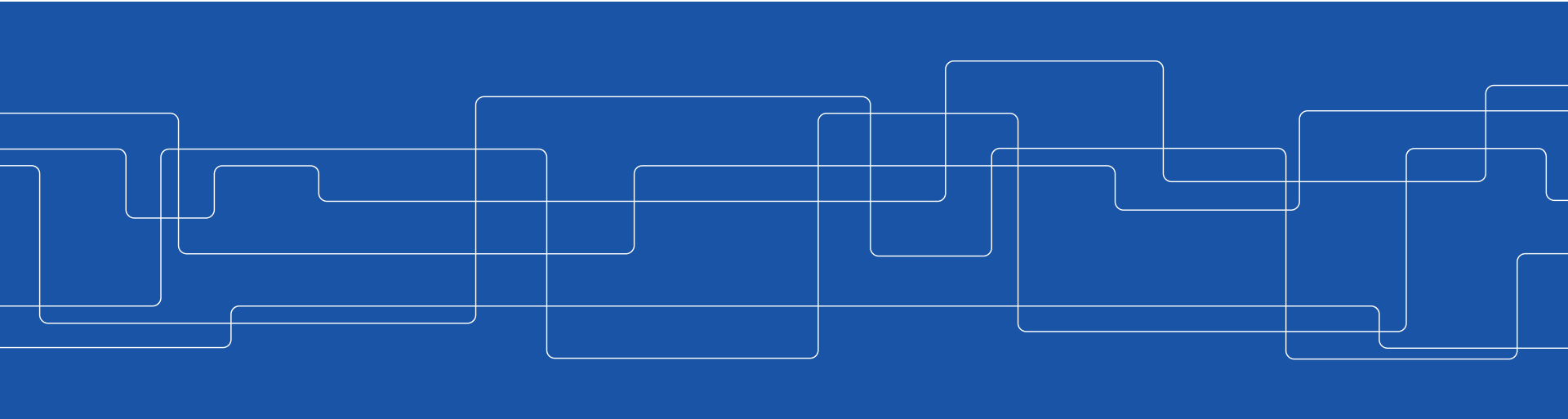
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Time

Vladimir Vlassov and Johan Montelius



Time

Why is time important?

The clock is not enough

In an asynchronous system, clocks can not be trusted entirely.

Nodes will not be completely synchronized.

We still need to:

- talk about before and after
- order events
- agree on order

Logical time

All events in one process are ordered.

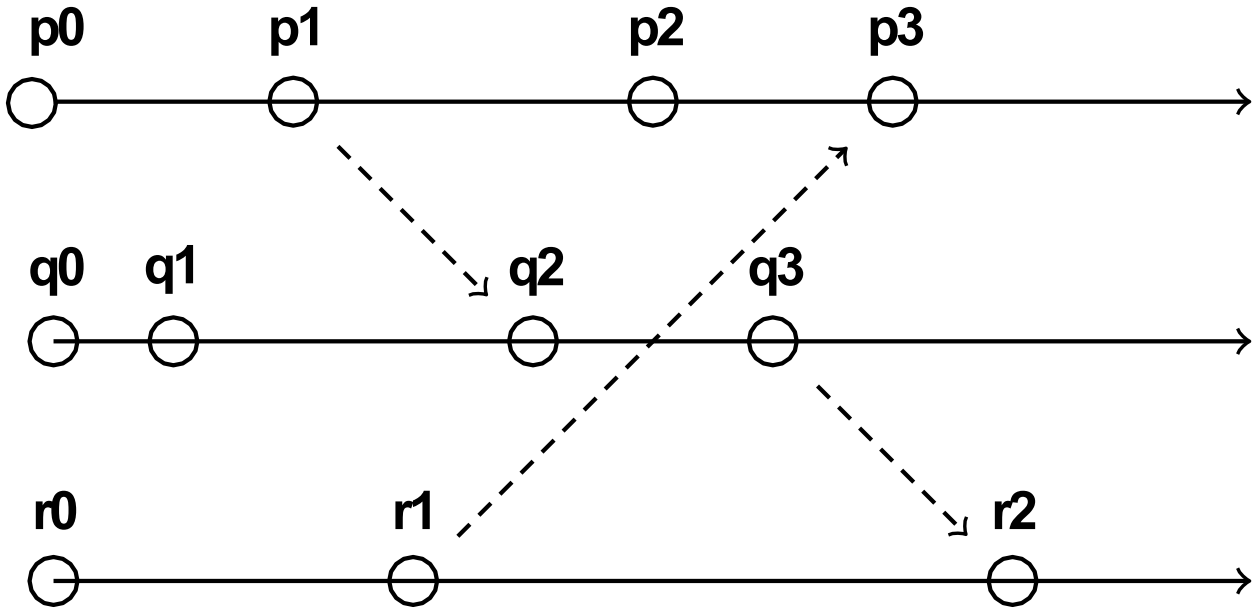
The sending of a message occurs before the receiving of the message.

Events in a distributed system are partially ordered.

The order is called ***happened before***.

Logical time gives us a tool to talk about ordering without having to synchronize clocks.

