) THE PROBLEMS & ANALYSIS ARE FUN PUZZLES!

HOW TO STUDY

- FOR MOST OF US FAILURES ARE VANISHINGLY

RARE

- NOT SOMETHING WE ARE USED TO REASONING ABOUT

- HARD TO TEST
  - → Need to control for when the failure occurs
  - → Lots of ongoing work on testing behavior under failures

- ANALYZE BEHAVIOR ASSUMING SOME MODEL
  - Models simplify and abstract reality
  - Usually impose stricter limits
  - Hope

    CORRECT UNDER MODEL

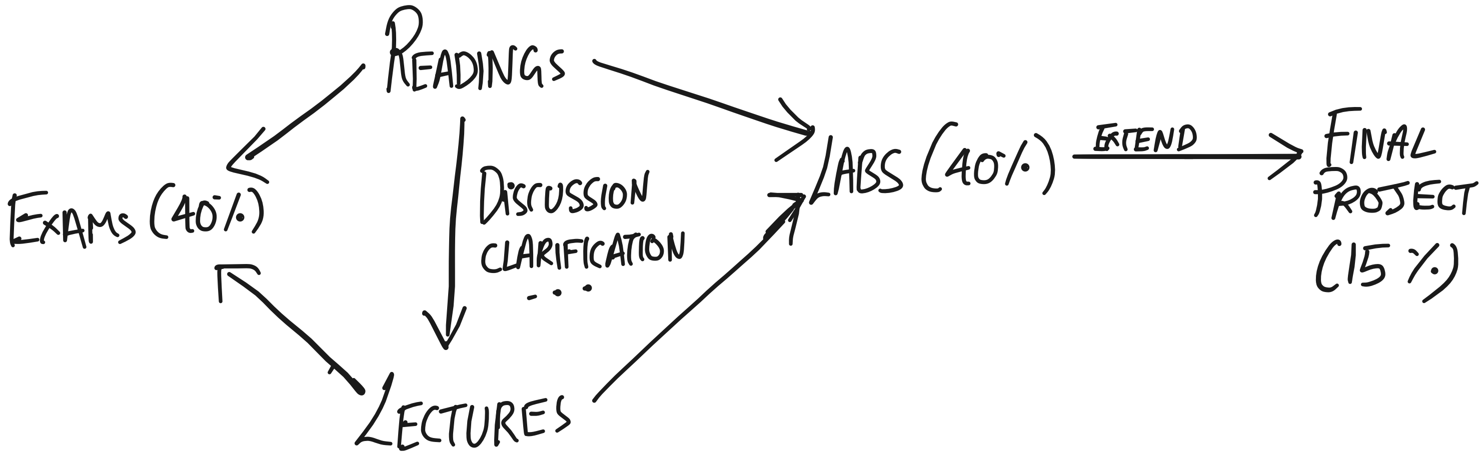
    $\Rightarrow$ CORRECT IN REALITY

- Analysis usually involves
    - Pen & paper
    - Proof assistants

COURSE MECHANICS

- PAPERS
    - MODELS: DISTRIBUTED SYSTEMS
      FAILURE MODELS
      CORRECTNESS CONDITIONS
    - WHY?
    - EXPECTATIONS

- LABS
    - ELIXIR
        ↳ Really a small subset
        → CODE MORE CLOSELY RESEMBLES WHAT IS IN THE PAPERS (MOSTLY)



