TIME AND ORDERING :-

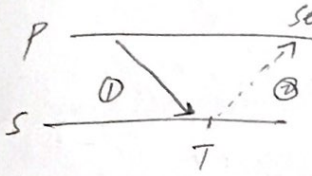
Challenges → Asynchronous Distributed System → Clock skew, Clock Drift

* Maximum Drift Rate ⇒ Error ~ 2*MDR

M → Maximum Acceptable skew Synchronization ⇒ $\frac{M}{2*MDR}$

Synchronization → External → with external standard
Internal → node clocks within bound (Berkeley Algorithm)

CRISTIAN'S ALGORITHM :-



Problem: Server to Process time not taken care of
Time in path ② not considered

RTT → known S → P Latency → L_{sp} P → S Latency → L_{ps}

Actual Time $\in [t + L_{sp}, t + RTT - L_{ps}]$

Depends on OS, buffer messages and internal processing.

Time set = $t + \left(\frac{RTT + L_{sp} - L_{ps}}{2}\right)$

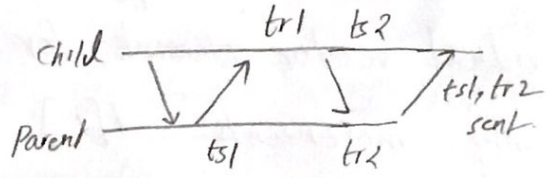
* Not allowed to decrease clock value → can change drift speeds

NTP (NETWORK TIME PROTOCOL) :-

Primary → UTC Time

Secondary

Tertiary / client



Offset $o = (tr1 - ts2 + ts2 - ts1)/2$.

$|o_{real} - o| < \frac{|L_2 + L_1|}{2}$

Round trip time

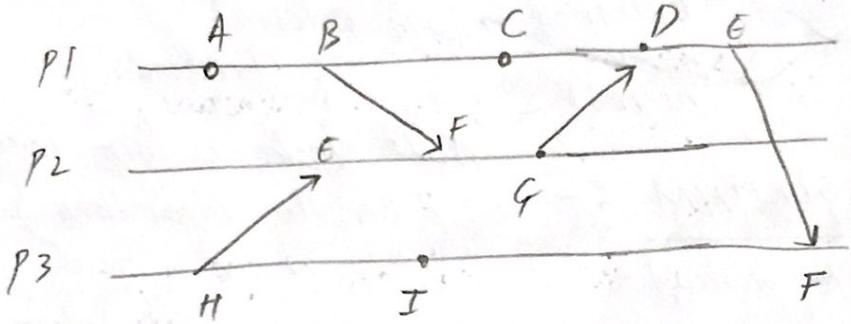
Error is bounded by $\frac{RTT}{2}$

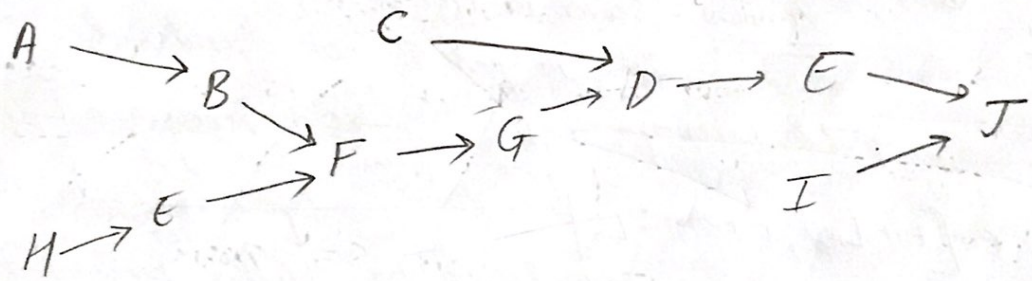
LAMPORT TIMESTAMPS :-

★ If Causality is satisfied by some metric, synchronisation is not required. Happens-Before denoted by $\rightarrow$

$a \rightarrow b$ and $b \rightarrow c \Rightarrow a \rightarrow c$ (Partial ordering. Not all events related)

EX.





ASSIGNING TIMESTAMP :

Receive event = max (local clock, message timestamp) + 1 ← New counter

Ⓐ May not imply causality for events. (Are unrelated in partial order) ← Called Concurrent events

VECTOR TIMESTAMPS :-

Same method as LAMPORT Timestamp but now increments happen in individual vector elements for each process.

Process i only increments $V[i]$

↳ Syncs all other elements with messages from other processes.

Causally Related if $VT_1 < VT_2$ (Atleast one element strictly less and other elements $\leq$)

GLOBAL SNAPSHOT :—

↳ For having a checkpoint
↳ Garbage collection
↳ Deadlock detection.
↳ Termination of computation

Ⓐ Capture instantaneous state of each process and also point to point channels

- FIRST APPROACH → ① Synchronise clocks
② Record states at known time t

PROBLEM → Time synchronisation has error.
→ State of messages in channels missing.

Ⓐ Synchronisation not required (Causality sufficient)

REASON : Ⓐ State can move from one to another with same sequence of events (causal paths cause changes)

SYSTEM MODEL :—

→ N processes
→ 2 uni-directional communication channels (FIFO)
→ No failure
→ All messages arrive intact

Note that these constraints bring determinism to the system

Ⓐ Algorithm must not interfere with applications.

Each process records own state → Global state is collected in distributed manner

CHANDY - LAMPORT GLOBAL SNAPSHOT ALGORITHM :—

① Initiate self state snapshot.
② Special message Marker sent to other processes (outgoing channe
③ Starts recording incoming messages on each channel

On receiving a Marker message on C_{ki}

- P_j records own state
- Marks C_{ki} as "empty"
- Sends marker messages
- Starts recording messages

⇒ If already seen a Marker → Just do this step

Ⓐ Later the snapshots may be collected by a central server to calculate global snapshot

⇒ Correct in terms of causality. → Concept of consistent cuts

CORRECTNESS → Liveness → Something "good" will happen
CORRECTNESS → Safety → Something "bad" never happens

Liveness: Termination, Consensus, Completeness in failure detection, No deadlock, Accuracy in failure detector.
Safety: No orphaned objects, Bad consensus

FOR GLOBAL STATES
- Liveness → Causal path from $S \to S'$ exist for that given property
- Safety → S satisfies P_r and all global states reachable from S satisfies P_r.

CHANDY-LAMPORT ALGORITHM (due to causal correctness)
Used to detect stable properties → Once true, always true

- Liveness: Termination
- Safety: Deadlocks, Orphaned objects