CSE138 Lecture 2

last time:

distributed systems: what and why?

this time:

- ✓ - time and clocks
- ✓ - Lamport diagrams, causality and happens-before
- ✓ - Network models
- - state and events
- if time:
  - - partial and total orders
  - - Lamport clocks

How do we use clocks:

1) to mark particular points in time.

"This class starts at 3:20"

"This item in my browser cache expires at 2024-01-28"

2) to measure durations or intervals of time.

"This class is 95 minutes long."

"This user spent 3 minutes and 42 seconds on our website."

Physical clocks

time-of-day clocks:

- usually synchronized between machines using NTP.
- bad for measuring durations/intervals of time

  because they can jump backward/forward!

  2016 Cloudflare leap second bug
- OK (ish) for timestamping particular points in time.

monotonic clocks:

- only go forward. (53,021 39,995)

```
import time
time.monotonic()
```

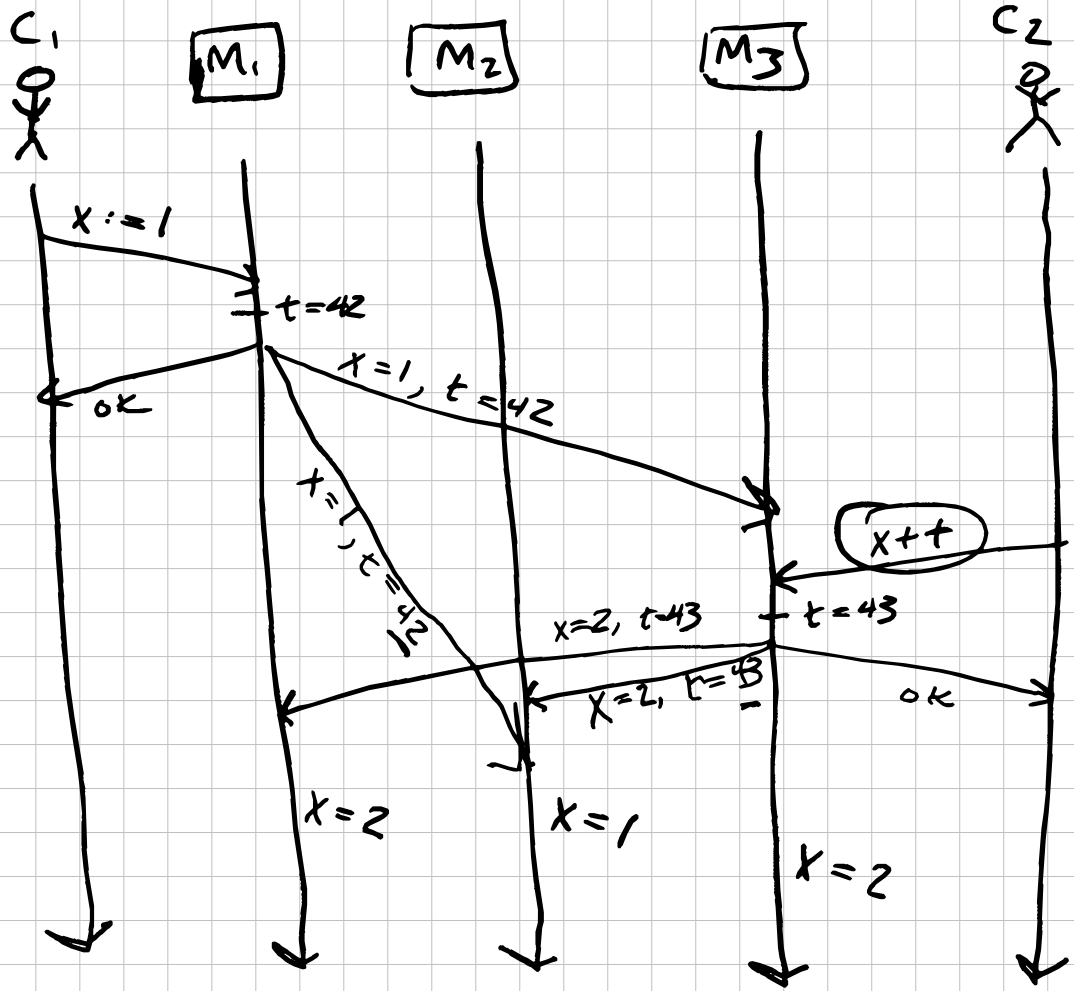
monotonic clock API in Python

- bad for timestamping particular points in time.
- good for measuring durations/intervals of time.

