

Distributed Algorithms

Causal & Total Order Broadcast
3rd exercise session

Matteo Monti <matteo.monti@epfl.ch>
Jovan Komatovic <jovan.komatovic@epfl.ch>

Exercise 1

Would it make sense to add the total-order property to the best-effort broadcast?

Exercise 2

What happens in our "Consensus-Based Total-Order Broadcast" algorithm, if the set of messages delivered in a round is not sorted deterministically after deciding in the consensus abstraction, but before it is proposed to consensus?

What happens in that algorithm if the set of messages decided on by consensus is not sorted deterministically at all?

upon event $\langle tob, Init \rangle$ **do**

$unordered := \emptyset;$

$delivered := \emptyset;$

$round := 1;$

$wait := FALSE;$

upon event $\langle tob, Broadcast \mid m \rangle$ **do**

trigger $\langle rb, Broadcast \mid m \rangle;$

upon event $\langle rb, Deliver \mid p, m \rangle$ **do**

if $m \notin delivered$ **then**

$unordered := unordered \cup \{(p, m)\};$

upon $unordered \neq \emptyset \wedge wait = FALSE$ **do**

$wait := TRUE;$

Initialize a new instance $c.round$ of consensus;

trigger $\langle c.round, Propose \mid unordered \rangle;$

upon event $\langle c.r, Decide \mid decided \rangle$ **such that** $r = round$ **do**
 // by the order in the resulting sorted list

forall $(s, m) \in sort(decided)$ **do**

trigger $\langle tob, Deliver \mid s, m \rangle;$

$delivered := delivered \cup decided;$

$unordered := unordered \setminus decided;$

$round := round + 1;$

$wait := FALSE;$

Exercise 3

The "Consensus-Based Total-Order Broadcast" algorithm transforms a consensus abstraction (together with a reliable broadcast abstraction) into a total-order broadcast abstraction.

Describe a transformation between these two primitives in the other direction, that is, implement a (uniform) consensus abstraction from a (uniform) total-order broadcast abstraction.