CPSC 465/565 Theory of Distributed Systems

James Aspnes

2023-09-11

Today's exciting topics

- Casual ordering
- Logical clocks
- Snapshots

Executions and schedules

For every **execution**

$$\Xi = C_0 \alpha_1 C_1 \alpha_2 C_2 \alpha_3 C_3 \dots$$

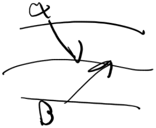
there is a **schedule**

$$S = \alpha_1 \alpha_2 \alpha_3 \dots$$

that includes only the events.

Given C_0 and S, and a deterministic protocol, we can reconstruct Ξ.

The problem with schedules



Schedules contain too much information!

Suppose α and β are events at different processes p and q.

Then $S_1 = \alpha\beta$ and $S_2 = \beta\alpha$ look the same to both p and q.

Formally:

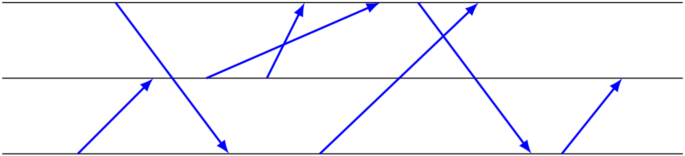
- $S|p$ is subsequence of S consisting only of events involving p.
- S_1 is **indistinguishable** from S_2 by p if $S_1|p = S_2|p$.
  - ($S_1 \sim_p S_2$ for short.)

In the example,

- $S_1 \sim_p S_2$ since $S_1|p = S_2|p = \alpha$.
- $S_1 \sim_q S_2$ since $S_1|q = S_2|q = \beta$.

So processes don't know order of α and β.

Causal ordering (setup)



Time $\rightarrow$

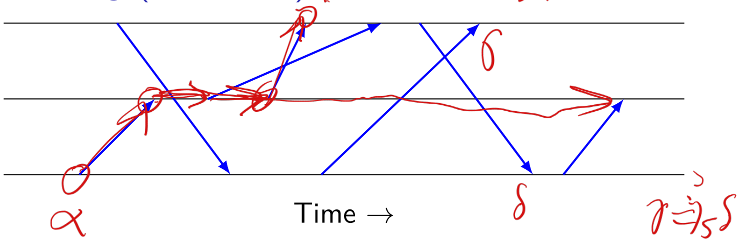
Want to capture order of events as observed by processes.

This will require some tweaks to our model:

- Events are either `send`(m) or `recv`(m) for a single message m.
- We'll assume m includes sender/recipient info.
- We'll also assume all messages are unique.

Causal ordering (definition)

$\alpha \Rightarrow_S \beta$



Given a schedule S, α happens-before β on S ($\alpha \Rightarrow_S \beta$) if

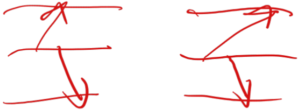
1. α and β are events of the same process and $\alpha <_S \beta$,
2. $\alpha = \texttt{send}(m)$ and $\beta = \texttt{recv}(m)$, or
3. There is a chain of events
   $\alpha = \alpha_0 \Rightarrow_S \alpha_1 \Rightarrow_S \alpha_2 \ldots \Rightarrow_S \alpha_k = \beta$.

Cases (1) and (2) only occur if $\alpha <_S \beta$.

Case (3) is transitive closure.

So $(\Rightarrow_S) \subseteq (<_S)$ and $\Rightarrow_S$ is a partial order.

Claim: $\Rightarrow_S$ captures all observable ordering in S



Formal version:

Define S to be a **causal shuffle** of S' if

- S is a permutation of S', and
- $(\Rightarrow_S) = (\Rightarrow_{S'})$.

Then

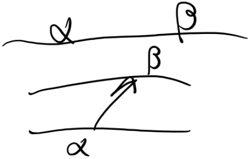
- $\underbrace{S \sim_p S'}$ for all p iff S is a causal shuffle of S'.

