Distributed Algorithms

Philipp Marian Grulich,
Lukas Wiegmann

Exercise Sheet 3

Exercise 3.1: Vector Clocks

i. Causal Order

Each message shall be delivered to all processes using the broadcast algorithm. With delivery of the message shall be ensured that the messages are delivered in an order satisfying causality. It would not be sufficient to use the vector as applied by the causal broadcast to achieve causal order because P_i only increments V_j [i] if it sends a message. A message sent by P_i is only delivered to P_j when the time stamp T fulfills the following conditions: $T[i] = V_j[i] + 1 \land \forall\, k \neq i: T[k] \leq V_j[k]$. It is sufficient to increase V_j [i] by one at delivery. Not sending messages to all nodes does not match with the delivery conditions and a causal order could not be achieved by only using the vector.

ii. Order Relation

1. A logical clock assigns a time stamp C(e) to each event e. The logical time stamp C(e) defines a partial order on the set of events: $e_1 < e_2 \Leftrightarrow C(e_1) < C(e_2)$.

2. According to the lecture the clock condition it the requirement to a logical clock. For all events a, b shall apply: $a \rightarrow b \Rightarrow C(a) < C(b)$. Thus, the clock preserves the causal order of the events. Each P_i has a local logical clock L_i, whose value is adopted at the occurrence of the following events: local event with process P_i, P_i sends a message, P_i receives a message. Each process P_i manages a counter C_i, that is increased by one when an event e occurs. The event gets the new value as a logical time stamp. The respective logical clock fulfills the clock condition because V(e) defines its vector time stamp and whenever an event e occurs, V(e) is computed as introduced in the lecture. This means that each process P_i holds a vector time stamp V_i consisting of n counters that are initially all zero. There are different updates of the vector time stamp:

    - update on local event: If an event occurs in a process P_i , it increments the i-th component of its vector,
    - update when sending a message: If P_i sends a message, the new version of V_i is sent along (piggybacked),
    - update when receiving a message: If P_i receives a message with vector time stamp T, it forms the maximum of the new version of V_i and T, component-by-component.

3. The partial order can easily be extended to a total order through the usage of unique process identities as tiebreaker.

    - The vector time stamp V'(e_j) of an event e_j is a pair $e_1 < e_2 \Leftrightarrow V'(e_1) < V'(e_2) \Leftrightarrow V_1 < V_2 \lor V_1 = V_2 \land P_1 < P_2$
    - since the process identities are unique, for two arbitrary events $e_1 \neq e_2 \rightarrow e_1 < e_2 \lor e_2 < e_1$