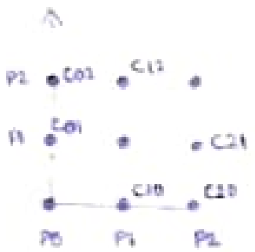
2.

A MODEL OF DISTRIBUTED COMPUTATIONS



$P0 \xrightarrow{m02} P2$

P → async process

C → channel, unidirectional

m → message

N processes } distributed program

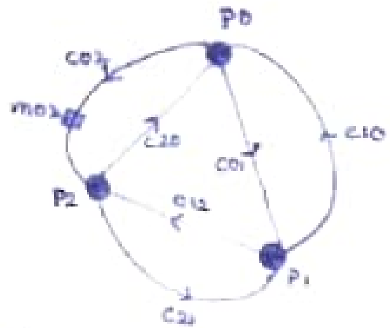
$N^2 - N$ channels

$\underbrace{\hphantom{N^2 - N \text{ channels}}}_{}$ distributed system (that uses message passing)

- no global shared clock
- communication delay unpredictable

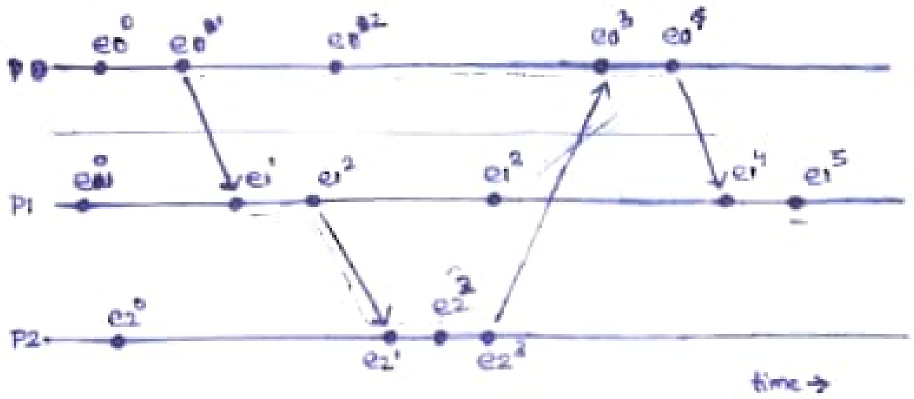
global state

- states of processes
- states of channels



Process ⇒ (internal) action → (send) action → (recieve) action (sequential) (atomic)

↳ also called events



space-time diagram of distributed execution

within a process, all events causally follow since they are sequential.

$e0^0 \rightarrow e0^1 \qquad e0^2 \rightarrow e0^4$

send and recieve of a message causally follow

$e0^1 \rightarrow e1^1 \qquad e2^3 \rightarrow e0^3$

now we can make some causal links

$e_0^1 \rightarrow_i e_0^2 \rightarrow_i e_0^3 \rightarrow_i e_0^4$ $\Rightarrow e_0^1 \rightarrow_i e_0^4$

$e_1^1 \rightarrow_i e_1^2 \rightarrow_i e_2^1 \rightarrow_i e_2^2$ $\Rightarrow e_1^1 \rightarrow e_2^2$

lamports "happens before"

$e_i^x \rightarrow e_j^y$ $\Leftrightarrow$ $\begin{cases} e_i^x \rightarrow e_j^y & \text{where } i = j \text{ \& } x < y \\ e_i^x \rightarrow_{msg} e_j^y & \\ e_i^x \rightarrow e_k^z \wedge e_k^z \rightarrow e_j^y & \text{where } \exists e_k^z \in H \end{cases}$

causal precedence relation

$\forall e_i^x \ \forall e_j^y \in H$

$H = \cup_i h_i, \ h_i = \{e_i^0, e_i^1, \ldots\}$

transitive

all events happen before e_1^5

$\therefore e_1^5$ has the knowledge of all other events

$e_i \not\rightarrow e_j \not\Rightarrow e_j \not\rightarrow e_i$

$e_i \rightarrow e_j \not\Rightarrow e_j \not\rightarrow e_i$

concurrent events are those which are not causally related

$e_2^2 \parallel e_1^3$, $e_1^3 \parallel e_0^4$ but $e_2^2 \not\parallel e_0^4$

not transitive

$e_i^x \parallel e_j^y \Rightarrow e_i^x \not\rightarrow e_j^y$ and,

$e_j^y \not\rightarrow e_i^x$

models of service of a communication network:

- non-FIFO random order
- FIFO channel is sequential
- causal causally-related messages are sequential
  $send(m_{ij}) \rightarrow send(m_{kj})$
  $\Rightarrow recv(m_{ij}) \rightarrow recv(m_{kj})$ } CO

replicated DB:

update in same order for consistency

INSERT, UPDATE

GLOBAL STATE OF A DISTRIBUTED SYSTEM

- states of processes (registers, stacks, memory, local app context, ...)
- states of channels (set of messages in transit in the channel)

