Logical Time, Causality & Global States

References

Lamport, L. (1978). Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, Vol. 21, No. 7, pp. 558-65.

Mattern, F. (1989). Virtual time and global states in distributed systems. *Proceedings of the Workshop on Parallel and Distributed Algorithms*, Amsterdam, North-Holland, pp. 215–226.

Fidge, C. (1991). Logical time in distributed computing systems. *IEEE Computer*, Vol. 24, No. 8, pp. 28-33.

References

Babaoglu, O. and Marzullo, K. (1993). Consistent global states of distributed systems: Fundamental concepts and mechanisms. *Technical Report UBLCS-93-1*, University of Bologna, 40 p.

Raynal, M. and Singhal, M. (1996). Logical time: capturing causality in distributed systems, *IEEE Computer*, Vol. 29, No. 2, pp. 49-56.

Baquero, C. and Preguica, N. (2016). Why logical clocks are easy, *Communications of the ACM*, Vol. 59, No. 4, pp. 43-47.

Causality

In an asynchronous distributed system where no global time frame exists, events of a computation can be ordered only based on the notion of “cause-and-effect”.

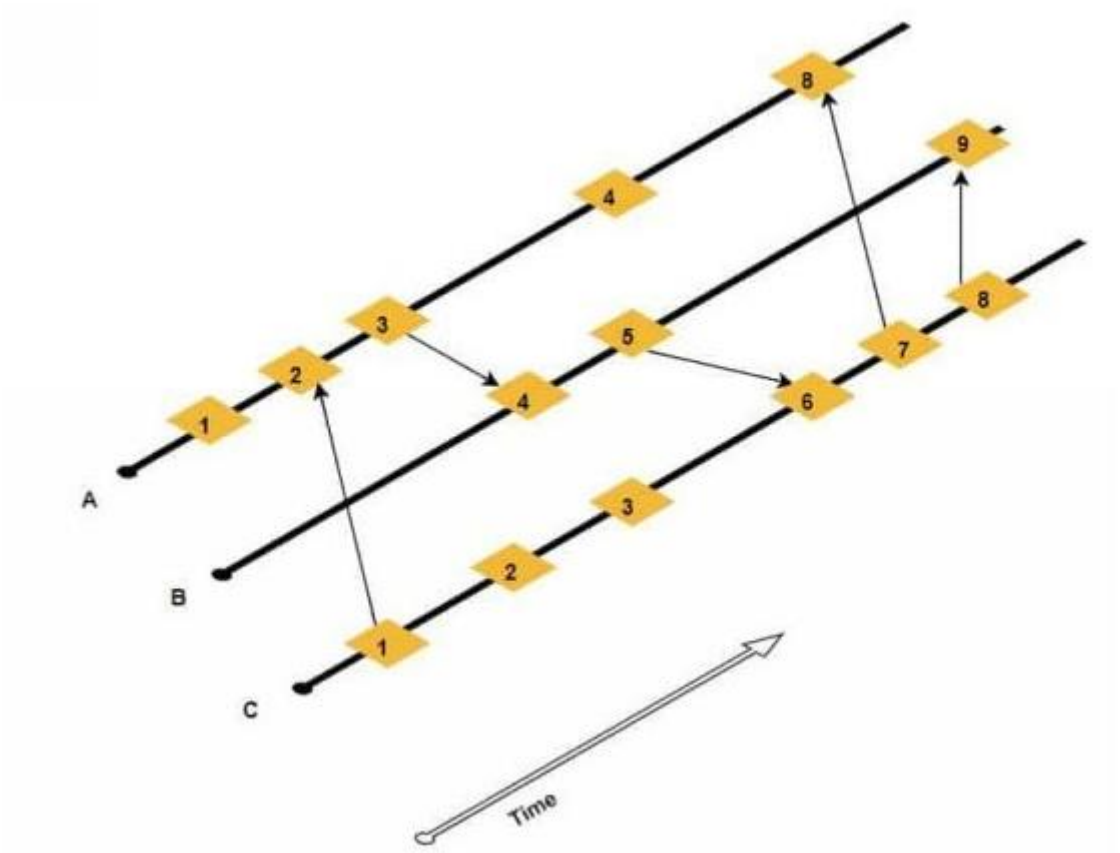
$e \rightarrow f$: e may causally affect f

We prefer not to interpret it as “happens before” because of the real-time connotation.

Logical Time

- **Scalar clocks**
- Vector clocks
- Matrix clocks

Scalar Clocks – Total order



Scalar clocks

Time domain is the set of integers.

Each process P_i is associated an integer variable C_i holding increasing values.

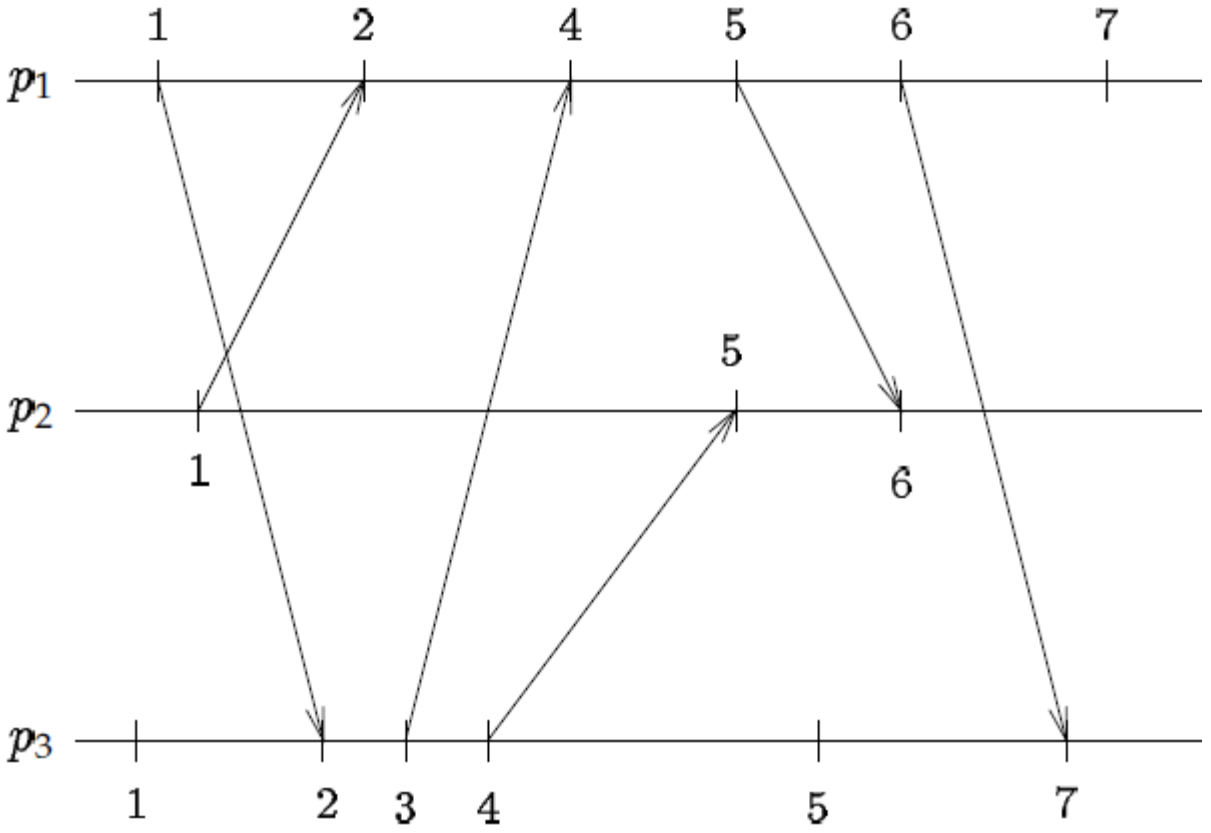
The logical local clock of P_i and its local view of the global time are here mixed up and represented by the only variable C_i.