$S|p = S'|p \quad \forall p$

Proof of claim: Indistinguishability implies causal shuffle

$S \sim_p S' \equiv S|p = S'|p$

$S|p$



($\Rightarrow$) If $S \sim_p S' \forall p$, then

1. S is a subpermutation of S'. Proof:
   - Every event α in S is an event of some process p, so
   - $a \in S \Rightarrow \alpha \in S|p = S'|p \subseteq S'$ (and vice versa)
2. $(\Rightarrow_S) \subseteq (\Rightarrow_{S'})$. Proof: Suppose $\alpha \Rightarrow_S \beta$, then one of:
   - α and β are events of same process p
     - $S|p = S'|p$ so $\alpha <_S \beta \Rightarrow \alpha <_{S'} \beta$, giving $\alpha \Rightarrow_{S'} \beta$.
   - $\alpha = \mathtt{send}(m)$, $\beta = \mathtt{recv}(m)$; same m in S' gives $\alpha \Rightarrow_{S'} \beta$.
   - $\alpha \Rightarrow_S \ldots \Rightarrow_S \beta$ from previous cases.
     - By induction on length of sequence get $\alpha \Rightarrow_{S'} \ldots \Rightarrow_{S'} \beta$.

Now reverse both arguments to get S is a permutation of S' with same happens-before relation.

Proof of claim: Causal shuffle implies indistinguishability

$\alpha \overset{S}{\Rightarrow} \beta$

$\alpha \underset{S'}{\Rightarrow} \beta$

same events in same order

1. S is a permutation of S'
   - Implies $S|p$ is a permutation of $S'|p$ for all p.
2. $(\Rightarrow_S) = (\Rightarrow_{S'})$
   - Implies $\alpha <_S \beta$ iff $\alpha <_{S'} \beta$ if events of same process p.

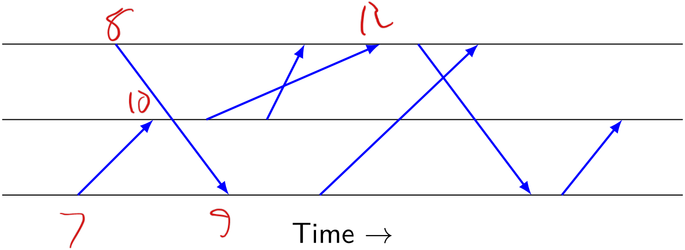
So $S|p$ and $S|p'$ have same events in same order: they are equal.

If S' is a causal shuffle of S,
and S is a sched.
$\Rightarrow$ S' is a sched.

proof: ind.

Logical clocks

10

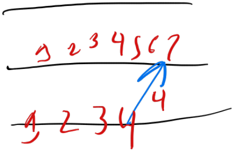


$\Rightarrow_S$ includes all order info processes can in principle deduce from S.

How can processes compute this?:

- Can't build global clock: not provided by model.
- Replace with **logical clock**
  - Logical clock = protocol that assigns a time to each event.
  - Times are from some totally ordered set.
  - $\alpha \Rightarrow_S \beta$ implies $t_\alpha < t_\beta$ (causality)
  - But maybe not the other direction: partial vs total order.

Lamport clocks (Lamport 1978)



- Each process p maintains local clock t_p.
- Local computation event: $t_p \leftarrow t_p + 1$.
- Send event: $t_p \leftarrow t_p + 1; \texttt{send}(m, t_p)$.
- $\texttt{recv}(m, t)$ event: $t_p \leftarrow \max(t_p + 1, t + 1)$.

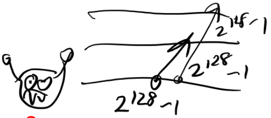
Time of event α is value of t_p at end of α.

