Goals of the lecture

- Direct dependency clocks
- Pred and Succ functions
- Matrix clocks
- Properties of matrix clocks

Overhead of the vector clock algorithm

$m = \#$ of messages sent/recd by any process

$n = \#$ of processes

- Space overhead : $n \log m$

- Time overhead : $O(n)$

- Communication overhead : $n \log m$

Direct Dependency Clocks

- Main drawback of the vector clock algorithm
  - $O(N)$ integers with every message
- Direct Dependency Clocks require only one integer to be appended to each message
- Used in Lamport's algorithm for mutual exclusion

Algorithm

For any initial state s:

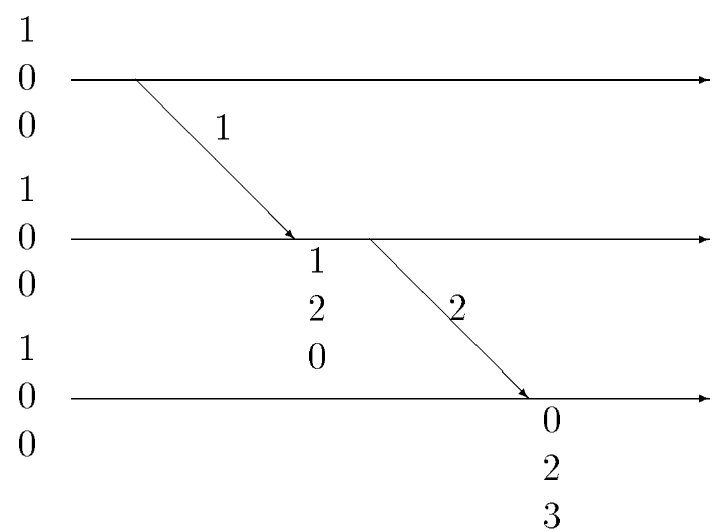
$$(\forall i : i \neq s.p : s.v[i] = 0) \wedge (s.v[s.p] = 1)$$

Rule for a send event (s, snd, t) or an internal (s, int, t):

$$t.v[t.p] := s.v[t.p] + 1;$$

Rule for a receive event $(s, rcv(u), t)$:

$$t.v[t.p] := \max(s.v[t.p], u.v[u.p]) + 1;$$
$$t.v[u.p] := \max(u.v[u.p], s.v[u.p]);$$



Properties of Direct Dependency Clocks

- projection on ith component results in the logical clock algorithm.

Lemma 1 $s \rightarrow t \;\Rightarrow\; s.v[s.p] < t.v[t.p]$

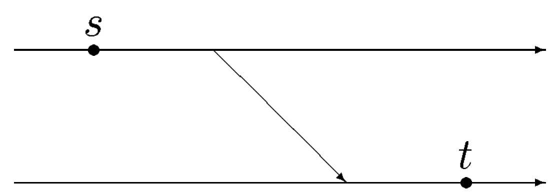
Converse ?

Lemma 2 $(\forall s, t :: s \not\rightarrow t \;\Rightarrow\; \neg(s.v[s.p] \leq t.v[s.p])$

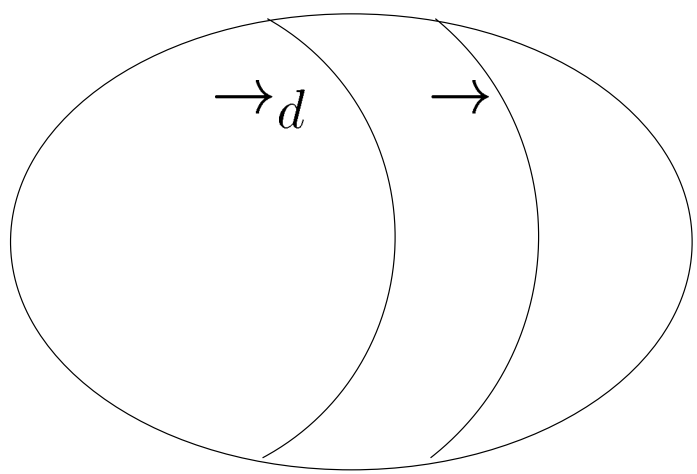
Converse ?

Direct Dependency

$$s \rightarrow_d t \equiv (s \prec t) \vee (\exists\, q, r : s \preceq q \wedge q \leadsto r \wedge r \preceq t)$$



Lemma 3 $\forall\, s, t : s.p \neq t.p : \neg(s \rightarrow_d t) \Rightarrow \neg(s.v[s.p] \leq t.v[s.p])$



Higher Dimensional Clocks

We describe a matrix clock and its properties below.