Scalar Clocks – Updating Rules

P_i uses the following rules to update its clock:

- ***R1 :*** When a local event occurs:
  - $C_i := C_i + 1$
- ***R2 :*** When a message M, timestamped C_M, is received:
  - $C_i := \max(C_i, C_M)$
  - $C_i := C_i + 1$

Scalar Clocks



Logical time increases along causal paths

Scalar Clocks

If event *e* has the timestamp C(*e*)=h, then

- there are h events on the longest causal path ending at *e*.
- h-1 represents the minimum logical duration, counted in events, required before producing *e*.

Scalar Clocks

Scalar clock system

- Respects causality

Clock (consistency) condition

$$e \rightarrow f \Rightarrow C(e) < C(f)$$

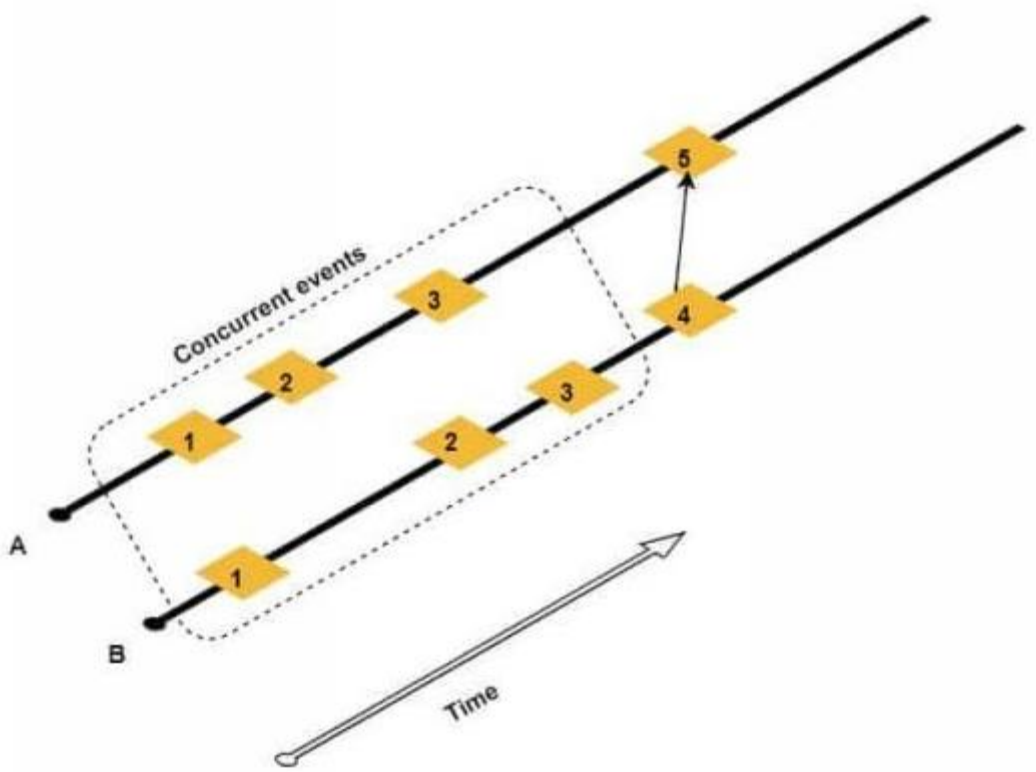
Scalar Clocks

Scalar clock system

- But does not capture it

$$C(e) < C(f) \not\Rightarrow e \rightarrow f$$

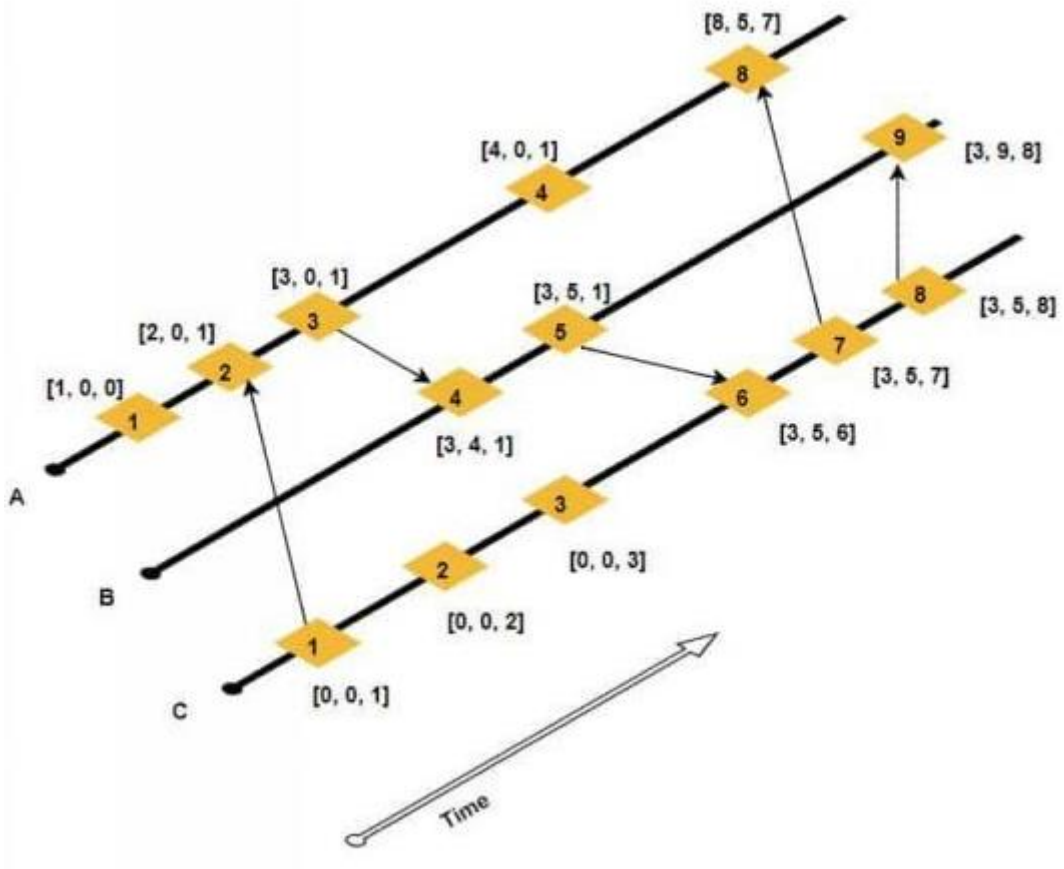
Scalar Clocks - Concurrent Events?



Logical Time

- Scalar clocks
- **Vector clocks**
- Matrix clocks

Vector Clocks - Capturing causality exactly



Vector clocks

A process P_i can always measure its progress by counting the number of events it has produced since the beginning (this number can be seen as its logical local clock).

Since there is one such clock per process, the time domain is n-dimensional: there is one dimension associated with each process.

