

# Times, Clocks, and the Ordering of Events in a Distributed System

Jacopo Notarstefano

`jacopo.notarstefano [at] gmail.com`

13 February 2013

# Main objective

In this paper Lamport discusses the partial ordering defined by the “happened before” relation, and gives a distributed algorithm for extending it to a consistent total ordering of all the events.

# Distributed system

## Definition (Distributed system)

A *distributed system* consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process.

# Examples of distributed systems

- A worldwide network of interconnected computers
- A cluster of workstation in a data center
- Processes on a single computer

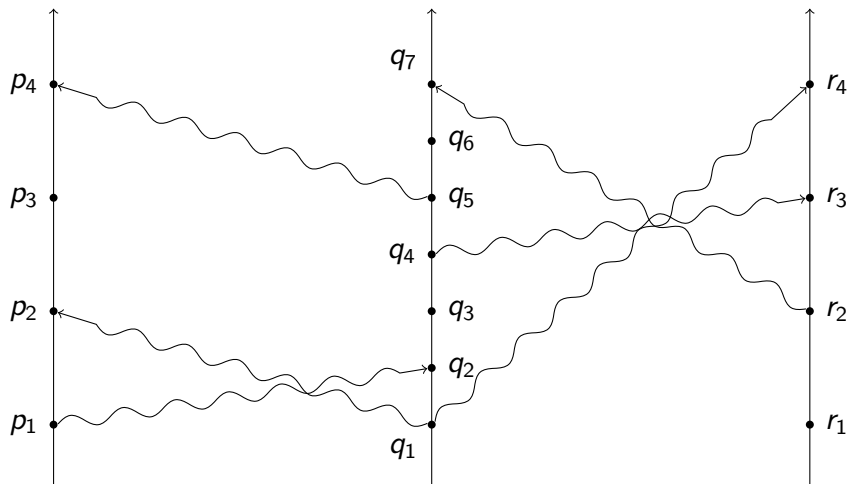
# The “ $\rightarrow$ ” relation

## Definition (The “ $\rightarrow$ ” relation)

The “ $\rightarrow$ ” relation on the set of events of a system is the smallest relation satisfying the following three conditions:

- 1 If  $a$  and  $b$  are events in the same process, and  $a$  comes before  $b$ , then  $a \rightarrow b$ .
- 2 If  $a$  is the sending of a message by one process, and  $b$  is the receipt of the same message by another process, then  $a \rightarrow b$ .
- 3 If  $a \rightarrow b$ , and  $b \rightarrow c$ , then  $a \rightarrow c$ .

# The “space-time diagram”



# Clocks

## Definition (Clock)

For each process  $P_i$  we define a *clock*  $C_i$  to be a function that assigns a number  $C_i\langle a \rangle$  to each event  $a$  in the process.

## Definition (System of clocks)

A *system of clocks* is a function  $C$  that assigns to the event  $b$  in process  $P_j$  the time  $C\langle b \rangle = C_j\langle b \rangle$ .

# The clock condition

## Definition (The clock condition)

We say that a system of clocks satisfies the *clock condition* if, for any events  $a$  and  $b$ , we have: if  $a \rightarrow b$  then  $C\langle a \rangle < C\langle b \rangle$ .