<u>To initialize:</u>

$M_k[\cdot,\cdot] := 0;$

$M_k[k,k] := M_k[k,k] + 1;$

<u>To send a message:</u>

Tag message with $M_k[\cdot,\cdot]$;

$M_k[k,k] := M_k[k,k] + 1;$ *increment local clock*

<u>Upon receipt of a message tagged with $W[\cdot,\cdot]$:</u>

for $i := 1$ **to** n **do**

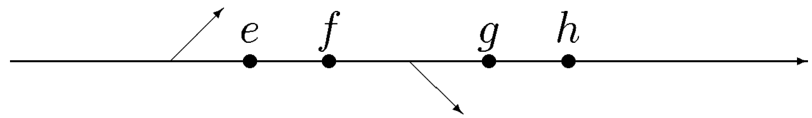
if $(M_k[i,i] < W[i,i])$ **then**

$M_k[i,\cdot] := W[i,\cdot];$ *copy vector clock for* $(i, W[i,i])$

$M_k[k,k] := M_k[k,k] + 1;$ *increment local clock*

$M_k[k,\cdot] := diagonal(M_k);$

Interval



$e \sim f$ iff there is no communication between the state e and f.

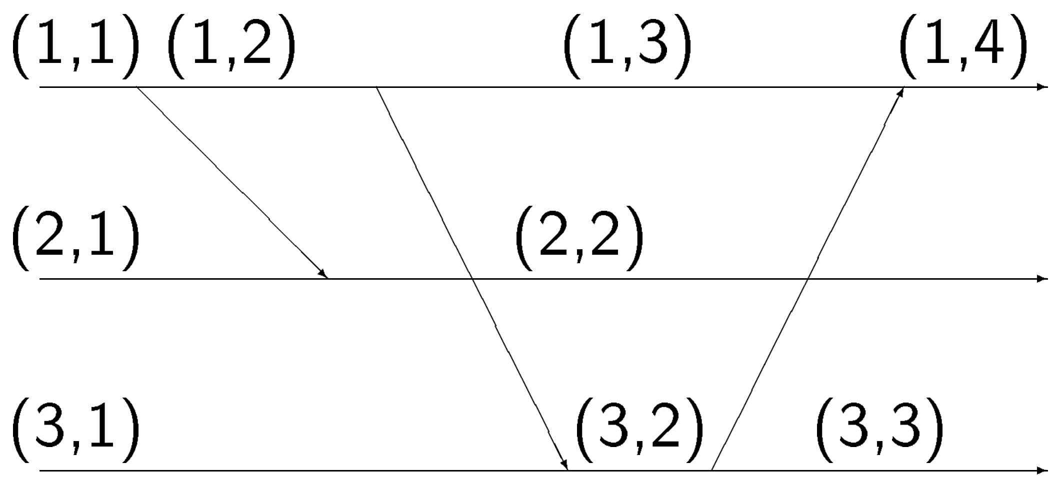
$\sim$ is an equivalence relation.

Further, $\sim$ is a congruence w.r.t. $\rightarrow$. That is,

$$s \sim s' \Rightarrow \forall u : u.p \neq s.p : (s \rightarrow u \equiv s' \rightarrow u) \wedge (u \rightarrow s \equiv u \rightarrow s'$$

Congruence exploited by assigning same identifier to all states in the same equivalence class (interval).

Interval [contd.]



Pred and Succ functions

- S_i = set of local states at P_i
- **Defn of Pred :** $pred.u.i = \max\{v \mid v \in S_i \ \wedge \ v \rightarrow u\}$
  returns $\perp$ if the above set is empty.

  Similar defn for Succ

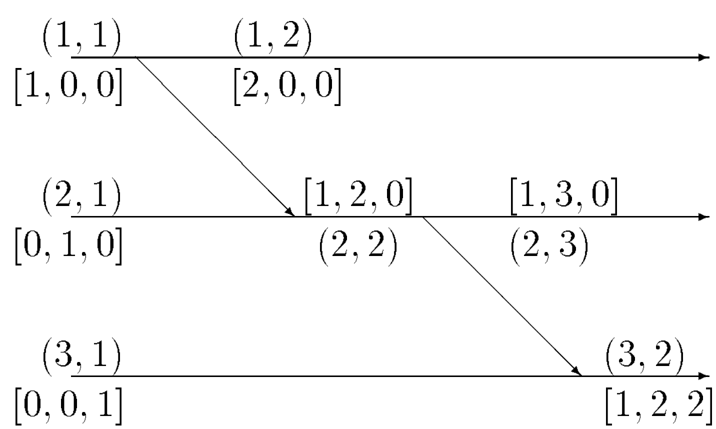
  Because of congruence, can use intervals instead of states.

