CSc72010

Homework (Due Thursday March 3)

Answers

3.1 Fill in more of the details for the inductive proof of the correctness of the LCR algorithm.

In order to prove the correctness of the LCR algorithm, we must prove two Lemmas. The first one claims that some process has to become a leader, and the second one that only one process can become a leader. Using the textbook notations, i_{max} is the process with the maximum UID, which is equal to $u_{max.}$ Also, each process has a unique UID u, which never changes during the process of leader election.

Lemma 3.2: Process i_{max} outputs leader by the end of round n.

To prove this Lemma, it is enough to prove the following assertion:

Assertion 3.3.1: After n rounds, $status_{i_{max}}$=leader.

This assertion can be proved by induction on the number of rounds. The preliminary invariant about smaller numbers of rounds needs to be used.

Assertion 3.2.2: For $0 \le r \le n - 1$, after r rounds, $send_{i_{max}+r} = u_{max}$.

Base r=0: Each process at r=0 initializes its $send_i$ to u_i. So, $send_{i_{max}}=u_{max}$.

Induction Hypothesis: Suppose that this is true for r. This means that the process at distance r from the process i_{max} has $send_{i_{max}+r} = u_{max.}$

Induction Step: By the induction hypothesis we know that process i_{max}+r has received the u_{max} and placed it at $send_{i_{max}+r}$. At round r+1, process i_{max}+r+1 will receive a message from i_{max}+r containing $u_{max.}$ Based on the fact that u_{max} is the unique maximum UID in the ring, $u_{max} > u_{r+1.}$ So, the process i_{max}+r+1 will place u_{max} in $send_{i_{max}+r+1}$, thus proving assertion 3.2.2

We then use assertion 3.2.2 and the special case, where r=n-1. Since assertion 3.2.2 holds also for round r+1, process i_{max} at round n receives its own UID. So, it declares itself a leader.

Lemma 3.3: No process other than i_{max} ever outputs the value leader.

To prove Lemma 3.3 it's enough to show that all processes other than i_{max} always have their status = unknown. Again, it helps to state a stronger invariant.

Assertion 3.3.1: No process other than i_{max} can receive its own UID in a message.

This assertion can be proved by induction on the number of rounds. In order to do this, we need the preliminary invariant that says something about smaller numbers of rounds. So, it's enough to show that at any round r no message can pass through the i_{max}

Assertion 3.3.2: For every r and any i, j, the following holds: After r rounds, if $i \neq i_{max}$ and $j \in [i_{max}, i)$, then $send_j \neq u_i$

Base r=0: Each process at r=0 initializes its $send_i$ to u_i. So, $\forall_{i,j}\ send_j \neq u_i$ for all $i \neq j$.

Induction Hypothesis: Suppose that this is true for r. This means that no process j between $[i_{max}, i)$ can have $send_j = u_i$.

Induction Step: From the induction hypothesis, we know that at the end of round r no process j between $[i_{max}, i)$ can have $send_j = u_i$. At round r+1, a process j in $[i_{max}, i)$ can only have $send_j = u_i$ when a new UID comes in this segment ($[i_{max}, i)$) of the ring. This can only happen when a message carrying u_i passes from i_{max}. However, this is impossible to happen since u_{max} is the maximum UID in the ring.

We then use assertion 3.3.2 and the special case, where r=n. So, at round n no process $j \neq i_{max}$ has $send_j = u_j$. This means that the only message survived after n rounds is the message carrying u_{max}. So, only i_{max} will receive back its own UID, something that proves 3.3.1

Theorem 3.4: LCR solves the leader election problem.

Lemmas 3.2 and 3.3 together imply theorem 3.4.

4.1 Fill in more of the details for the inductive proof of the correctness of the FloodMax algorithm.

Theorem 4.1: In the FloodMax algorithm, process i_{max} outputs leader and each other process outputs non-leader, within diam rounds.

To prove theorem 4.1, it is enough to prove the following assertion:

Assertion 4.1.1: After diam rounds, $status_{i_{max}}$ = leader and $status_j$ = non-leader for every $j \neq i_{max}$.

The key to prove assertion 4.1.1 is the fact that after r rounds, the maximum UID has reached every process that is within distance r of i_{max}, as measured along directed paths in G. This condition is captured by the invariant:

Assertion 4.1.2: For $0 \leq r \leq$ diam and for every j after r rounds, if the distance from i_{max} to j is at most r, then max-$uid_j = u_{max}$. Two additional auxiliary invariants are useful, in order to prove assertion 4.1.2. The first one says that all processes run synchronously. So the state $rounds_i$ is the same at all processes. The second invariant says that no process can ever receive a UID > u_{max}.

Assertion 4.1.3: For every r and j, after r rounds, $rounds_j = r$.
This can be proved using induction on the number of rounds.
Base r=0: It is correct, since at the beginning each process i initializes its $rounds_i$ to 0.
Induction Hypothesis: Suppose that this is true for $r - 1$.
Induction Step: By the induction hypothesis we know that at round $r - 1$ every process j, has $rounds_j = r - 1$. Since FloodMax is running on a synchronous network, at the next round all processes will execute the command "rounds := rounds + 1". So, at round r every process j will have $rounds_j = r$, something that proves assertion 4.1.3.

Assertion 4.1.4: For every r and j, after r rounds, max-$uid_j \leq u_{max}$.
This is true since $u_{max} > u_i$ for all $i \neq i_{max}$. So, at any round r no process can ever receive an UID > i_{max}, which implies assertion 4.1.4

Having proved that all processes run simultaneously and that no process can ever receive a UID > u_{max}, we can now prove assertion 4.1.2 using induction.

Base r=0: i_{max} has max_$uid_{i_{max}} = u_{max}$, since each process j initializes max_$uid_j = u_j$.
Induction Hypothesis: Suppose that this is true for r < diam - 1. This means that all processes j at distance r from the process i_{max} have max-uid_j= u_{max}.
Induction Step: By the induction hypothesis we know that if process j is distance r from i_{max}, then j has set max-uid_j to u_{max} by the end of round r. Suppose process k is distance r+1 from i_{max}. Then there is a process j_0 that is a neighbor of k and that is distance r from i_{max}. During

round r+1, k will receive a message from j_o containing u_{max}. Since u_{max} is larger than max-uid_k by assertion 4.1.4, process k will set max-uid_k to u_{max} at round r + 1.

We then use assertion 4.1.2 and the special case for r=diam. Since assertion 4.1.2 holds also for round diam, it means that each process at distance diam from i_{max} will have set its max-uid to u_{max}. It follows from the algorithm that each process outputs either leader or non-leader at the end of round diam.

Mac-address-tables:
From hulk to lantern (0012.80a8.2480):
	22 out of hulk into port 1 on giant, 11 out of giant into port 21 on lantern
From lantern to goblin(000e.8494.c980):
	21 out of lantern to 11 on giant, 12 out of giant into 23 on goblin
From goblin to giant (000e.83f0.9600):
	23 out of goblin to 12 on giant
From giant to hulk (0012.daa1.0700):
	1 out of giant to 22 on hulk