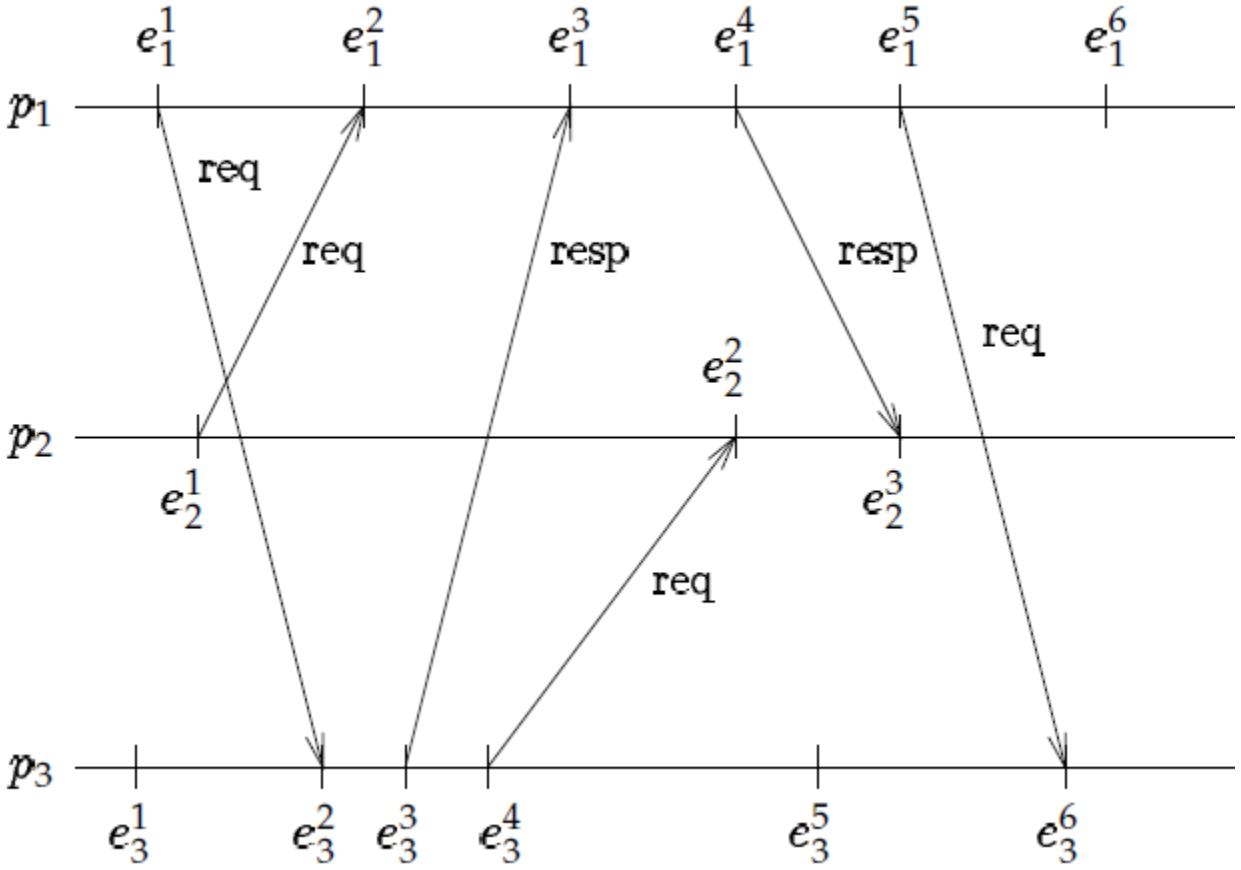
Each process P_i manages a vector $VC_i[1..n]$ that represents its view of the logical global time.

Vector Clocks – Updating Rules

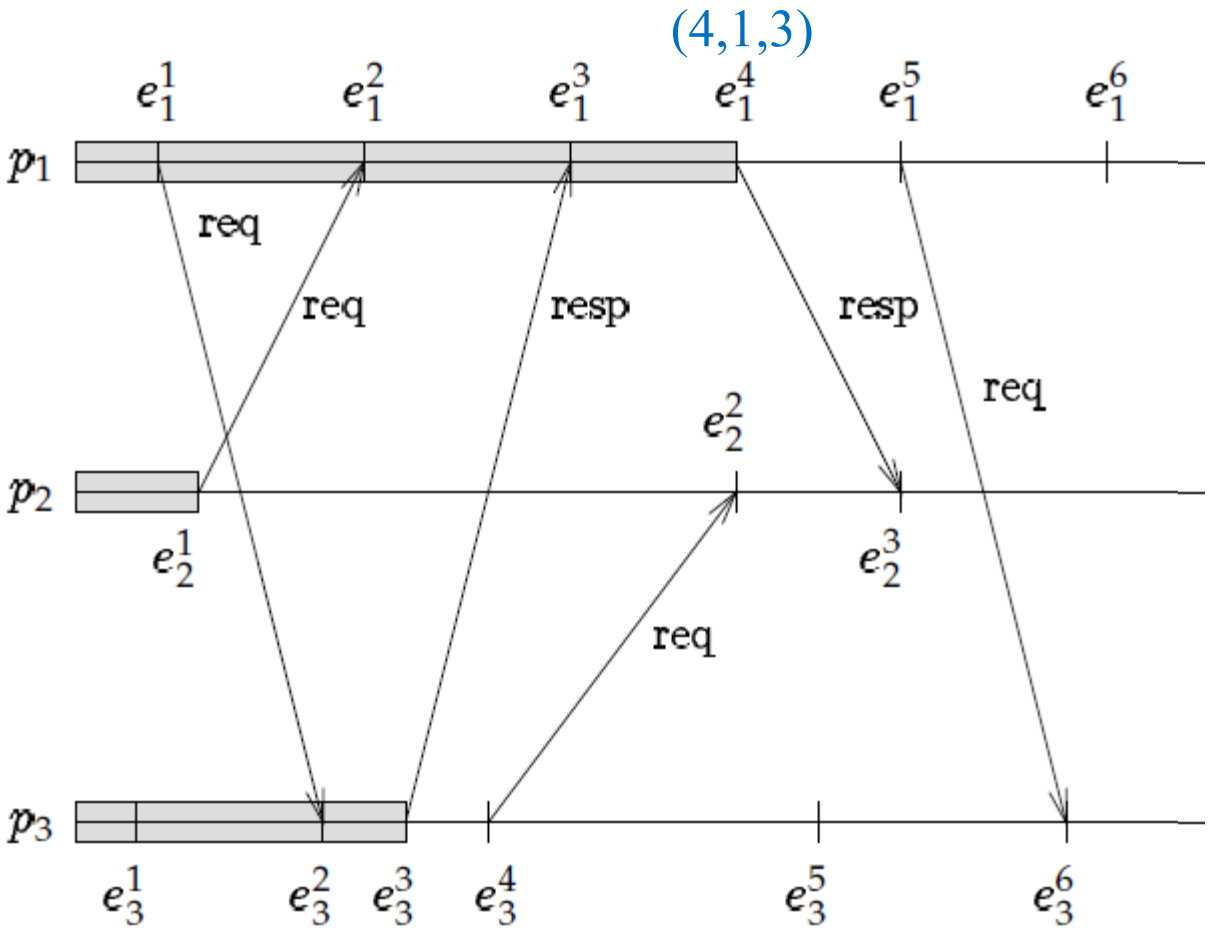
P_i uses the following rules to update its clock:

- ***R1 :*** When a local event occurs:
  - $VC_i[i] := VC_i[i] + 1$
- ***R2 :*** When a message M, timestamped VC_M, is received:
  - $VC_i := \max(VC_i, VC_M)$
  - $VC_i[i] := VC_i[i] + 1$

