Distributed Algorithms

Homework 3

Group G10

Piotr Mrowczynski 387521
Gabriel Vilén 387555
Stefan Stojkovski 387529
Tong Li 387568

Exercise 3.1: Vector Clocks

i. Causal Order:

The concept of the "causal broadcast" has been introduced in the lecture as an application of vector clocks. In order to guarantee the proper execution of the algorithm it was necessary to send each message to all participating nodes. If we want to avoid sending each message to all nodes, why wouldn't it be sufficient to only use the vector as applied by the causal broadcast to achieve causal order?

Answer:

The causal broadcast is introduced to reduce the asynchrony of communication channels as perceived by application processes. The order in which messages are delivered to application processes cannot violate the precedence order of the corresponding broadcast events. Which means that when a message m is delivered to a process, all messages whose broadcasts causally precede the broadcast of m, have already been delivered to that process.

As each message will be send to all participating nodes, it will be ensured that the messages are delivered in an order satisfying causality. Each process will have a table of vector clock and will wait for the gap to be filled.

If we use vector only without broadcast, we don't have all message of the past on the process, then the process will have no idea of the past, then we can't ensure all the previous message are delivered to the target process and will have the message delivered out of order.

ii. Order Relation

If e is a system event, then V (e) defines its vector timestamp and C(e) $\in$ N is the corresponding sum of the vector's components. Whenever an event e occurs, V (e) is computed as introduced in the lecture.

1. Show that C(e) together with the "less than"relation (<) defines a partial order of the set of system events.

Answer:

1). If *a* and *b* are events in the same process P_i, and *a* comes before *b*, then C_i(a) < C_i(b).

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(a) < C_j(b).

2. Show that the respective logical clock fulfills the clock condition.

Answer:

1). Each process P_i increments the i-th component of its vector V_i between any two successive events. C_i get increase too.

2). If event *a* is the sending of a message *m* by process P_i,then the message *m* contains a timestamp T_m = V_i(a).

3). Upon receiving a message *m*, process P_j sets V_j greater than or equal to its present value and greater than T_m, component-by-component. C_j get larger than C_i.

3. Show that the partial order can be extended to a total order by applying process IDs together with the vector timestamp.

Answer:

To break ties, we use any arbitrary total ordering < of the processes, that's the process IDs. More precisely, we define a relation ⇒ as follows: if *a* is an event in process $P_i$and *b* is an event in process $P_j$then a⇒b if and only if either

(i) V_i(a) < V_j(b)

or

(ii) V_i(a) = V_j(b) and $P_i < P_j$

It is easy to see that this defines a total ordering, and that the Clock Condition implies that if a→b, then a⇒b. In other words, the relation ⇒ is a way of completing the "happened before" partial ordering to a total ordering.