

CPSC 416 – Distributed Systems

Lecture 04 Global state: The problem

UBC Vancouver

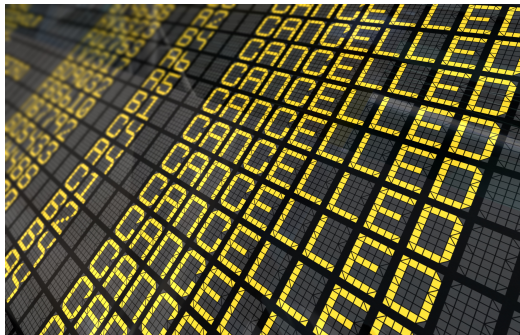
2026-01-15

Failure-first question

How can you know “what the system is doing” when there is no single moment you can observe?

Southwest Airlines meltdown (December 2022)

- 10-day disruption (Dec 21–31)
- 16,700+ cancelled flights
- \$821 million in losses
- Root cause: 20+ year old scheduling system
- Manual processes to reassign crew
- System couldn't track global state



The core problem

Southwest couldn't answer: *"Where is everyone right now?"*

- Pilots and flight attendants in wrong cities
- No consistent view of crew locations
- Phone calls to manually track people
- By the time you get answers, they've moved

This is a **global state** problem in a distributed system.

Why do we care about global state?

Debugging: “Is there a deadlock right now?”

Checkpointing: “Can we restart from a consistent point?”

Garbage collection: “Is this object referenced anywhere?”

Termination detection: “Has the computation finished?”

Resource tracking: “Where are all the crew members?”

All require answering: *What is the state of the entire system?*

The problem: no global observer

- No shared memory — each process has only local state
- No global clock — we just covered this in Lecture 03
- Messages in flight are *part of the state*
- Can't “stop the world” — processes keep running

If you query each process, by the time you get all answers, they've moved on.

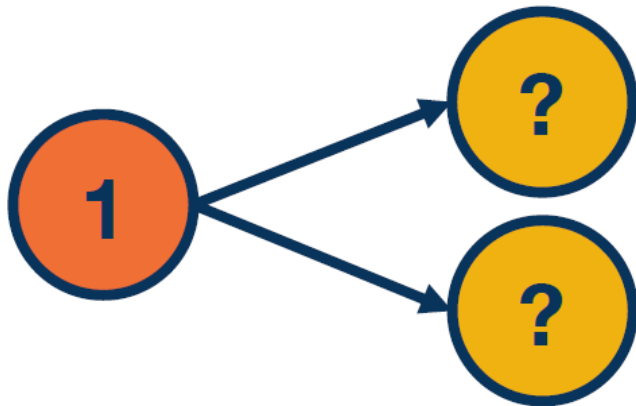
Non-determinism in distributed systems

Decoupled processes can perform operations in arbitrary order.



Even with identical code, different runs produce different interleavings.

The network makes it worse



Messages can be delayed, reordered, or arrive at different times.
Event order is **not** determined by send order.

Global state = process states + channel states

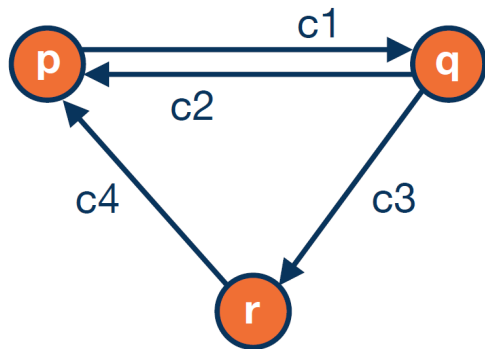
- **Process state:** local variables, program counter, etc.
- **Channel state:** messages sent but not yet received

A **global state** is a snapshot of all process states *plus* all channel states.

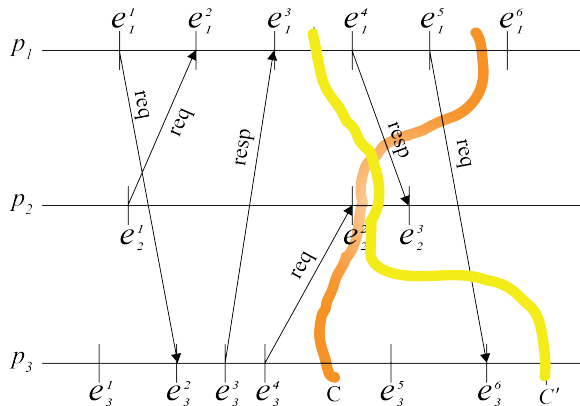
But: different observation orders give different “snapshots.”

Formalizing the model

- **Processes:** independent actors
- **Channels:** directed edges
- Assume: FIFO, reliable
- State changes via:
 - ▶ Local computation
 - ▶ Send message
 - ▶ Receive message



A cut through the execution



A **cut** = one local state per process (a “slice” through time).

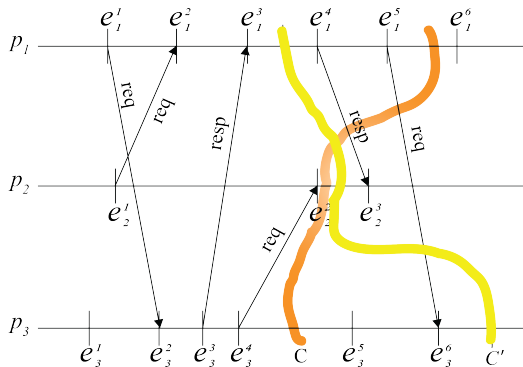
Consistent vs. inconsistent cuts

Cut C (orange): consistent

- Every message received was also sent (in the cut)
- Corresponds to a *possible* execution state

Cut C' (yellow): inconsistent

- Shows a receive without the corresponding send
- “Message from the future” — impossible state



Formal definition: consistent cut

A cut C is **consistent** if and only if:

For every message m : if the *receive* of m is in C , then the *send* of m is also in C .

Equivalently: the cut respects **causality** (happens-before from Lecture 03).

An inconsistent cut shows effects without causes — **time travel**.

The naive approach: external observer

- Stop the entire system
- Query every process for its state
- Query every channel for in-flight messages
- Resume the system

Problems:

- Can't actually stop a distributed system atomically
- Even if you could, stopping is expensive
- We want snapshots *during* normal operation

Challenges for distributed snapshots

- ❶ No external observer who can freeze everyone
- ❷ No instantaneous communication
- ❸ No global clock to coordinate “take snapshot now”
- ❹ Network delays are variable and unpredictable
- ❺ Processes keep executing during the snapshot

How can we possibly get a consistent picture?

The question we'll answer next time

Can we compute a consistent cut *without* stopping the system?

- Without a global clock
- Without freezing processes
- While messages are still in flight
- And still guarantee the cut is consistent

Spoiler: Yes. Chandy and Lamport (1985) showed how.

Logistics

- Readings + quiz (Week 02 on Canvas Readings page)
- Self-assessment opens at 13:00, due next class 11:30 (Pacific)
- Capstone Weekly Report 1 is live (keeps the exploration cadence)

Recommended reading for next time:

- Chandy & Lamport, “Distributed Snapshots” (1985)