Space-Time Diagram of a Distributed Computation

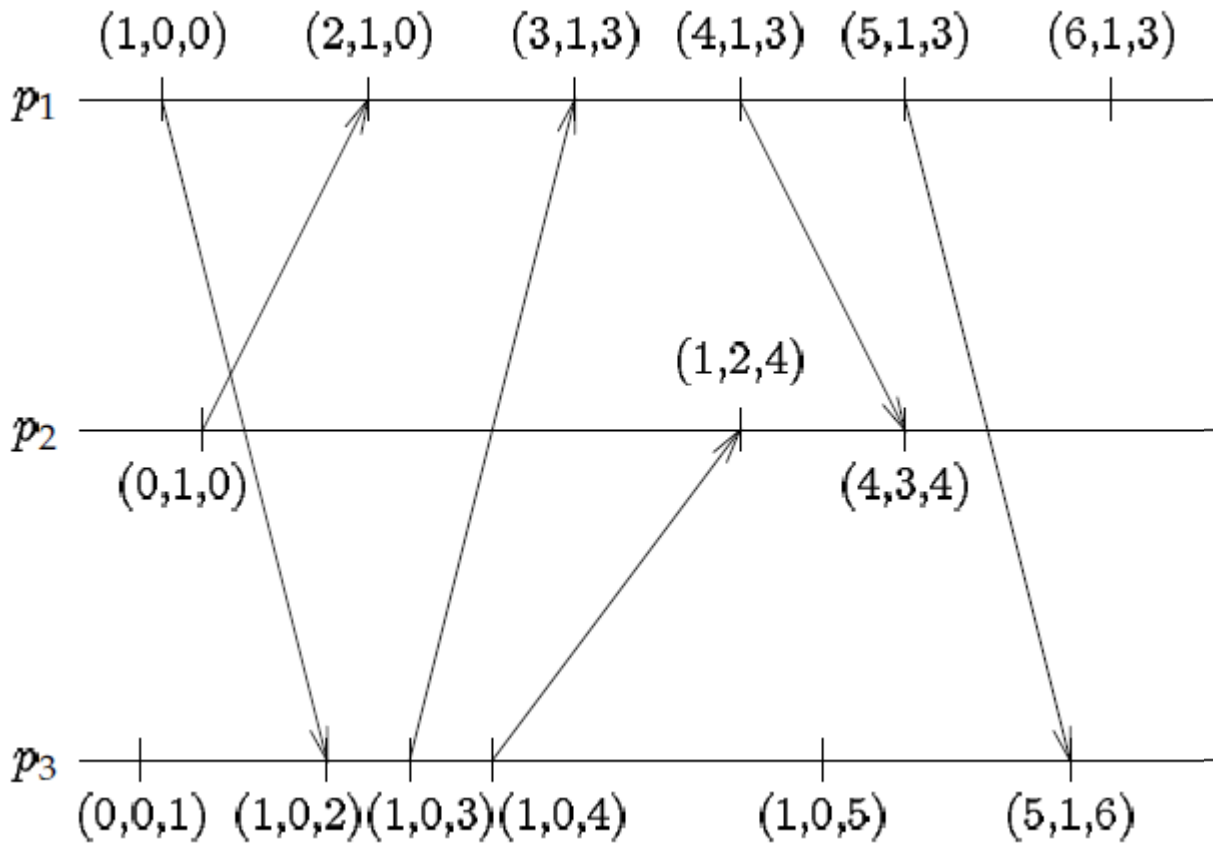


Causal History of Event e_1^4



VC_i is a digest of the current causal past of p_i

Vector Clocks



Vector clocks increase along causal paths

Vector Clocks

The ith component of vector clock at P_i, $VC_i[i]$, denotes the number of events that have occurred at P_i until that instant.

Logical local clock

The jth component of vector clock at P_i, $VC_i[j]$, denotes the number of events issued by P_j, as known by P_i.

Local image of the value of VCj[j]

Vector clocks impart knowledge to the processes.

Vector Clocks

If event *e* of P_i has the timestamp $VC_i(e)$, then

- $VC_i(e)[j]$, for all $j \neq i$, denotes the number of events executed by P_j that causally precede *e*.
- $\Sigma_j VC_i(e)[j]$ -1 represents the total number of events that causally precede *e* in the distributed computation.

Vector Clocks

$VC1 \leq VC2 \Leftrightarrow \forall i: VC1[i] \leq VC2[i]$

$VC1 < VC2 \Leftrightarrow VC1 \leq VC2$ **and** $\exists i: VC1[i] < VC2[i]$

$VC1 \parallel VC2 \Leftrightarrow \neg(VC1 < VC2)$ **and** $\neg(VC2 < VC1)$

Definitions

Vector Clocks

Strong clock condition

$$e \rightarrow f \Leftrightarrow VC(e) < VC(f)$$

Concurrent condition

$$e \parallel f \Leftrightarrow VC(e) \parallel VC(f)$$

An isomorphism exists between the set of partially ordered events and their vector timestamps.

Vector Clocks - dotted

If we know the occurrence sites i and j of the events

Simple strong clock condition (i≠j)

$e_i \rightarrow f_j \Leftrightarrow VC(e_i)[i] \leq VC(f_j)[i]$

Simple concurrent condition (i≠j)

$e_i \parallel f_j \Leftrightarrow VC(e_i)[i] > VC(f_j)[i]$ and $VC(f_j)[j] > VC(e_i)[j]$

Past and future cones of an event

An event e_j could have been affected only by all events e_i such that $e_i \rightarrow e_j$.

All such events e_i belong to the past of e_j.

Let Past(e_j) denote all events in the past of e_j in a computation $(H, \rightarrow)$

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

Past and future cones of an event

The future of an event e_j, denoted by Future(e_j), contains all events e_i that are causally affected by e_j.

In a computation $(H, \rightarrow)$

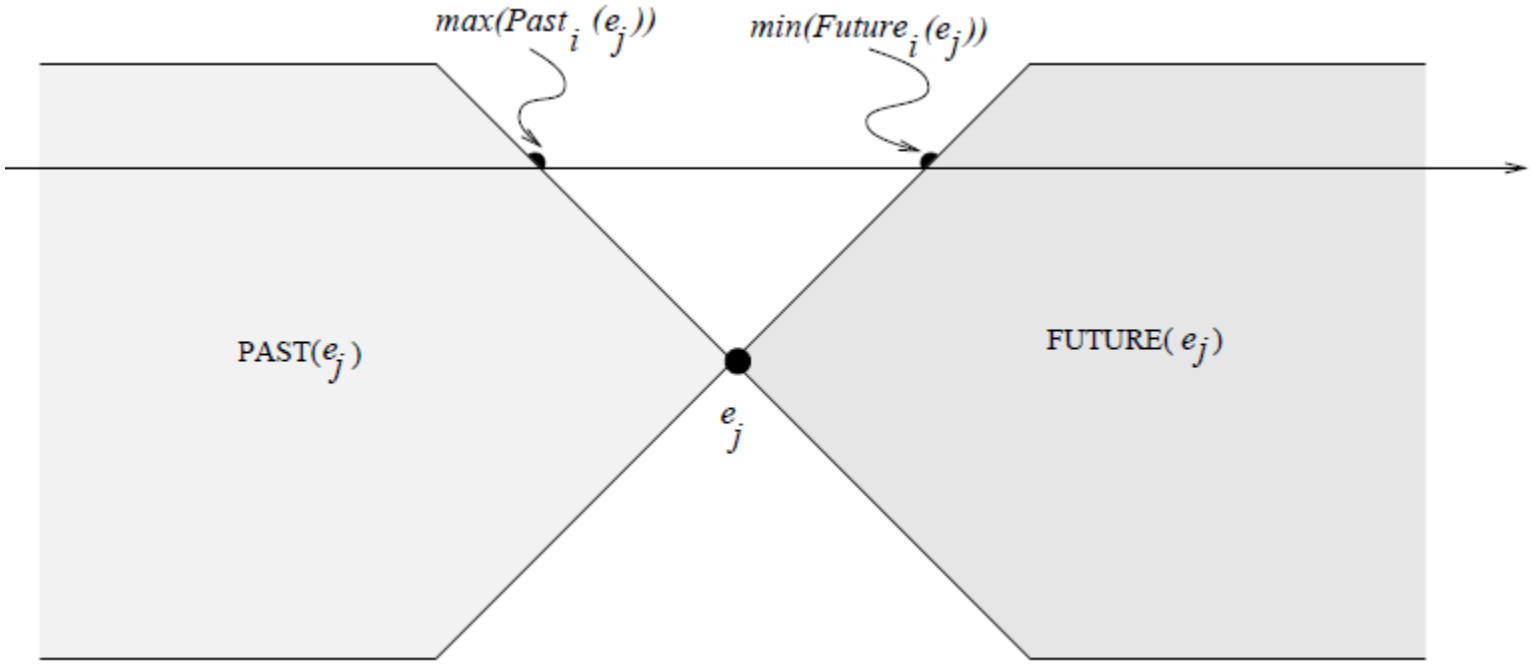
$$\text{Future}(e_j) = \{e_i \mid \forall e_i \in H, e_j \rightarrow e_i\}.$$

All and only those events of computation H that belong to the set

$$H - \text{Past}(e_j) - \text{Future}(e_j)$$

are concurrent with event e_j.

