Welcome to CSCI 7000-001 Lec 2 (Jan 19)!

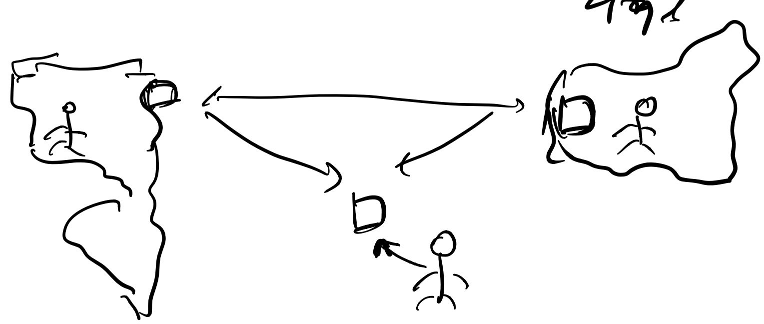
Today's topics

1. Fundamental uncertainty in Distributed programming
2. Time and causality

Distributed System: A system of interconnected computational nodes coordinating to execute a computational task.

→ Why Distributed System

* Redundancy / reliability. No SPOF.
* Availability: Service / application has to be readily available to most of the users most of the times 99.9999 % of time
  4 9's



* Scalability

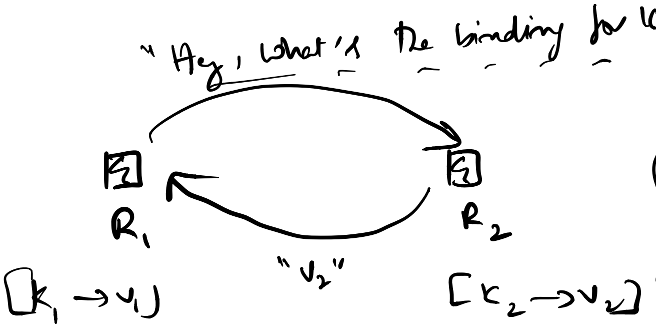
→ Partial failures : What are partial failures?

Failure scenarios in a distributed system:

- * Network links might fail.
- * Machines can crash → Hard drive crashes
- * Network congestion → messages are dropped.
- * Machines / Network can be very slow
- * Machines may misbehave
- * Messages may be corrupted

(Machines may misbehave / Messages may be corrupted) → Byzantine failures

---



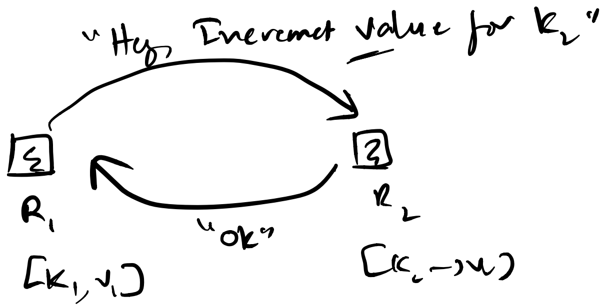
- * R_2 can fail
- * Req / response could be dropped.
- * Req / response is very slow

→ Undetectability of failures

"Timeouts"

1s

- * Timeouts + Retries have performance implications
- * Programming becomes complicated.