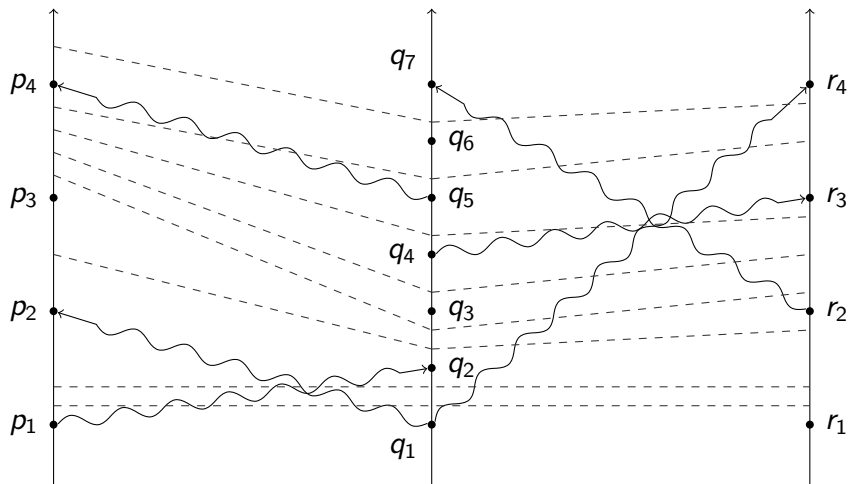
## Lemma

*The clock condition is satisfied if the following conditions hold:*

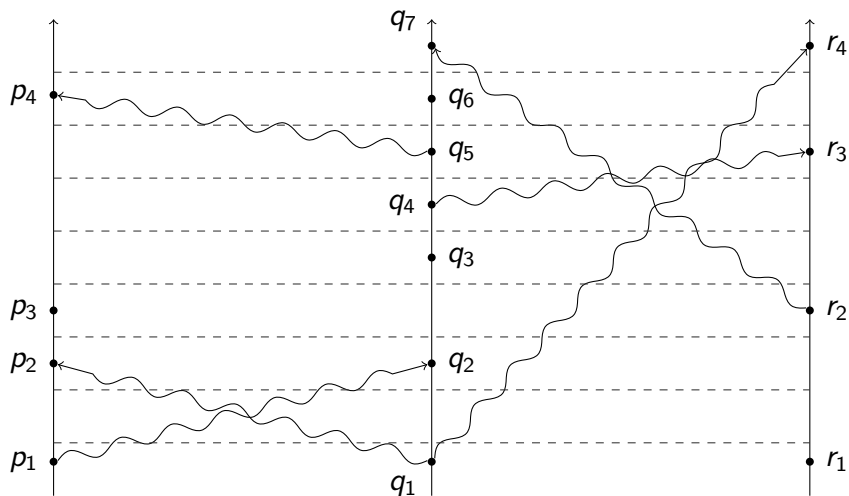
- 1 If  $a$  and  $b$  are events in process  $P_i$  and  $a$  comes before  $b$ , then  $C_i\langle a \rangle < C_i\langle b \rangle$ .
- 2 If  $a$  is the sending of a message by process  $P_i$  and  $b$  is the receipt of that message by process  $P_j$ , then  $C_i\langle a \rangle < C_j\langle b \rangle$ .



# The “space-time diagram”, revisited



# The “space-time diagram”, rearranged



# Implementation of the clock condition

## Lemma

*To guarantee that the system of clocks satisfies the clock condition we need to obey the following implementation rules:*

- 1 *Each process  $P_i$  increments  $C_i$  between any two successive events.*
- 2 *If event  $a$  is the sending of a message  $m$  by process  $P_i$ , then the message  $m$  contains a timestamp  $T_m = C_i\langle a \rangle$ .*
- 3 *Upon receiving a message  $m$ , process  $P_j$  sets  $C_j$  greater than or equal to its present value and greater than  $T_m$ .*

# The “ $\Rightarrow$ ” relation

## Definition (The “ $\Rightarrow$ ” relation)

Let  $\prec$  be a total ordering on the processes. If  $a$  is an event in process  $P_i$  and  $b$  is an event in process  $P_j$ , then  $a \Rightarrow b$  if and only if either

- 1  $C_i\langle a \rangle < C_j\langle b \rangle$  or
- 2  $C_i\langle a \rangle = C_j\langle b \rangle$  and  $P_i \prec P_j$ .

# A mutual exclusion problem

A fixed collection of processes share a single resource, which can be used by one process at a time. We want to find an algorithm that satisfies:

- 1 A process which has been granted the resource must release it before it can be granted to another process.
- 2 Different requests for the resource must be granted in the order in which they are made.
- 3 If every process which is granted the resource eventually releases it, then every request is eventually granted.

# A wrong solution