Distributed System 1 - synch example

Francesco Penasa

February 27, 2020

`solo on 26/02/2020`

1 Totally ordered Multicast

to solve inconsistent problems we need to make sure that everybody agrees on the same order. **totally ordered multicast** delivers messages in the same global order.

1.1 Totally ordered multicast with scalar clocks

Assumptions reliable nodes/links and FIFO links (TCP)

Algorithm outline

1. Messages are sent simultaneously to all nodes
2. all messages carry a timestamp with the sender's (scalar) clock
3. receivers store all messages in a queue, ordered according to its (logical) timestamp
4. a message is delivered to the application only when it is the highest in the queue and all of its acknowledgments have been received. (message delivered when the queue is completed)

Causal history of an event e is the set of events that happen-before e, plus e itself

Vector clock associated to event e is a n-dimensional vector $VC(e)$ such that

$$VC(e)[i] = c_i \quad where \quad \Theta_i(e) = h_i^{c_i}$$

n = number of processes h = the causal history over process p_i

if you look at the example it is simple, every process memorizes the time of every other process.

2 exercises

2.1 First exercise

fatto sul quaderno

2.2 Secondo exercise

1. P_1 $receive(m4) == P_1$ $receive(m2)$ both have the same clock but are two distinct events
2. P_2 $receive(m4) \neq (6, 5, 5)$
3. P_3 $send(m2) \neq (6, 4, 4)$

3 Global sate

local state denoted as σ_i^k after the execution of event e_i^k

global state n-tuple of local states, one for each process $\Sigma = (\sigma_1, \ldots, \sigma_n)$