COLLABORATION

COMMUNICATION

ME → YOU : Website, Campuswire

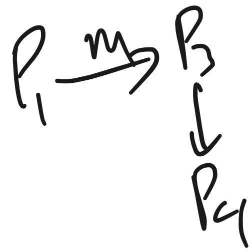
YOU → ME : Campuswire + EMAIL

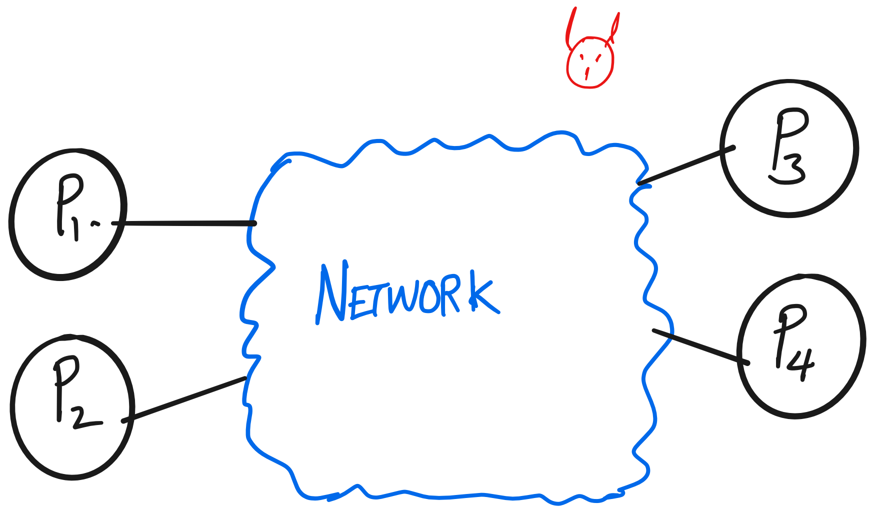
→ When possible: Please make posts public

Help others!

ASYNCHRONOUS MODEL + Message Passing

PROCESSES $\Pi = \{P_1, P_2, \ldots, P_n\}$





Message Passing

- PROCESSES COMMUNICATE BY SENDING AND RECEIVING MESSAGES

Asynchrony Assumption

- NO BOUNDS ON TIME TAKEN B/W SEND(m) & RECV(m)
- NO BOUNDS ON TIME TAKEN TO PROCESS A MESSAGE m

- No Global Clock

Consequences:

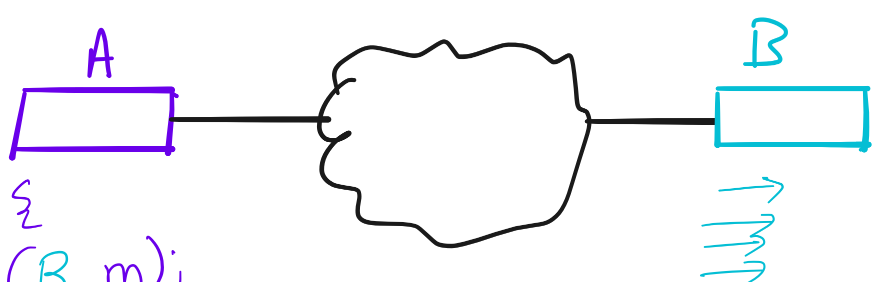
- Cannot decide whether process

p has failed.

Why?

Additional Assumptions

- Unreliable network: MESSAGES CAN BE DROPPED OR REORDERED
- FAIRNESS: Any message sent infinitely often must be received infinitely often

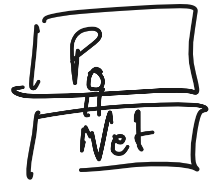



```
while(1) {
  send(B, m);
}
```

```
for (i=0 : 10) {
    send(B, m)
}
```

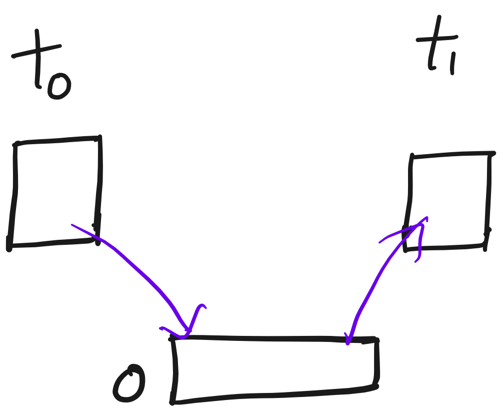
FAIRNESS ALLOWS US TO IGNORE UNRELIABLE NETWORK.

- THEORY:



- PRACTICE:

Going back to threads



Shared Memory

- HAS EQUIVALENT RESULTS.
- THIS CLASS LARGELY FOCUSES ON MESSAGE PASSING. COMMONLY ADOPTED MODEL.