$PS_i^x \sim e_i^x$

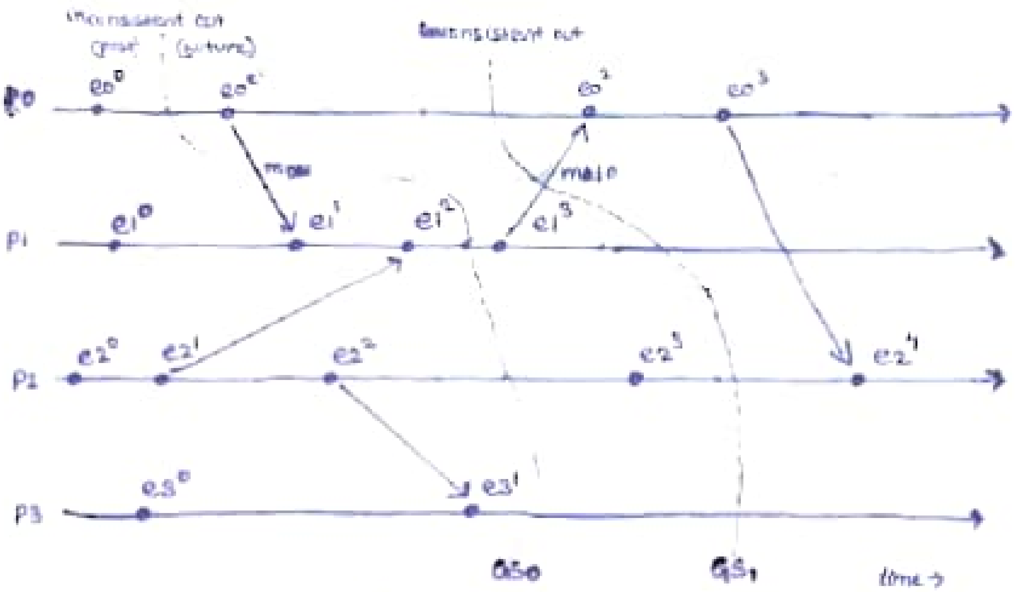
$CS_{ij}^{x,y} = \{ m_{ij} \mid send(m_{ij}) \in PS_i^x \wedge recv(m_{ij}) \notin PS_j^y \}$

$GS = \{ \cup_i PS_i^{x_i} , \cup_{j,k} CS_{jk}^{y_j, z_k} \}$

for a global snapshot to be meaningful, the states of all components should be consistent, and not violate causality (a message cannot be recieved if it was not sent).

a GS is consistent iff →

$\forall m_{ij} : send(m_{ij}) \notin PS_i^{x_i} \Rightarrow m_{ij} \notin CS_{ij}^{x_i, y_j} \wedge recv(m_{ij}) \notin PS_j^{y_j}$



$GS_0 = \{ PS0^0 , PS1^2 , PS2^2 , PS3^1 \}$ is inconsistent as m_{01} has not been sent yet

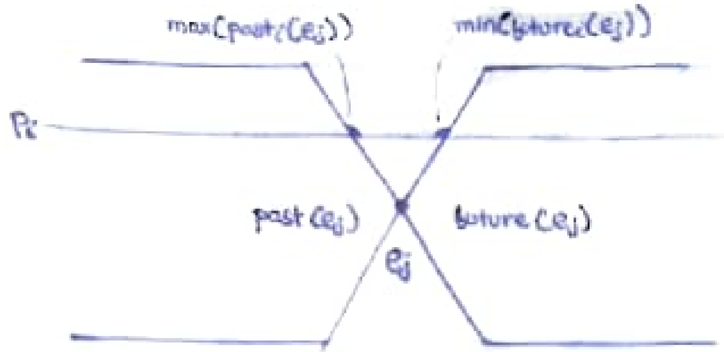
$GS_1 = \{ PS0^1 , PS1^3 , PS2^3 , PS3^1 \}$ is consitent, C_{10} contains m_{10}.

a GS is transitless iff

$\forall i \, \forall j : 0 \le i, j \le N :: CS_{ij}^{x_i, x_j} = \phi$

all channels are empty.

a GS is strongly consistent iff it is consistent & transitless



latest (send) event that affected e_j in P_i is $max(past_i(e_j))$

earliest (recieve) event that was affected by e_j in P_i is $min(future_i(e_j))$

latest (send) event of every process that affected e_j is max-past(e_j)

$max\_past(e_j) = \cup_{(\forall i)} \{max(past_i(e_j))\}$

(surface of the past cone of e_j) (consistent cut)

earliest (recieve) event of every process that is affected by e_j is min-future(e_j)

$min\text{-}future(e_j) = \cup_{(\forall i)} \{min(future_i(e_j))\}$

(surface of the future cone of e_j) (consistent cut)

all events that happen before "an event" are in the past cone.

$past(e_j) = \{e_i \mid \forall e_i \in H,\ e_i \rightarrow e_j\}$

all events that happen after "an event" are in the future cone

$future(e_j) = \{e_i \mid \forall e_i \in H,\ e_j \rightarrow e_i\}$

all events that lie outsid the past and the future cones of "the event" are concurrent with it.

$\therefore e_j \parallel H - past(e_j) - future(e_j)$