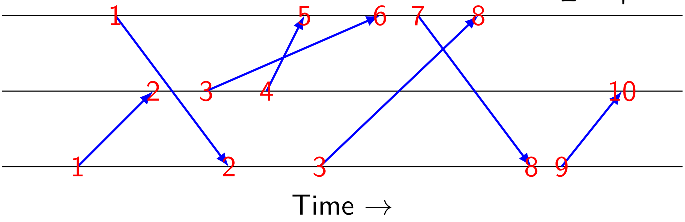
Claim: timestamp order $\supseteq (\Rightarrow_S)$

Proof: Suppose $\alpha \Rightarrow_S \beta$, then one of

1. α, β on same process p; t_p is strictly increasing.
2. $\alpha = \texttt{send}(m), \beta = \texttt{recv}(m)$:
    - t_α is time t in message.
    - $t_\beta \geq t + 1$.
3. Transitive closure: follows from previous cases.

Lamport clocks





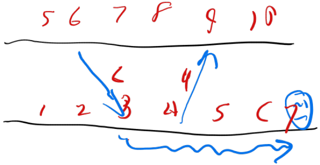
Pro:

- Total order on events consistent with observable order.
- Annotation only: no interference with underlying protocol.

Con:

- Can get big! A single bad message can max out all clocks.

Neiger-Toeug-Welch clocks



(Neiger-Toueg 1987, Welch 1987)

- Each process maintains its own local clock t_i.
- t_i increments every now and then, and for each new event.
- $\texttt{send}(m, t_i)$ includes send event's timestamp.
- Receiver j **buffers** (m, t) until $t < t_j$.

Time of receive event is when message is finally delivered.

Claim: $\alpha \Rightarrow_S \beta$ implies $t_\alpha < t_\beta$. Proof:

1. Events of same process: local clock increases.
2. $\texttt{send}(m)$ vs $\texttt{recv}(m)$: buffering rule.
3. Transitive closure: $<$ on timestamps is transitive.

Bad cases for Neiger-Toeug-Welch clocks

What if some process's local clock is very far off?

- I think it's 2077.
- I receive messages with no buffering!
- But nobody accepts my messages for another 54 years.
- Other processes' local clocks are not affected.

NTW clocks work best if local clocks are mostly synchronized.

Vector clocks

x = 0 1 2 3

y = 4 1 6 2

$x \le y$ iff

$\forall i: x_i \le y_i$

Lamport clocks and Neiger-Toueg-Welch clocks give a total order.

What if we want to recover partial order $\Rightarrow_S$ exactly?

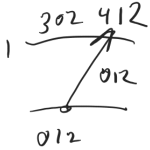
Idea:

- Instead of totally ordered timestamps from $\mathbb{N}$,
- Use partially ordered timestamps from $\mathbb{N}^{[n]}$.

Ordering: $x \le y$ iff $x_i \le y_i$ for all i.

(Fidge 1991, Mattern 1993)

Vector clock: implementation



Timestamp $= \langle t_1, t_2, \dots t_n \rangle$ where $t_i =$ local clock at i.

Assign each event smallest timestamp consistent with

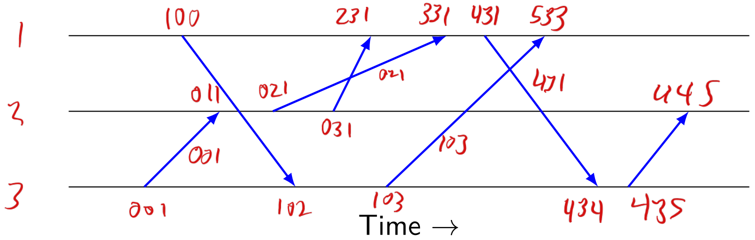
1. $\alpha < \beta$ if $\alpha <_S \beta$ and both events are of same process.
2. $\alpha < \beta$ if $\alpha = \texttt{send}(m)$, $\beta = \texttt{recv}(m)$.

(2) requires sending t along with m as in other logical clocks.

Implementing the rule:

- Track local vector $t^i = \max t^\alpha$ of all events I know about.
- Start at $t^i = \langle 0, 0, \dots, 0 \rangle$ (before any events).
- For a local computation or send event, update $t^i_i \leftarrow t^i_i + 1$.
- For a receive event of message with timestamp s:
  - Update $t^i_i \leftarrow t^i_i + 1$, then
  - For all j, set $t^i_j = \max(t^i_j, s_j)$.