Partial order



Lamport clock

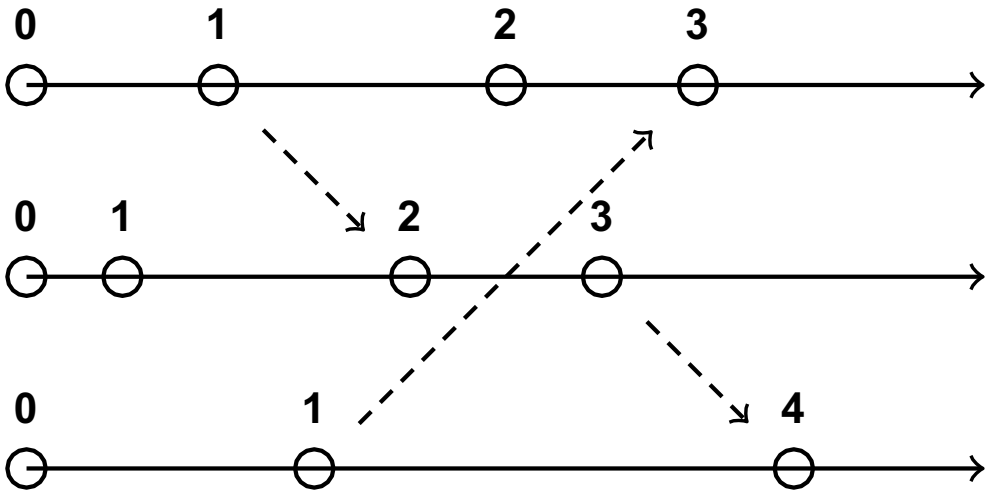
One counter per process:

- initially set to 0
- each process increments only its clock
- sent messages are tagged with a timestamp

Receiving a message:

- set the clock to the greatest of the internal clock and the time stamp of the message

Lamport clock



If *e1* **happened before** *e2*, then the time stamp of *e1* is less than the time stamp of *e2*.

e1 happened-before e2 → L(e1) < L(e2)

What do we know if the time stamp of *e1* is less than the time stamp of *e2*?

Let’s play a game

DON’T VIOLATE THE “HAPPEND BEFORE” ORDER!

Can we do better

We should be able to timestamp events to capture the partial order.

We want to look at two timestamps and say:

If the time stamps are ordered, then the events are ordered

$T(e1) < T(e2) \rightarrow e1$ *happend-before* $e2$

Vector clock

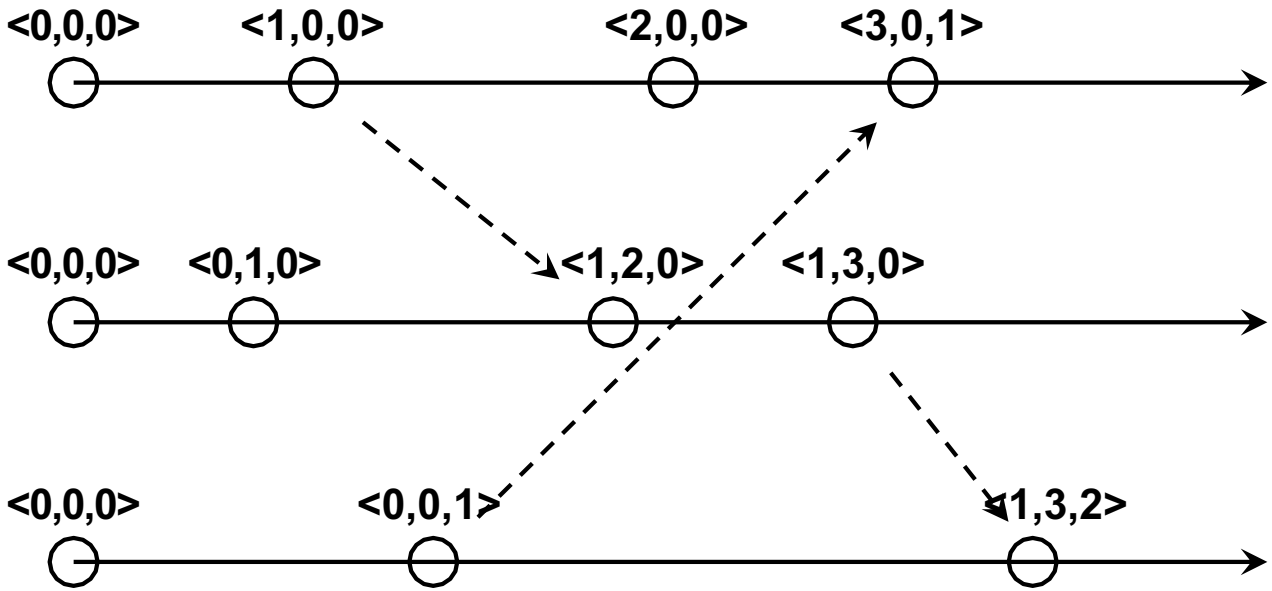
A ***vector*** with one counter per process:

- initially set to <0,....>
- each process increments only its index
- sent messages are tagged with a vector

Receiving a message:

- **merge** the internal clock and the time stamp of the message

Vector clock



V (e1) < V (e2) → e1 happend-before e2

How do we define < over vector clocks?

Compare vector timestamps

Vector timestamps can be compared as follows

- $V = V^*$ *iff* $V[j] = V^*[j]$ *for* $j = 1, 2, \ldots, N$
- $V \leq V^*$ *iff* $V[j] \leq V^*[j]$ *for* $j = 1, 2, \ldots, N$
- $V < V^*$ *iff* $V \leq V^* \wedge V \neq V^*$

V (e1) < V (e2) → e1 happend-before e2

If neither $V(e1) \leq V(e2)$ nor $V(e1) \leq V(e2)$ then the *e1* and *e2* are *concurrent*.

Pros and cons

The partial order is complete; we can look at the time stamp and determine if two events are ordered.

The vectors will take up some space and could become a problem.

What should we do if more processes come and leave? There is no easy mechanism to add new clocks to the system.

Vector clocks could be overkill.

Summary

We have to use something else if we can not trust real clocks to be synchronized.

Logical time captures what we need:

- Lamport clock: sound
- Vector clock: complete

Implementation issues:

- do we have to timestamp everything
- how do we handle new processes