The University of Texas
at Austin

CS371D Distributed System

Problem Set 03

Edited by LaTeX

Department of Computer Science

STUDENT

Jimmy Lin
xl5224

INSTRUCTOR

Lorenzo Alvisi

TASSISTANT

Chao Xie

RELEASE DATE

April. 22 2014

DUE DATE

May. 02 2014

TIME SPENT

20 hours

May 3, 2014

Contents

1 Problem 1: Leader Election Protocol

1.1 Pseudo-code Representation

First of all, we thought of all algorithms that requires the comparison between the identity of processes and found that the time complexity was at least $O(nlog_2(n))$. And we have to give up this approach.

We realize that this question actually involves in the distributed version of trade-off between time complexity and space complexity. Since linear space complexity is quite tricky, we start to think of protocols with high time complexity and finally it pays off.

We develop a multi-round solution for leader election in synchronous uni-directional ring network. This is because the communication is synchronous. The designed protocol for leader election is as follows.

Here, we refer the terminology "next" and "prev" to the process that it connects to and that are connecting to it, respectively.

Input: n Processes with unique id

Input: a ring-shape network with uni-directional and synchronous communication

Result: a system with new leader confirmed by all n processes

For each replica,

- All processes are initially sleeping.
- At every round, each process i determines whether it is going to wake up by itself.
  - Process i is not allowed to wake up if it receives a leader request from prev
  - If waken up, send a leader request with its id i to next.
  - If not wake up, do nothing.
- In the following round, when each process i receives leader request from process j
  - If $j < i$, reject the leader request with id j since the spread of leader request with id i is slower than that of process id j. And process j propose its leader request to its next.
  - If $j > i$, retain leader request with id j for 2^j rounds. Note that we do this exponential retainment to distinguish leader request in different message(signal) frequency.
  - If $j = i$, the leader request returns back to the proposer and hence process i becomes leader of this distributed system.

1.2 Correctness Proof

In the case that all processes wakes up in the same round, (actually this is the unresolvable scenario for most protocols) all processes send its leader request to their respective "next" process. And according to the well-designed mechanism, the spread of leader request for different processes lies in different frequency period. Specifically, the leader request with lower id value will traverse through the ring network faster than those requests with higher id value. That is, in this case, the process with lowest id will have its leader request message going through the ring network in a fastest rate. And hence it defeats all other competing processes, and finally become the only new leader.

Here we justify the fact that the messages used in the above case is linear on the total nubmer of processes. For leader request with lowest of id i traversing the whole ring network, the number of delivery made to the leader request with id j is $\frac{n}{2^{j-i}}$. By summing over $j-i$ from 1 to ∞, it can be derived that the total number of messages employed are no more than

$$\sum_{j-i=1}^{+\infty} \frac{n}{2^{j-i}} \leq \frac{1-(\frac{1}{2})^{+\infty}}{1-\frac{1}{2}} \cdot n = 2n$$

For other cases, it is obvious to see that the messages required is less than the scenario presented above and the lowest awakened process will finally become the unique leader.

2 Problem 2: Solvability of S failure detector

1 $V_p := (\perp, \ldots, \perp, \text{vp}, \perp, \ldots, \perp)$ {p's estimate of the proposed values}
2 $\triangle_p := (\perp, \ldots, \perp, \text{vp}, \perp, \ldots, \perp)$
3 {phase 1} {asynchronous rounds r_p, $1 \leq r_p \leq n1$}
4 **for** *r_p := 1 to n-1* **do**
5 send $(r_p, \triangle_p, p)$ to all
6 wait until [$\forall q$: received $(r_p, \triangle_q, q)$ or $q \in D_p$]
7 Op := V_p
8 $V_p := V_p \oplus (\oplus q$ received $\triangle_q)$
9 $\triangle_p := V_p \star O_p$ value is only echoed rst time it is seen
10 {phase 2}
11 send (r_p, V_p, p) to all
12 wait until [$\forall q$: received (r_p, V_q, q) or $q \in D_p$]
13 $V_p := \otimes q$ received V_q {computes the "intersection", including V_p}
14 {phase 3}
15 decide on leftmost non-$\perp$ coordinate of V_p

2.1 Proof for Validity

Uniform validity: If a process decides a value v, then some process proposed v.

In the phase three, process p make its decision based on the first non-$\perp$ element of the V_p, which summarizes all cumulative responses from other processes over $n-1$ rounds (in phase 1) and those responses (in phase 2). Hence, the decision v made by process p must have been proposed by some process. And it can be concluded that the solution provided by S failure detector satisfies the validity property.

2.2 Proof for Termination

Termination: Every correct process eventually decides some value.

For those communication in phase 1 and phase 2, if the process q is faulty, it will be detected by the failure detector. Otherwise, the response from process q can be guaranteed (eventually). Hence, there will be no blocking on the "wait" command. Therefore, it can be fully guaranteed that once the process starts executing this protocol, it will eventually reach the terminal state.

2.3 Proof for Agreement

Agreement: No two correct processes decide differently.

In phase 2, every process sends messages to ask for response to its decided value and proceed to phase 3 with the value resulted by intersecting its own decision in phase 1 and others' response in phase 2. In this way, the derived V_p would have the same value on its left-most non-$\perp$ position. Hence, no two correct would decide differently.

3 Problem 3: Ben-Or Lite

After removing the line 12, the algorithm is of the following type.

1 $a :=$ input bit; $r := 1$
2 repeat forever
3 {phase 1}
4 send (a_p,r) to all
5 Let A be the multiset of the first $n - f$ a-values with timestamp r received
6 if $(\exists v \in \{0, 1\} : \forall a \in A)$ then $b_p := v$
7 else $b_p = \perp$
8 {phase 2}
9 send (b_p,r) to all
10 Let B be the multiset of the first $n - f$ b-values with timestamp r received
11 if $(\exists v \in \{0, 1\} : \forall b \in B)$ then decide(v)
12 .
13 else $a_p :=$ \$ {\$ is chosen uniformly at random to be 0 or 1}
14 $r := r + 1$

Algorithm 1: Ben-Or Lite Algorithm

It can be eaisly observed that processes could fail to learn from the values decided by other processes without the line of 12. In this way, the agreement property will be violated. Hence, we can say that the consensus cannot be satisfied by Ben-Or Lite Algorithm.

4 Problem 4: Disprove consensus solution

Disproof: The Validity property will be violated. The decision value may probably be different from the proposed value.

The problem lies in the fact that if two processes propose b and b' sequentially. The final decided version could be overlapped version by b and b', that is, $b_0, b'_1, b'_2, b_3, ..., b_n$ and $b'_0, b_1, b_2, b'_3, ..., b'_n$.