Surfaces of the past and future cones of an event



max(Past(e_j)) consists of the max(Past$_i$(e_j)) at every process p_i

min(Future(e_j)) consists of the min(Future$_i$(e_j)) at every process p_i

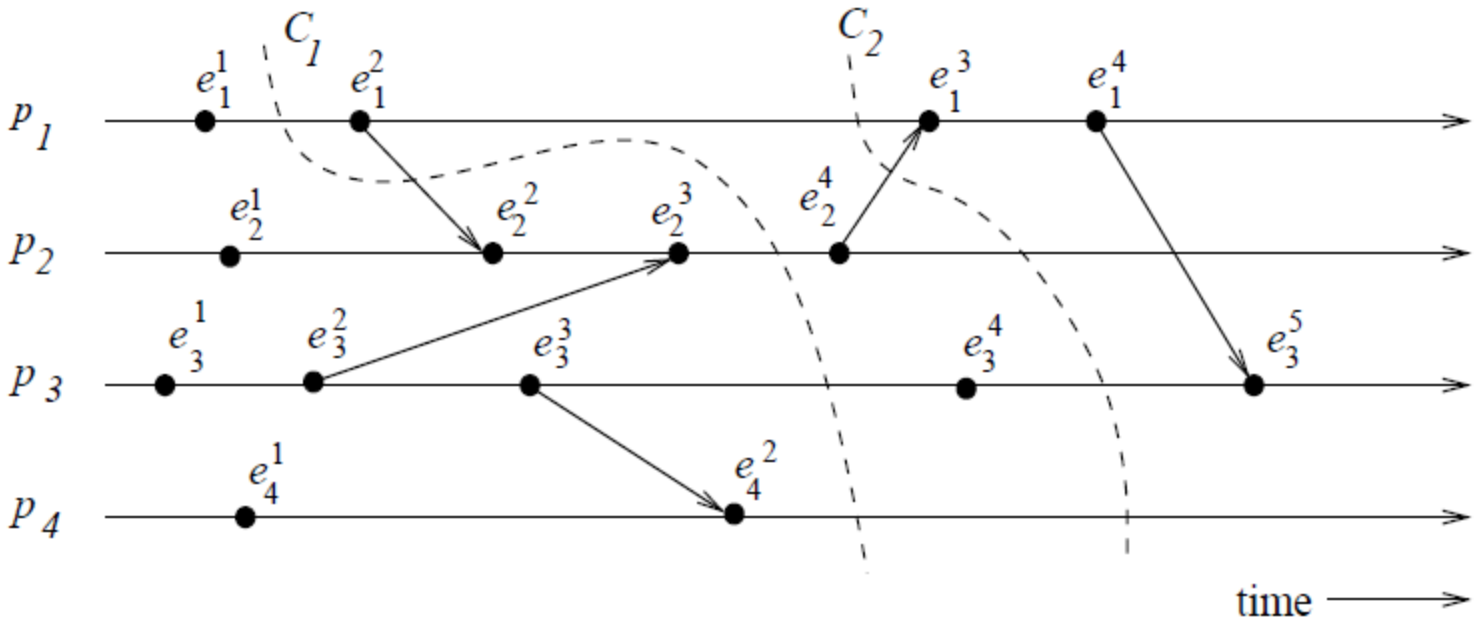
Cuts of a distributed computation

In a space-time diagram, a **cut** is a zigzag line joining one arbitrary point on each process line.

A cut slices the space-time diagram, and thus the set of events in the distributed computation, into a PAST and a FUTURE.

The PAST contains all the events to the left of the cut and the FUTURE contains all the events to the right of the cut.

Cuts of a distributed computation



Cuts of a distributed computation

Every cut corresponds to a global state and every global state can be graphically represented as a cut in the computation's space-time diagram.

Cuts in a space-time diagram provide a powerful graphical aid in representing and reasoning about global states of a computation.

Cuts of a distributed computation

In a consistent cut, every message received in the PAST of the cut was sent in the PAST of that cut.

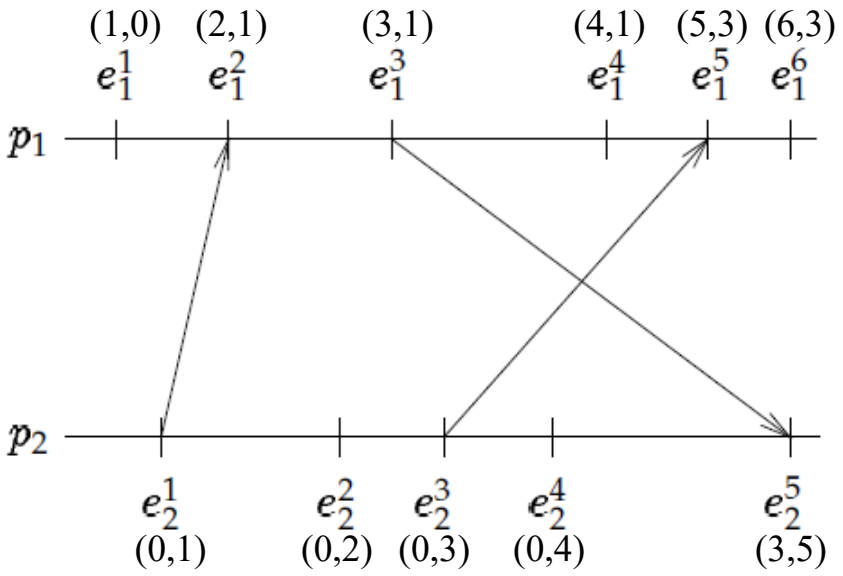
Cut C2 is a consistent cut

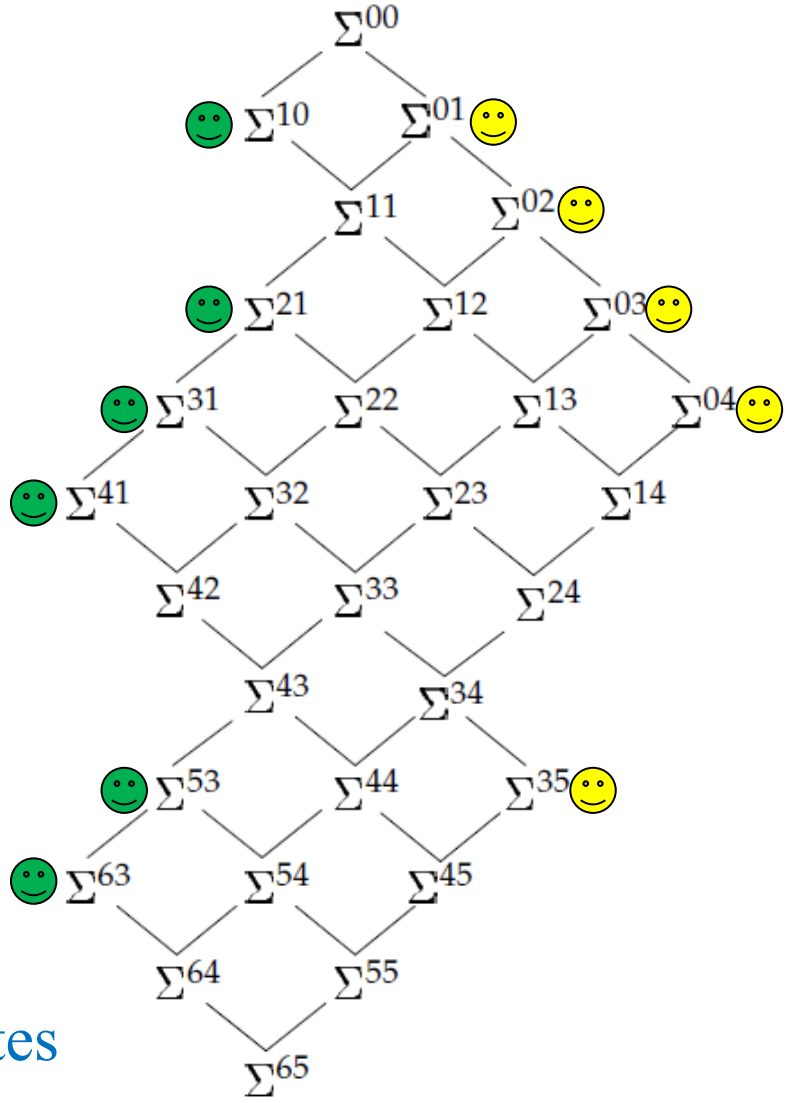
All messages that cross the cut from the PAST to the FUTURE are in transit in the corresponding consistent global state.

