UNIVERSITY OF MARYLAND, BALTIMORE COUNTY

CMSC 621 Advanced Operating Systems

MAKE UP MIDTERM

Sushmitha Manjunatha

10/2/2017

QUESTION 1

PART 1

Lemma : Let S' be a permutation of the events in S. Then the following two statements are equivalent:

1. S' is a causal shuffle of S.

2. S' is the schedule of an execution fragment of a message-passing system with S|p = S'|p for all S'.

Proof:

a) A implies B

- First let us show the similarity. Consider some value p, then every event at p in S also occurs in S', and they must occur in the order same as the first case of the definition of the happens-before relation. This is leading to the to showing S' is the schedule of some execution fragment, as it is stated that any events initiated by p are consistent with p's programming.
- Also observe here all other events are receive events
- By the second case of the definition of happens-before relationship: For each of the receive event e' in S, there must be some matching send event e also in S; thus e and e' are both in S' and occur in the right order.

b) B implies A

- One thing that can be observed here is : S' is a permutation of S

  This because, since every event e in S' occurs at some process p, if S'|p = S|p for all p, then there is a one-to-one correspondence between events in S' and S.

  Now second thing that need to be showed is that : S' is consistent with $\Rightarrow_S$.

  Let consider e $\Rightarrow_S$ e'.

  1. e is a send event and e' is the corresponding receive event. Then e $<_{S'}$ e' because S' is the schedule of an execution fragment.
  2. e and e' are events of the same process p and e $<_S$ e'. But then e $<_{S'}$ e' because S|p = S'|p.
  3. e $\Rightarrow_S$ e' by transitivity. Then each step in the chain connecting e to e' uses one of the previous cases, and e $<_{S'}$ e' by transitivity of $<_{S'}$.
- Thus, we can prove statement 1 which is ,S' is a causal shuffle of S. is equivalent to statement 2 i.e S' is the schedule of an execution fragment of a message-passing system with S|p = S'|p for all S'.

PART2:

Claim

If we order all events by clock value, we get an execution of the underlying protocol that is locally indistinguishable from the original execution.

PROOF:

Let e $<_L$ e' if e has a lower clock value than e'.

Consider if e and e' are two events of the same process: then e $<_L$ e'.

Consider If e and e' are send and receive events of the same message: then again e $<_L$ e'.

By applying lemma 1 i.e

Let S' be a permutation of the events in S. Then the following two statements are equivalent:

1. S' is a causal shuffle of S.

2. S' is the schedule of an execution fragment of a message-passing system with S|p = S'|p for all S'.

Thus, for *any* events e, e', if e $\Rightarrow_S$ e', then e $<_L$ e'.

PART 3:

Claim

Fix a schedule S; then for any e, e', VC(e) < VC(e') if and only if e $\Rightarrow_S$ e'.

PROOF:

We know the update rules of Vector clock which is, When a process executes a local event or a send event, it increments only its own component x_p of the vector. When it receives a message, it increments x_p and sets each x_q to the max of its previous value and the value of x_q piggybacked on the message.

The if part follows immediately from the update rules for the vector clock

- For the only if part, suppose e does not happen-before e'. Then e and e' are events of distinct processes p and p'. For VC(e) < VC(e') to hold, we must have $VC(e)_p < VC(e')_p$; but this can occur only if the value of $VC(e)_p$ is propagated to p' by some sequence of messages starting at p and ending at p' at or before e' occurs.

In this case we have e $\Rightarrow_S$ e'.