Solution of Problem Set 5

November 27, 2005

Question 1(13.11):

The modified algorithm is still correct. Note that process i which executes an $update(w)_i$ operation first performs an *embedded-snap*, and then performs a single *write* to $x(i)$, and then output ack_i. Assume process j which executes a *snap* operation has seen three different tags in $x(i)$, denoted as T_1, T_2, T_3, and S_1, S_2, S_3 are associated *embedded-snap* respectively. S_3 must start after *snap* begins, because tag T_1, T_2 are seen. At the same time, S_3 must end before *snap* finishes, because T_3 is seen and T_3 can be seen if and only if S_3 is finished and process i performs *write* operation. Therefore, S_3 is totally contained in *snap*, and then it can be used as *snap*'s return value. It is possible that the *ack* of the third *update* occurs after *snap* finished, but it doesn't affect the algorithm's correctness.

To see this, assume α is an execution of the above algorithm. Let α' be the same execution as α except moving all the late *ack*s within the corresponding *snap*s while after the preceeding *write* operations. This is still a possible fair execution, because *ack*s are processes' outputs and don't affect the execution. Also, an atomic assignment of serialization points for α' is also an atomic assignment for α since every operation interval in α' is a subinterval of its corresponding interval in α. Using the similar technique of **Theorem 13.13**(Lynch p426), we can show that α' is atomic. Then we can use the same serialization to show that α is atomic.

S_1, S_2 cannot be used as *snap*'s return value because they may happen before *snap* starts.

Question 2(13.18):

It is impossible to solve the agreement problem with 1-failure termination using snapshot atomic objects. The atomic snapshot objects can be implemented with single/writer multi/reader shared registers. Fisher-Lynch-Patterson prove that it is impossible to solve consensus with failures using single/writer multi/reader shared registers. Therefore, we cannot solve consensus with any failure using snapshot objects.

Question 3(Fetch-and-permute objects and consensus)

The lower bound is at least $k-1$. The following is the algorithm to solve consensus among $k-1$ processes with stopping failures. The FAP object is initialized to the permutation $123\cdots k$. Let π_i be the permutation swapping the ith element with kth element, and let r_i be a single/write multi/reader shared register for which process i is its only writer. Then for process i:

1. Write i's initial value to r_i;
2. $S = FAP(\pi_i)$;
3. If $S = 123\cdots k$, decide on i's initial value;
4. Otherwise, decide on r_j where j is the position of the element k in S.

The algorithm is correct because only the process i which first performs the permutation to the FAP object can read $S = 123\cdots k$ and decide on its own value; other successive processes can know which process makes the first permutation by examing the element k's position because this position will never be changed.

The upper bound I know is $k!$ for there are only $k!$ distinguishable states of the FAP object.