A cut is inconsistent if a message crosses the cut from the FUTURE to the PAST.

Cut C1 is an inconsistent cut

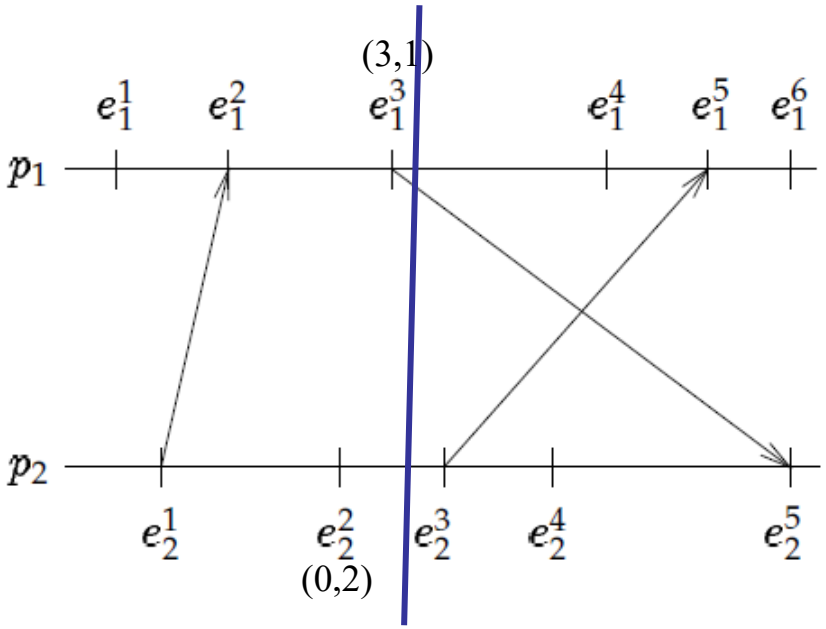
A Process is a “Local” Observer





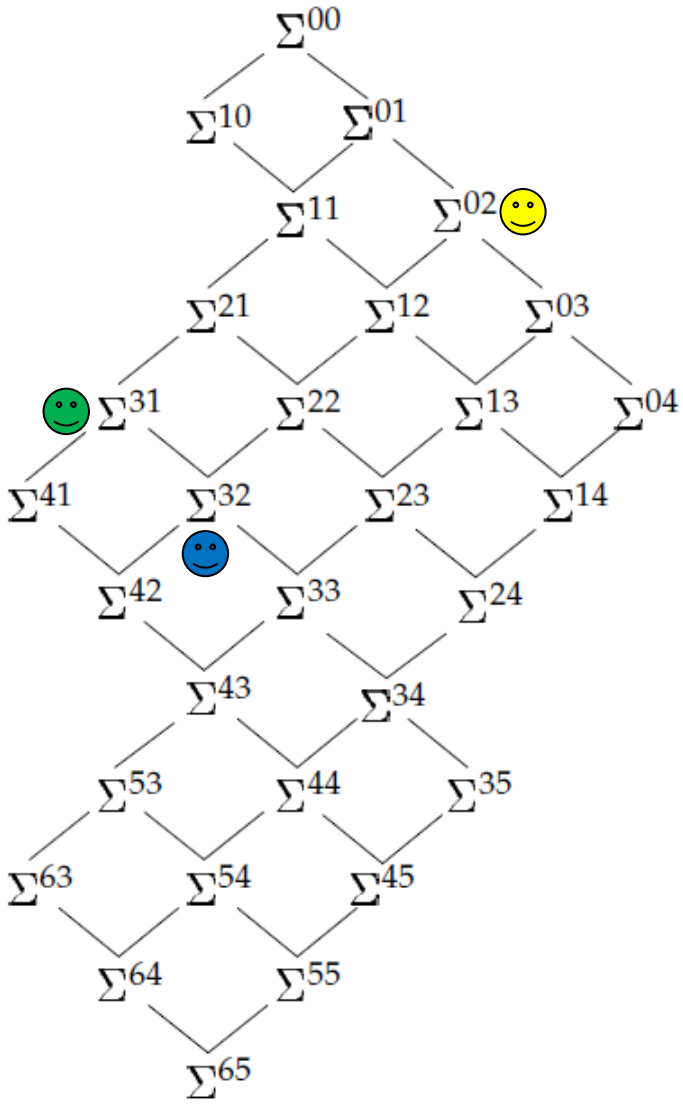
Lattice of consistent global states

A Vector Clock denotes a Global State

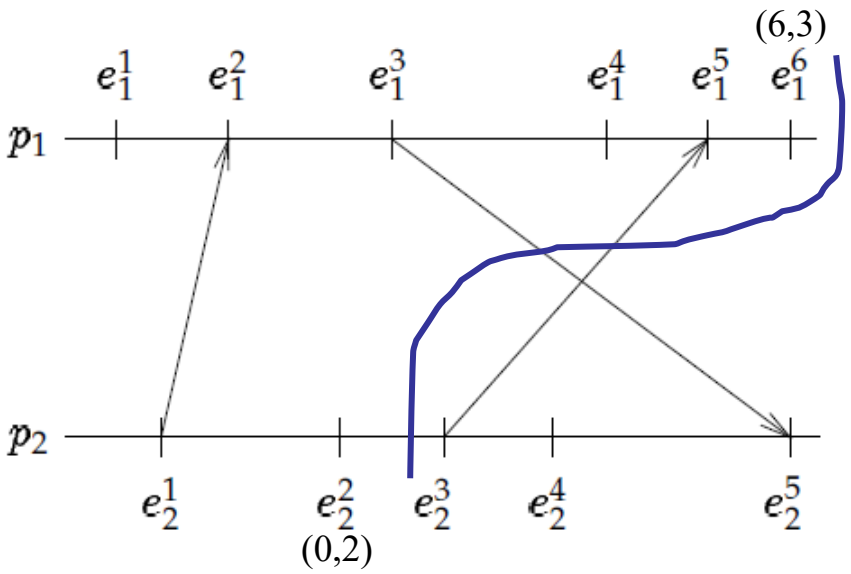


Consistent cut X

VC(X) = ((3,1)[1],(0,2)[2])
VC(X) = (3,2)

