Distributed Systems

Lecture 2

Where we are

- What? Algorithms/protocols that run on multiple nodes can communicate over a network
- Focus on message passing → nodes communicate by sending & receiving messages
- Assume the asynchronous model → messages can be dropped or arbitrarily delayed → fairness
- Why do any of this? → Fault tolerance
- Scaling

- Challenge: Distinguishing between Failed & Slow Nodes
  \[ \Rightarrow \text{Node might be slow or messages delayed} \]

Today

- Tools for reasoning about distributed algorithms & protocols

  \[ \Rightarrow \text{Events & how to order them} \]

  \[ \Rightarrow \text{Traces} \]

  \[ \Rightarrow \text{Correctness properties: Safety & Liveness} \]

- A 'simple' protocol for capturing the state of a distributed system

  \[ \downarrow \]

  \[ \text{To check correctness} \]

Specifying Algorithms

\[ x, y \in \mathbb{Z} \]

\[ \text{SWAP}(x, y): \]

1. \[ x = x \land y \]
2. \[ y = x \land y \]
3. \[ x = x \land y \]

\[ x = y_{in}, y = x_{in} \]

for thinking about algorithms as sequential
Generally used to thinking about against 2 simple steps.

For long running systems, also common to think about correctness in terms of order in which things occur.

Eg. Linearizability (2 weeks from now)

\[ \text{WRITE}(x, 1) <_{hb} \text{READ}(x) \]

\[ \Rightarrow \text{READ}(x) \text{ returns } 1 \text{ or newer WRITE} \]

Today's Question:

In a distributed system, how do we decide whether \( \text{EVENT } e_0 \text{ happens - before } e \)?

Events - Things A Process Does

From last class
- Receives
- Sends
- Why Not Compute?

Environmental
- Process Failures
- Process Launches

Causality

Some Events Have A Natural Notion Of Order

On \( \text{Recv}(m) \) From \( X \):

- Compute + Updates
  - Send \((y, m')_0\)
  - Send \((z, n'_1)\)
  - ...
Producing a Trace in total order

Correctness conditions for distributed protocol (or algorithm) limit what traces are produced (or are admissible).

Send Pong only when Pong is received:

\[ P_0, \text{send}(p, \text{Pong}) \quad P_i, \text{recv}(P_0, \text{Pong}) \quad P_i, \text{send}(p, \text{Pong}) \]
Types of Restrictions/Correctness Properties

- Safety
  - Some behavior (seq. of events) NEVER occurs

Liveness

- Some property can/will eventually hold

"Eventually all replicas agree on the set of keys & values"
UNIFORM / ABSOLUTE ZIVENESS

L? GIVEN A TRACE PREFIX, WHAT STEPS NEED TO BE TAKEN TO MEET PROPERTY

• GENERAL ADMISSION
  FOR ANY PREFIX P, CAN FIND SUFFIX Sₚ
  S.T. CONDITION HOLDS FOR Pₛₚ

• UNIFORM ADMISSION
  THERE IS A SUFFIX S, S.T. FOR ANY PREFIX P
  CONDITION HOLDS FOR Pₛ

• ABSOLUTE
  GIVEN A TRACE T FOR WHICH CONDITION HOLDS &
  ANY ADMISSION PREFIX P
  CONDITION HOLDS FOR Pₜ

HARDER/EASIER ?
CHANDY-LAMPORT

- Goal & Properties?
  - Compute a global state

Assumptions
- Ordered channels + reliable
- No failure