Vector clock: correctness

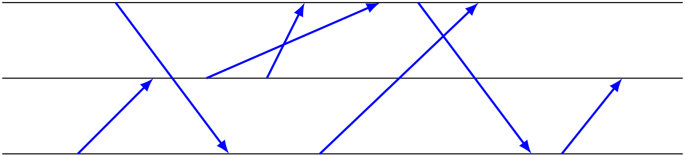


Claim: $\alpha \Rightarrow_S \beta$ iff $t^\alpha < t^\beta$.

Easy direction: $\alpha \Rightarrow_S \beta$ implies $t^\alpha < t^\beta$. Proof: same as always.

1. Same process: max + t^i incr.
2. send(m) & recv(m): max + t^i incr
3. TC.

Vector clock correctness:



Time $\rightarrow$

The issue: t_j could come from β' before on γ here

Claim: $\alpha \Rightarrow_S \beta$ iff $t^\alpha < t^\beta$.

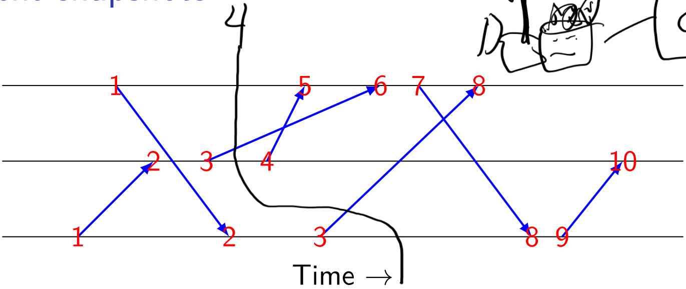
Harder direction: $\alpha \not\Rightarrow_S \beta$ implies $t^\alpha \not< t^\beta$.

- Lemma: For any event β of process i, and any $j \neq i$, $t_j^\beta = \max_{\gamma \text{ event of } j, \gamma \Rightarrow_S \beta} t_j^\gamma$.
- Proof of lemma: induction on execution.
  - t_j^i can only rise on `recv` event.
  - ~~Corresponding send event gives γ.~~ Send event α has t_j^α ... Ind hyp.
  - If t_j^i unchanged, apply induction hypothesis to previous event.
- If $\alpha \not\Rightarrow_S \beta$, α event of j, then $t_j^\alpha > \max_{\gamma \Rightarrow_S \beta} t_j^\gamma$ implies $t^\alpha \not< t_\beta$. ?

Do not trust this proof.

fix is to apply Ind hyp to each & take max.

Consistent snapshots

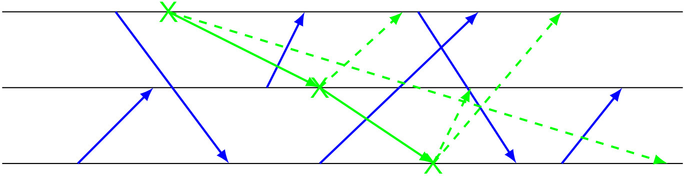


Any totally-ordered logical clock allows **snapshots**:

1. Pick a logical time in advance.
2. Record the last state at each process before this time.
3. Collect states somewhere.
4. Causal shuffle implies consistency.

Problem: How to pick a time?

Chandy-Lamport snapshot



Assumption: FIFO channels.

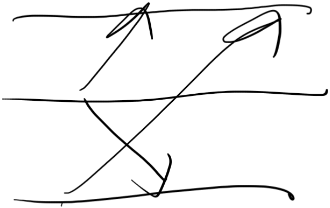
- Initator records state and sends snap to all neighbors.
- Each other node does same upon receiving snap for the first time.

Claim: Pre-snap states $<$ post-snap states is causal shuffle.

Proof: Argue $\alpha \Rightarrow_S \beta$ implies we won't put β before, α after.

1. Same process p: follows from order on $S|p$.
2. $\alpha = \texttt{send}(m), \beta = \texttt{recv}(m)$: follows from FIFO channels.
3. Transitive closure etc.

Next time



Synchronizers!