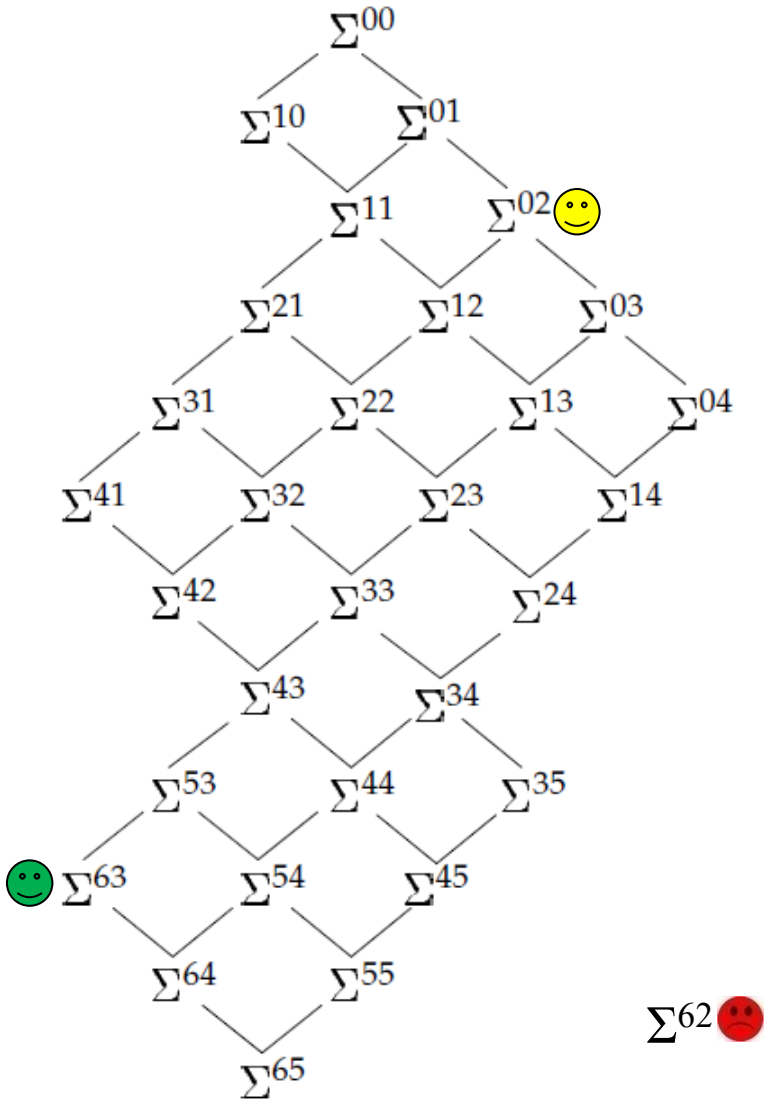


A Vector Clock denotes a Global State



Inconsistent cut Y

VC(Y) = ((6,3)[1],(0,2)[2])
VC(Y) = (6,2)

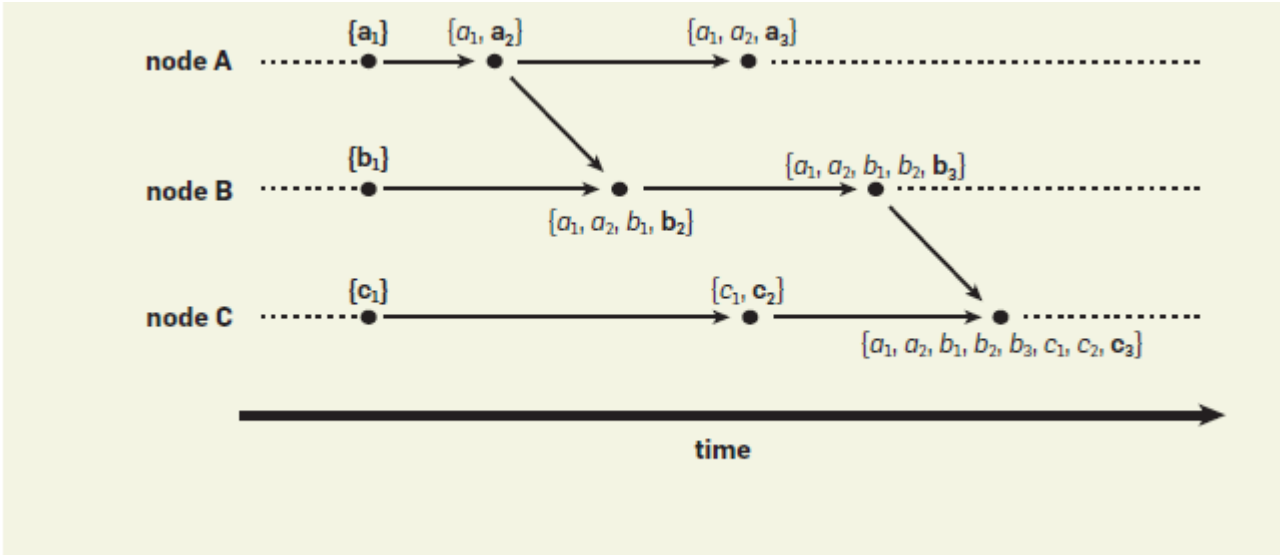


Vector Clocks in Action

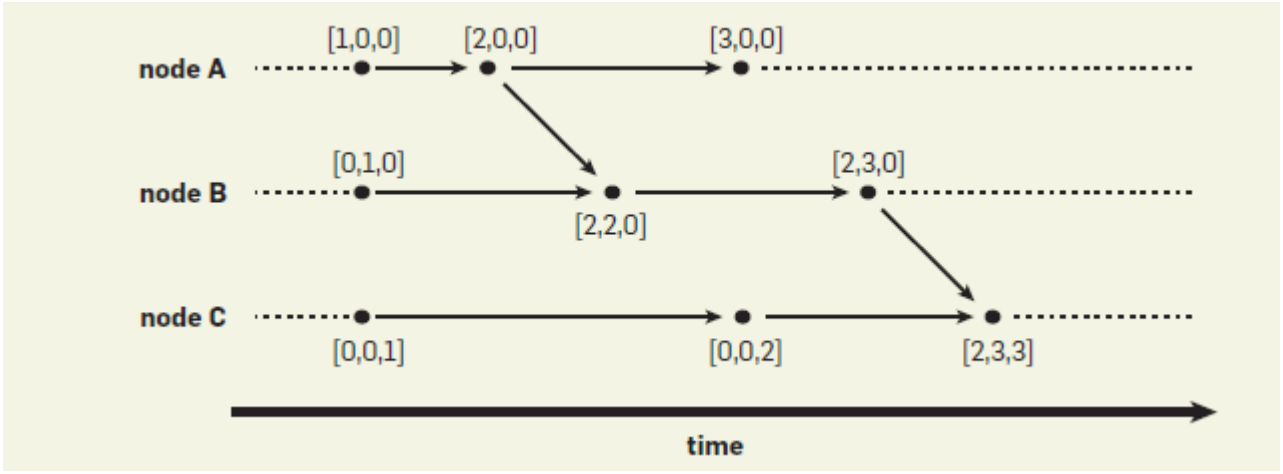
Schiper, A., Eggli, J., and Sandoz, A. (1989). A new algorithm to implement **causal ordering**. *Distributed Algorithms - Lecture Notes in Computer Science*, Vol. 392, pp. 219-232.

Alagar, S., and Venkatesan, S. (1995). An optimal algorithm for distributed snapshots with **causal message ordering**. *Information Processing Letters*, Vol. 50, No. 6, pp. 311–316.