Naive Timeouts + retries could break application semantics

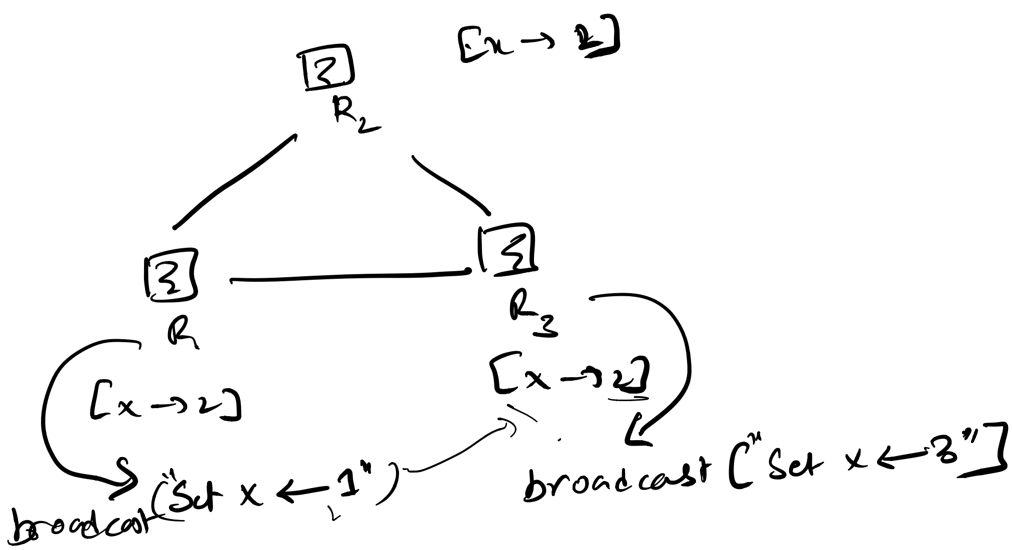
→ Partial failures are undetectable

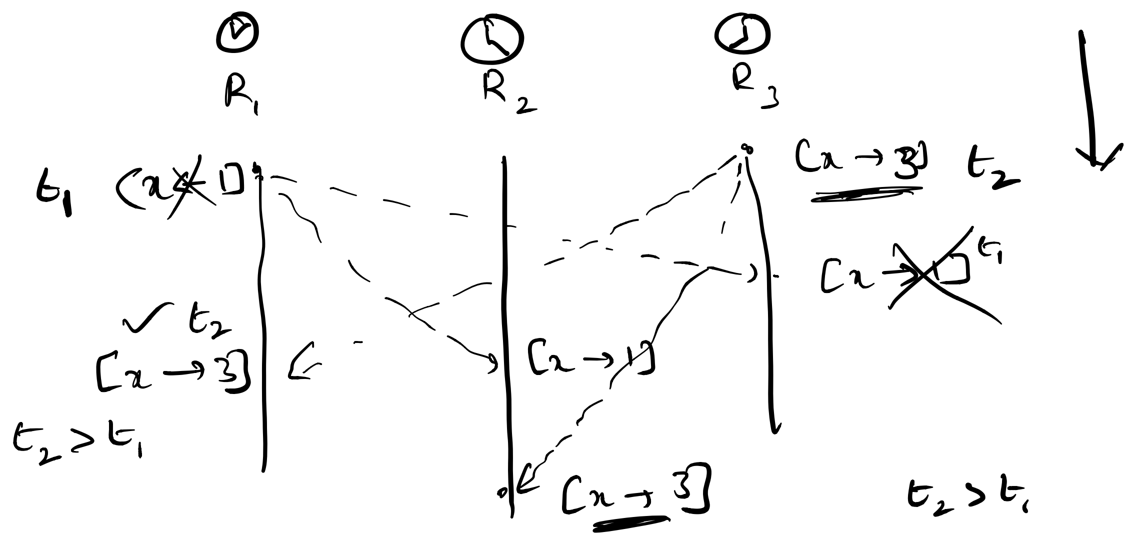
⇒

Fundamental uncertainty in distributed programming.

Models

→ Axis of time





Divergence

Synchronous model of a distributed system

(+) Programming is a bit easier

(−) Impractical.

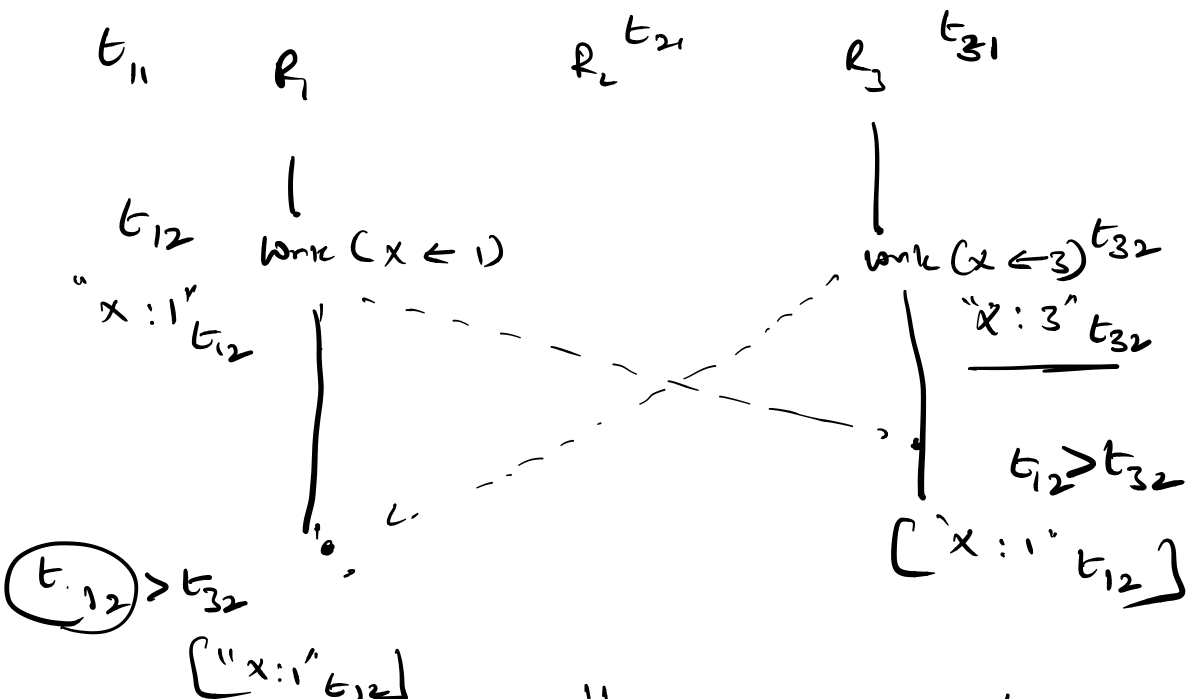
Alternative: Each machine runs on its own local clock — Asynchronous model

→ Need not be wall clock time.