Why aren't time-of-day clocks good for marking points in time?

from:

"Designing Data-Intensive Applications"
by Martin Kleppmann

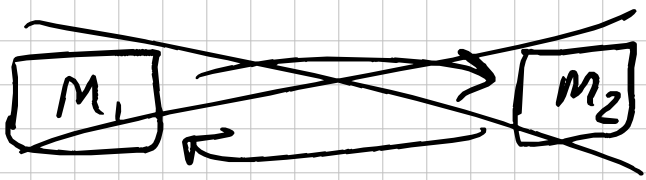


Logical clocks

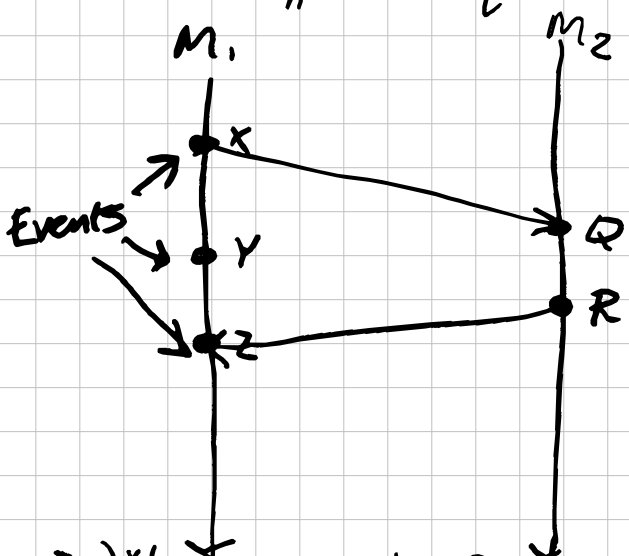
only measure the ordering of events.

$A \rightarrow B$ "A happened before B"

- A could have caused B.
- B could not have caused A.



Lamport diagrams (time diagrams, sequence diagrams, message sequence charts)



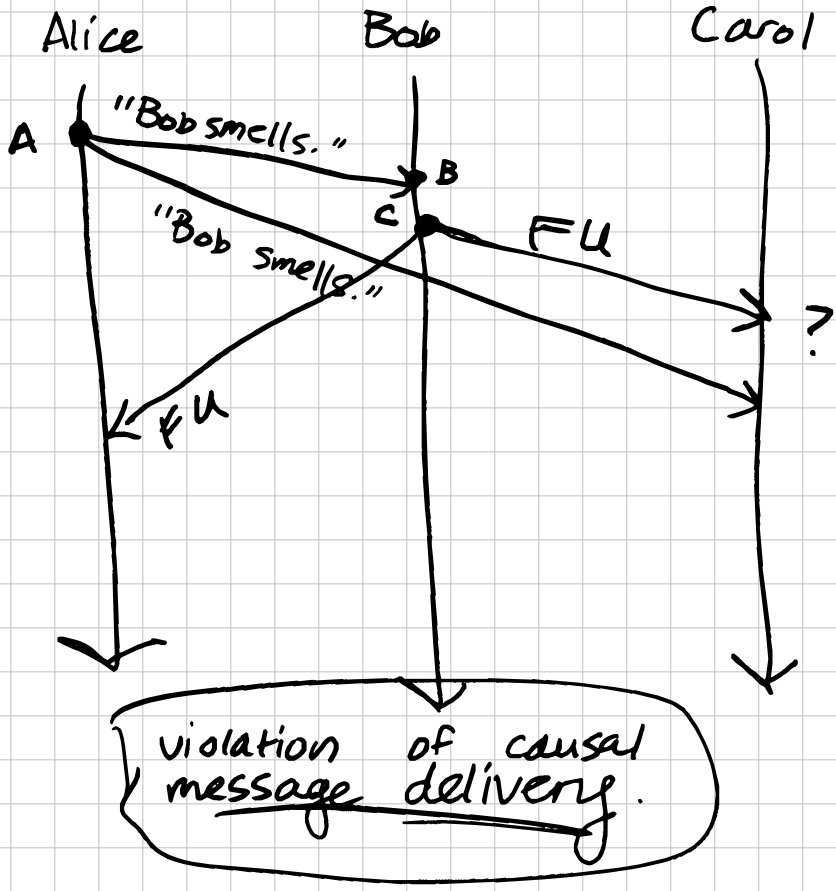
($\rightarrow$)"happens-before":

Given two events e_1 and e_2, we say $e_1 \rightarrow e_2$ if any of the following are true:

- e_1 and e_2 occur on the same process with e_1 first.
- e_1 is a message send event, and e_2 is the corresponding receive event.
- if there's another event, e', such that $e_1 \rightarrow e'$ and $e' \rightarrow e_2$, then $e_1 \rightarrow e_2$.

→ makes happens-before a transitive relation.

Classic Lamport diagram:



Network models ← set of assumptions about network behavior

- synchronous network -

a network in which there exists some bound n such that no message takes longer than n units of time to be delivered.

↑ physical

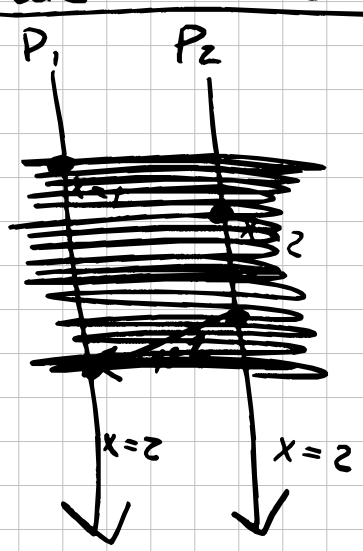
- asynchronous network -

a network in which there exists no bound n such that, etc.

Same as above.

↑ We'll mostly be discussing asynchronous networks

State and events



- set x to 1
- receive x=2 from P2
  x=2

- set x to 2
- send x=2 to P1
  x=2

Example: Event store

given a history of events, you can determine what state you're in.

given a state, you don't necessarily know what got you there.