Causal Histories and Vector Clocks

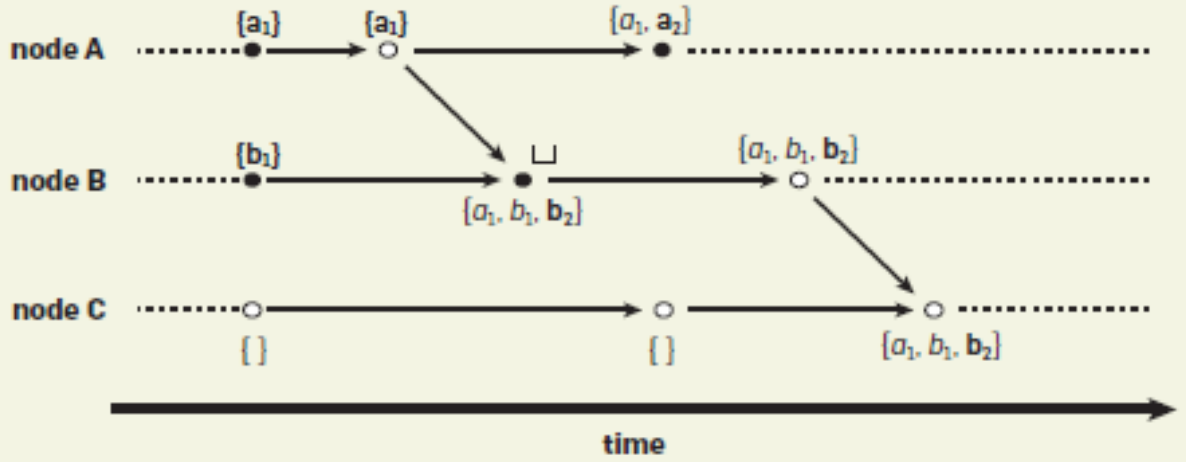


Causal histories

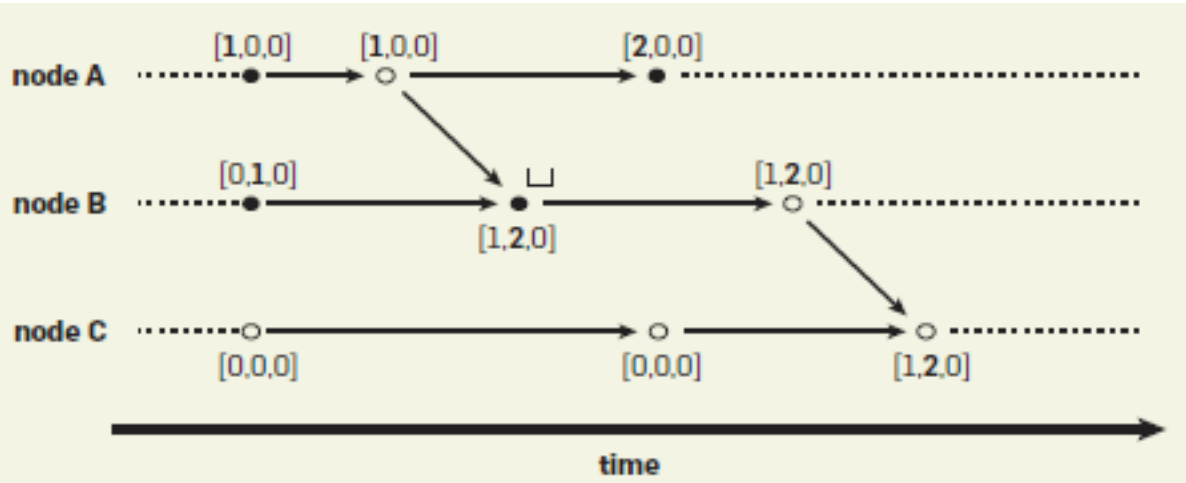


Vector clocks

Tracking only relevant events



Causal histories

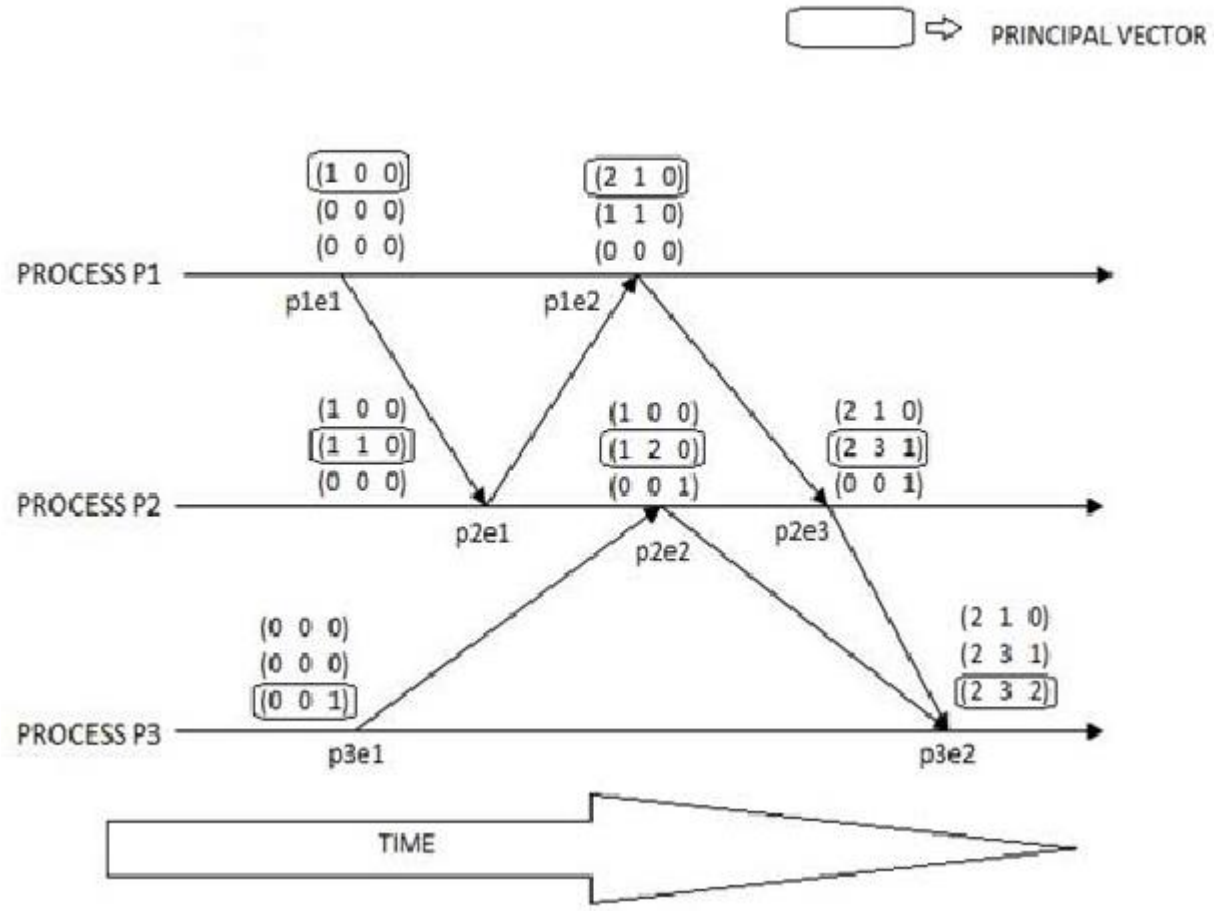


Version vectors (1983)

Logical Time

- Scalar clocks
- Vector clocks
- **Matrix clocks**

Matrix Clocks



Matrix Clocks

A process P_i maintains a matrix $MC_i[1..n, 1..n]$ denoting P_i's local view of the logical global time.

$MC_i[i,i]$ denotes the local logical clock of P_i and tracks the progress of the computation at P_i (number of events produced by P_i).

$MC_i[i,j]$ denotes the latest knowledge that P_i has about the local logical clock, $MC_j[j,j]$, of P_j.

$MC_i[j,k]$ represents what P_i knows about the latest knowledge that P_j has about the local logical clock, $MC_k[k,k]$, of P_k.

Matrix Clocks – Updating Rules

P_i uses the following rules to update its clock:

- ***R1 :*** When a local event occurs:
  - $MC_i[i,i] := MC_i[i,i] + 1$
- ***R2 :*** When a message M, timestamped MC_M, is received from P_j:
  - $MC_i[i,:] := \max(MC_i[i,:] , MC_M[j,:])$
  - for each $k \neq i$ $MC_i[k,:] := \max(MC_i[k,:];MC_M[k,:])$
  - $MC_i[i,i] := MC_i[i,i] + 1$

Matrix Clocks

The ith row of the matrix clock at P_i, indicated by MCi[i,:], is the vector clock VC_i.

The jth row of the matrix clock at P_i, indicated by MC_i[j,:], gives the latest vector clock value of P_j, as known to P_i.

Each row of a matrix clock exhibits all the properties of vector clocks.

The kth column of the matrix clock at process P_i, indicated by MC_i[:,k], gives the latest scalar clock values of P_k, i.e., MC_k[k,k], as known to each process in the system.

Matrix Clocks

Let $\min(MC_i[:,k]) = x$

- To P_i's knowledge, all the processes know that P_k's local time has progressed until x (xth event)
- This can be used by P_i to forget the events of P_k that are older than x+1

Let $\min(MC_i[:,i]) = x$

- To P_i's knowledge, all the processes know its local time has progressed until x
- This can be used by P_i to forget its past events that are older than x+1

Logical Time in Distributed Systems

Raynal, M.
Presentation, *IRISA, 112 p.*
ftp://ftp.irisa.fr/local/caps/DEPOTS/TrUE/ASR/Logical-time.pdf