Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Martin Steffen, Immo Grabe

Verteilte Algorithmen

Wintersemester 2003/04 — **Serie 1** — 22. Oktober 2003

Thema : Leader Election (Aufgaben mit Lösungshinweisen)

Ausgabetermin: 22. Oktober 2003

Abgabe: 27. Oktober 2003

Die Vorlesung wird durch (i.d.R) wöchentliche Übungen begleitet, die in 2er-Gruppen bearbeitet werden. Zwei etwas aufwendigere Übungen, die auch stärker gewichtet werden, müssen alleine gelöst werden. Die Termine für den Mitt- und den Endsemestertest werden rechtzeitig bekanntgegeben.

Ausgabe der Zettel ist i.d.R. Montags in der Vorlesung, *Abgabe* eine Woche drauf, Montag vor der Vorlesung. Konkret: Wir *leeren den Schrein immer nach der Vorlesung,* das ist sozusagen die *Deadline!* Besprechung und Rückgabe dann in der folgenden Übung.

Aufgabe 1 (Komplexität des LCR (4 Punkte)) Bearbeiten Sie Übung 3.2 aus [**?**].

Solution:

Rating: 1/1/2.

We are given a ring of size n and let the processes be indexed by i with positions from 0 to $n-1$. The LCR works in an *directional* manner, for instance clockwise. Assume without loss of generality that the UIDs[1] are drawn from the totally ordered domain $0, \ldots, n-1$.

(a) For an $\Omega(n^2)$ message complexity, assign the UID in decreasing order clockwise. Wlog., process i gets thus identity $n-i$.

(b)

An assi

To get an assignment This UID's

The hard thing is: the mean complexitity.

Instead of calculating with conditional probabilities, we try a different approach and ask: given a id at the beginning of the algorithm, what is the probability that this id survives k rounds. The answer is that it survives if it *is the maximum* of the following k ids. The probability for that is $1/k$. This gives the sum

$$\sum_{i=0}^{n} \frac{n}{i} \ .$$

[1] UID's are not the processes themselves.

For large n, this gives asymptotically $n \log n$, as expected.

One interesting *faulty* argument goes as follows:

> Independant of the concrete choice of process ids in the ring, there are fixed n rounds. Now we calculate the mean number of messages as follows: In the first round, there are n messages, each process sends out its id in the first round. In the second round, there will be $n/2$ messages as the probability that a message is "survives" the neighbor is 50%. Now we continue, and again in the next round, again the survivors are cut in half. This looks as a clean series:
>
> $$\sum_{i=0}^{n} \frac{n}{2^i} \ .$$
>
> The upper bound for this series is $2n$.

Uh, that's incorrect. It's a linear complexity but it should be $n \log n$.

So, where's the error? Well, the argument for the first and the second round is impeccable, the trouble starts in the *third* one. The claim, that the survivors of the first round have a 50% chance to survive their neighbors is wrong! The reason is that the survivors have the tendency towards larger ids, and therefore the chance to survive in the third round the neighbor — which might be a survivor[2] or not— is larger that 50%. In an extreme case of this argument, the probability of a token to survive the round n is much larger than 50%! The reason is that there a much *more* cases in which there are one id left (in which case exactly one dies), than that there are 2 ids left. The gist is, that the probability to survive the next round *depends* in the number of rounds survived so far.

Aufgabe 2 (Modifikation des HS-Algorithmusses (3 Punkte)) Bearbeiten Sie Übung 3.8(a).

1. Modifizieren Sie den HS-Algorithmus so daß er unidirektional arbeitet.
2. Argumentieren Sie, daß die $O(n \log n)$ Schranke des HS nicht mehr gilt
3. Geben Sie eine *obere Schranke* für die Komplexität an.