→ Count time from a specific pre-define Event — last system restart.

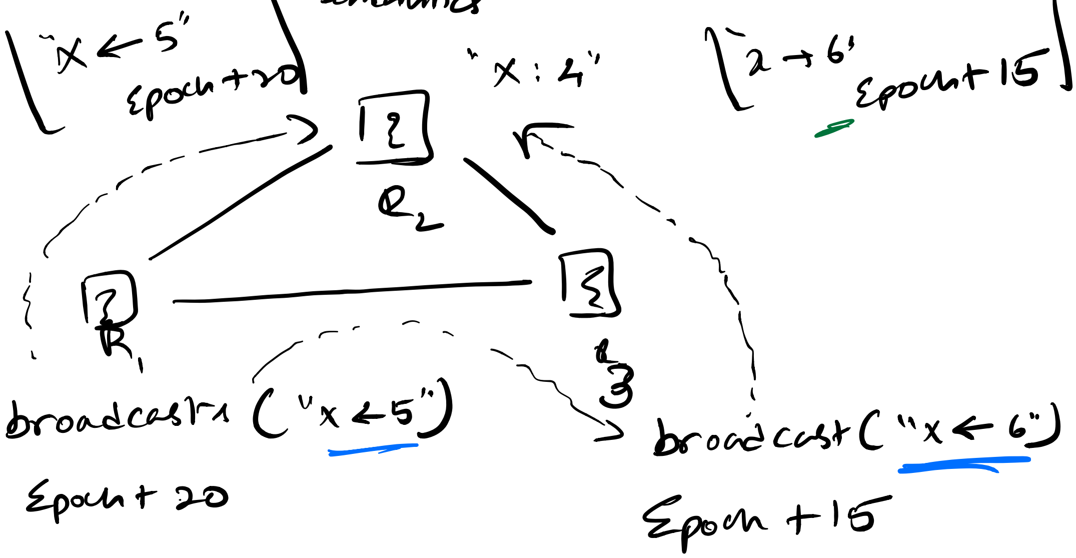
Monotonic clock.

eg: Unix time: Epoch + 2 million S



"Convergence!!"

But may not match user-intended semantics



Execution does not respect Causal

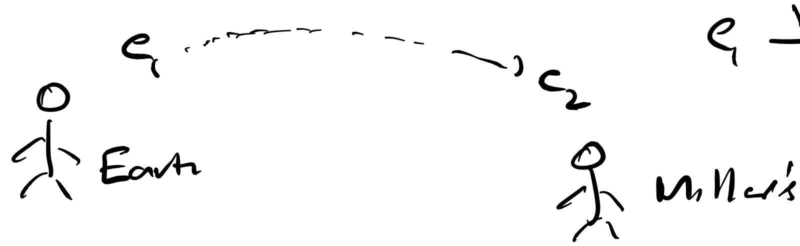
Ordering of Events!!

$(x \to 6) \xrightarrow{\text{depends on}} (x \to 5)$

in other words

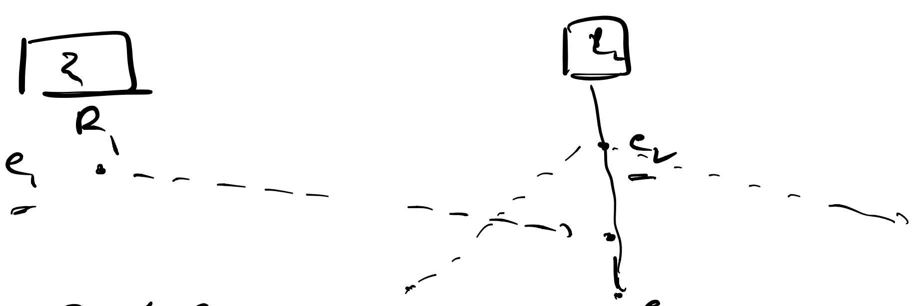
$(x \to 5) \xrightarrow{\text{happened-before}} (x \to 6)$

Causality is the primary notion of **time** in an asynchronous distributed system.



$e_1 \xrightarrow{hb} e_2$

Logical clock



$e_1 < e_3$

$e_1 \not< \not> e_2$

→ physical clock : totally ordered time

→ Logical clock : Partially-ordered time

Vector clocks

t_1 t_2

$t_1 < t_2$: t_1 has causally preceded t_2

But vector clock timestamps t_1 & t_2 may not be comparable!)

In which case neither $t_1 \xrightarrow{hb} t_2$ nor $t_2 \xrightarrow{hb} t_1$

E	M
2	3

Vector of scalar time stamps