Lemma 4 $(p, i) = pred.(q, j).p \ \Rightarrow \ (\forall\, k : k > i : (p, k) \not\rightarrow (q, j))$

For Vector Clocks

V_k^n = vector clock in the interval (k, n)

$$\forall i, k, n : k \neq i : pred.(k, n).i = V_k^n[i]$$

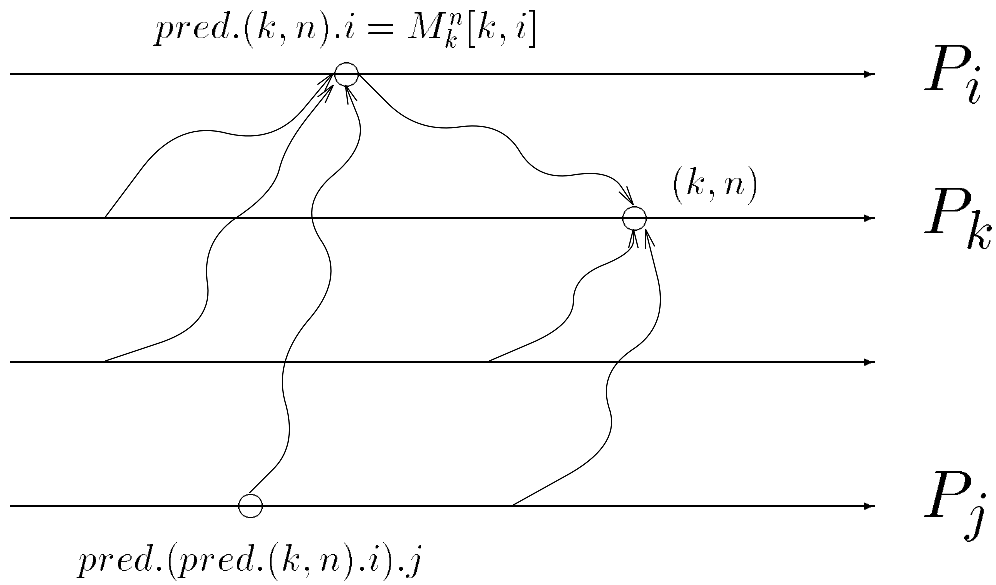


Properties of Matrix Clock

- For P_i, i^{th} row implements the vector clock
- $M_k^n[k, k] = n$
- Diagonal is same as the i^{th} row

Properties of Matrix Clock [Contd.]

- $i \neq j \;\Rightarrow\; (j, M_k^n[i,j]) = pred.(i, M_k^n[i,i]).j$
- $i \neq j \;\Rightarrow\; pred.(pred.(k,n).i).j = (j, M_k^n[i,j])$

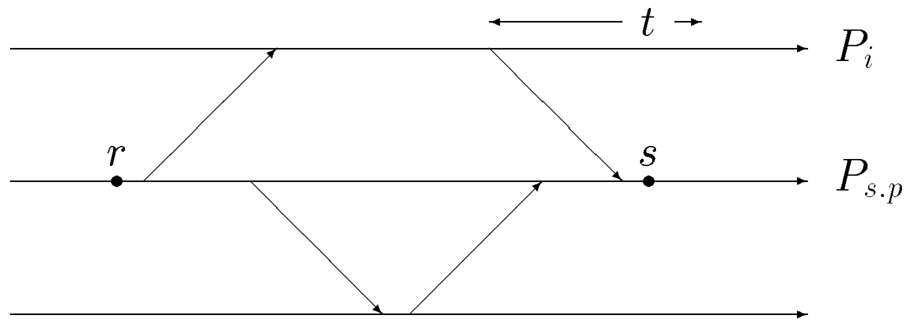


Discarding Obsolete Information

If process $s.p$ finds that its $s.p$ column is uniformly bigger than $r.v[s.p, s.p]$ (that is, its local clock in some previous state r), then the information at r has been received by all processes.

Lemma 5 *Let* $s.p = r.p$. *If* $(\forall i : s.v[i, s.p] \geq r.v[r.p, r.p])$, *then* $\forall t : t \| s : r \rightarrow t$.

Proof:



$$t \| s \Rightarrow pred(s).i \rightarrow t \quad (1)$$

$$(\forall i : s.v[i, s.p] \geq r.v[r.p, r.p]) \Rightarrow r \preceq pred.(pred(s).i).(s.p) \quad (2)$$

From the above two equations: $r \rightarrow t$. ∎