Solution:

For the code, see Figure **??**

[2]By this we mean, that its *token* is still on the journey.

```
Process(1)    // HS-unidicectional

  // messages

   m : UID * {in,out} * int

  // state

  u  = i : UID
  send+ = (i, out,1)  :  M⊥;
  send- = ⊥     :  M⊥;

  status = unknown    : {unknown, leader};
  phase  = 0          : int;

 // message generation

 send the current value of send+ to i+1          // unidirectional ring topology


 // transition function

  send+ := ⊥;     // flush the output vars

  if message from (i-1) =  (v,out,h) then         // out-reception from lower neighbor
    case v > u and h > 1:  send+ := (v,out,h-1)   //  relay it further
         v > u and h = 1:  send- := (v,in,1)      //  reflect back
         v = u          :  status :=  leader     // getting own id -> leader

  if    message from i-1 is (v,in,1) and v ≠ u // in reception ->
   then  send+ := (v,in,1)                              // relay

   if message from i+1 is (u,in,1)               // coming back unsuccessful
   then                                          // start another phase
          phase := phase + 1;
          send+ := (u, out, 2^phase);
```

Figure 1: Unidirectional HS

Basically, the algorithm works like HS, only that one half of the code is cut away, i.e., it still works in phases and in each phase, the processes sends out the exploration message at a distance which doubles reach round before it returns back. The difference to the original HS is, the messages are sent only into to the right.

At first sight, it seems that nothing changes, i.e., that the same complexity argument can be applied. In the second part of the exercise we have to argue that $O(n \log n)$ does no longer hold.

The crucial difference in the behavior is the following. Let's call a *survivor* of round r a process which receives its own exploration message back, i.e., a process which enters the next phase by counting up towards the end of the code.

In the original HS, a survivor is a process with the maximum id in the *middle* of a circle of radius approx. 2^l. In the stupidly modified HS, a process survives if it is the *max of all processes to the right to it* in a distance 2^l. Both versions of the algorithm terminate after the same number of *phases:* the process with the maximum id recognises this in the phase, where for the first time

$$2^{\mathsf{phase}} \geq n \ .$$

This means, the number of phases, for both algorithms, is $O(\log n)$.[3]

Let us make an approximation of the overall number of messages sent during the algorithm. The estimation is based on three part: the number of phases, the number of active processes ("survivors") at the beginning of each phase, and the number of messages of one survivor in a given phase. Let n be the size of the ring, then the number p of phases is bounded by:

$$p \leq \lceil \log n \rceil \tag{1}$$

For the survivors $s(p)$ at the beginning of the phase p, we distinguish between phase 0 and later phases. Initially, all n processes take part. For later phases, consider that (at least) the process with the maximum id has killed, in the preceding phase, all process within its reach. This leads to the following inequation:

$$s(p) \leq \begin{cases} n - 2^{p-1} & p \geq 1 \\ n & p = 0 \end{cases} \tag{2}$$

Finally, each still surviving processes can send out a token at most to a distance of 2^p, from which it will travel back:

$$m(p) \leq 2 * 2^p = 2^{p+1} \tag{3}$$

Hence we can argue:

$$\begin{array}{lll} m \leq & 2n + \sum_{i=1}^{p} s(i)m(i) & \leq \\ & \sum_{i=1}^{p} (n - 2^{i-1})2^{i+1} & = \\ & \sum_{i=1}^{p} n2^{i+1} - 2^{2i} & = \\ & \sum_{i=1}^{p} n2^{i+1} - \sum_{i=1}^{p} 2^{2i} & \end{array}$$

The first of the sums is $\leq 4n^2$.

There had been a little bit of discussion what the *straightforward* adapation of the algorithm is. Some argued that this amounts to LCR, but that's unplausible. The only plausible solution is to just cut-away one half of HS, and leave the rest unchanged. This means, the algorithm works also with hop-counts and phases where the distance increases *exponentially.*

Anyway. One *faulty* argument "shows" that this straightforward generalization is, as HS, also $n \log n$ as communication complexitity. The argument goes as follow:

> What changes is, when the token travels only in one direction, that the number of "Auscheiders" swept away by the token is only (*) 50% in comparison. So in phase l, instead of at most
>
> $$\lfloor \frac{n}{2^{l-1} + 1} \rfloor$$
>
> survivors there are double as many. They send only half as many messages, hence the overall calculation yields the same result.

The error is the statement (*).

Rating: 1 for finding, 2 for analysis. For a worst-case analysis, an argument is needed that it's really the worst case. (0.5 points minus).

[3] Initually, the message buffers are filled with signal of hop-counter 1, which means that we are in phase 0; this also corresponds to the initial value of `phase`. The hopcounter is first checked for equivality with 1 (not zero) and then, in the next phase, sent with a decremented value. This means, each signal travels exactly $2^{\texttt{phase}}$ edges. This means, the leader election already happens in the border case where $2^{\texttt{phase}} = n$