Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Philipp Bamberger

Distributed Systems, Summer Term 2019
Exercise Sheet 1

1. Happens Before in Shared Memory

Given are n processors and m shared variables. Every processor can access every shared variable with atomic read and write operations. Define a *happens before* relation similar to the one for message passing.

Sample Solution

Given some schedule S. Happens before relation contains:

1. All events (e, e') where e precedes e' in S and e and e' are events of the same process
2. All events (e, e') where e precedes e' in S and which access the same shared variable i and one of the events accessing variable i in between e and e' (including e and e') is a write event.

Now take the transitive closure of the relation defined by 1 and 2

2. Unique Maximal Cut Preceeding a Given Cut

Given a schedule S with a cut C. Show that there is a unique consistent cut C' of S which precedes the cut C.

Sample Solution

Proof Sketch: Start with the (non-consistent) cut C. Whenever there is an unmatched receive event in one local view remove it and all proceeding events in that local view from the cut. This process terminates and will succeed in the unique consistent maximal cut.
The defined cut precedes the cut C and it is consistent because it does not contain any unmatched receive event (Lemma in the lecture). It is maximal because if it is increased (within C) it will be increased by a receive event contradicting the consistency.
It is easy to see that the maximal cut is unique. Let C' and C'' be two maximal consistent cuts preceding cut C and consider the intersection $D = C' \cap C''$ of both cuts which is also a consistent cut preceding C. Assume that there is some event e in $C' \setminus D$ or $D \setminus C''$. W.l.o.g. let $e \in C'$ and $e \notin C''$. Then $C'' \cup \{e\} \cup \{e' \mid e' \Rightarrow_S e\}$ is a consistent cut preceding C and strictly larger than C'', a contradiction. Hence $C' = C''$.

3. Happens Before Relation

Let S be a schedule with events a, b and c. Show that if $a \not\Rightarrow_S b$ and $a \not\Rightarrow_S c$ holds, then there exists some causal shuffle S' of S in which b and c occur before a.

Sample Solution

The formal proof is a little bit tedious. The idea is the same as in the lecture Causality part I, slide 12ff. The proof is easiest to be understood if you draw the same picture as given on the slides.
For every node i let $e_i \in S|_i$ be the event s.t. $a \Rightarrow_s e_i$ and for all $e \in S|_i$ with $a \Rightarrow_S e$ we have $e_i \Rightarrow_S e$. (e_i might not exist for some i, but then this local view does not matter.)
Let i_a, i_b and i_c be the indices of the local views in which a, b and c occur and associate global (discrete) times $\iota(e)$ with every event $e \in S$ s.t. no two events have the same time and the time is consistent with the schedule S. Note that $e_{i_a} = a$.
Define $\Delta = \max\{\iota(e_{i_b}), \iota(e_{i_c})\} - \iota(e_{i_a})$. Now we obtain a causal shuffle of S by shifting all elements in view i with $e_i \Rightarrow_S e$ by $\Delta + 1$ time units (to the later time). We can deduce a new schedule from these times which is a causal shuffle of S with the desired properties.

Exercise 4: Logical Clocks

You are given the clique graph on n nodes. Find two executions A and B, in which each node sends exactly one message to every other node, such that

- the largest Lamport clock value in A is as small as possible and
- the largest Lamport clock value in B is as large as possible.

Solution

At every node there are 1 send and $n-1$ receive events, so n is a lower bound on the clock value. This bound is matched if all messages are sent before any message is received.
The largest clock value is as large as possible if the messages are sent one after another along an Euler tour of the graph. Such a tour does always exist because every node v has $deg_{in}(v) = deg_{out}(v) = n-1$. With every message accross a (directed) edge the clock value is increased by one. So the maximal clock value will be $\#directed\ edges = n^2 - n$.

5. Clock Synchronisation

Model:

- n processors connected in a path.
- Hardware clock H_v for each processor, increments are in an interval of $[1-\rho, 1+\rho]$ (clock drift),
- Logical clock L_v for each processor,
- Message delays are in the interval $[0, T]$.

The following (simple) algorithm executed at every processor synchronizes the logical clocks according to hardware clocks and messages from neighbors.
Algorithm: If no messages are received the logical clock runs as fast as the hardware clock and the logical clock values are forwarded in every time unit.
If messages from neighbors are received set the logical clock to the maximum of the clock values received from neighbors and the own clock value; forward new clock values to all neighbors immediately.
Exercise: Give an execution which shows that the maximal skew (the difference between the maximum and minimum (logical) clock value) $(n-1)T$ can be obtained.

Sample Solution

(See lecture notes clock synchronization no. 2, slide 26 and work with message delay T.)
Proof Sketch Let the (hardware) clock of the i-th node run at speed $1 - \rho + \frac{2i\rho}{n-1}$ and let all messages take time T. Together with the given algorithm this uniquely defines a schedule.
The clock values of the first and last node of this path will eventually be $(n-1)T$: As long as the algorithm does not change the value of its logical clock due to incoming messages the difference between clock values of the nodes steadily keep growing. Let's look at two neighboring nodes. As long as the difference in their time values is less than T they do not react to incoming messages - that is because by the time they receive a message their own clock value has already passed that time (ρ is considered to be small). Extending this argument on the whole path the clock values of the first and the last node can actually obtain the difference $(n-1)T$.