Distributed Systems

ALI KAMANDI, PH.D.

SCHOOL OF ENGINEERING SCIENCE

COLLEGE OF ENGINEERING

UNIVERSITY OF TEHRAN

KAMANDI@UT.AC.IR

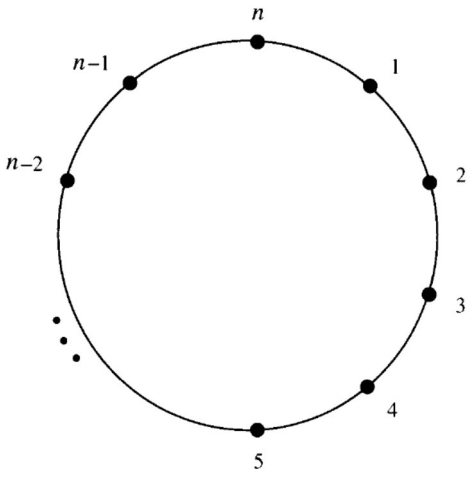


Figure 3.1: A ring of processes.

شرایط مساله

تمامی نودهای غیر لیدر در نهایت به این تصمیم می رسند که لیدر نیستند و تنها یک نود به عنوان لیدر معرفی می شود.

حلقه می تواند یک طرفه (unidirectional) یا دو طرفه(bidirectional) باشد.

تعداد نودها ممکن است از قبل مشخص باشد یا نه.

هر نود یک شناسه منحصر به فرد دارد UID .

Unique Identifier

Theorem 3.1 *Let A be a system of n processes, $n > 1$, arranged in a bidirectional ring. If all the processes in A are identical, then A does not solve the leader-election problem.*

LCR Algorithm

Le Lann, Chang and Roberts

هر نود، شناسه خود را در حلقه ارسال می کند. وقتی یک نود شناسه ای را دریافت نمود، آن را با شناسه خودش مقایسه می کند.

- اگر شناسه دریافتی بزرگ تر باشد، همان شناسه را ارسال می کند.
- اگر کوچک تر باشد، از آن صرفنظر می کند.
- اگر مساوی شناسه خودش باشد، خودش را به عنوان لیدر معرفی می کند.

لیدر، نود با بزرگ ترین شناسه خواهد بود.

```
1  initially do
2      leader ← 0
3      maxId ← id_i
4      send id_i to clockwise neighbor
5  upon receiving j do
6      if j = id_i then
7          leader ← 1
8      if j > maxId then
9          maxId ← j
10         send j to clockwise neighbor
```

Algorithm 6.1: LCR leader election

تحلیل صحت عملکرد

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

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

Assertion 3.3.2 *For $0 \leq r \leq n-1$, after r rounds, $send_{i_{max}+r} = u_{max}$.*

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

Assertion 3.3.3 *For any r and any i, j, the following holds. After r rounds, if $i \neq i_{max}$ and $j \in [i_{max}, i)$ then $send_j \neq u_i$.*

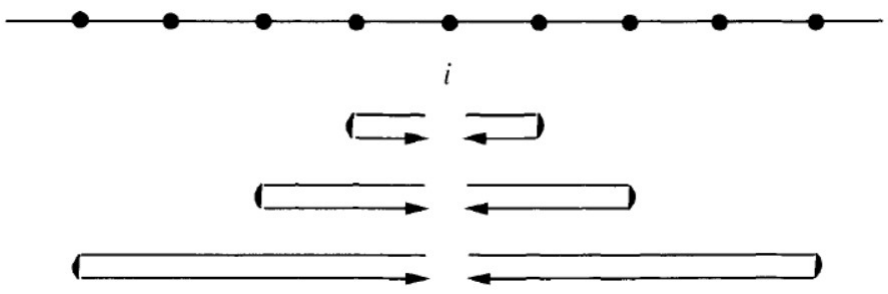
تحلیل پیچیدگی

پیچیدگی زمانی:O(n)

پیچیدگی پیامی: $O(n^2)$

برای اینکه تکلیف همه نودها مشخص شود و همه نودها مطلع شوند که کدام نود به عنوان لیدر انتخاب شده است، لازم است طی n دور و با ارسال n پیام به همه اطلاع رسانی شود. این شیوه برای همه الگوریتم های انتخاب لیدر قابل اجرا است.

HS Algorithm (Hirschberg & Sinclair)



Hirschberg-Sinclair algorithm

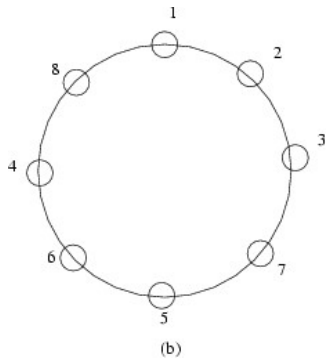
Initially:
- All processes are leaders

Round 0:
- 6 , 7 and 8 are leaders

Round 1:
- 7, 8 are leaders

Round 2:
- 8 is the only leader

At most log(N) rounds



(b)

***HS* algorithm (informal):**

Each process i operates in phases $0, 1, 2, \ldots$. In each phase l, process i sends out "tokens" containing its UID u_i in both directions. These are intended to travel distance 2^l, then return to their origin i (see Figure 3.2). If both tokens make it back safely, process i continues with the following phase. However, the tokens might not make it back safely. While a u_i token is proceeding in the outbound direction, each other process j on u_i's path compares u_i with its own UID u_j. If $u_i < u_j$, then j simply discards the token, whereas if $u_i > u_j$, then j relays u_i. If $u_i = u_j$, then it means that process j has received its own UID before the token has turned around, so process j elects itself as the leader.

All processes always relay all tokens in the inbound direction.

تحلیل پیچیدگی

پیچیدگی زمانی:O(n)

پیچیدگی پیامی: O(n log(n))

منابع

Nancy Lynch, Chapter 3

James Aspnes, Chapter 6

تمرينات

3.3. Modify the *LCR* algorithm so that it also allows all the non-leader processes to output *non-leader*, and so that all the processes eventually halt. Present the modified algorithm using the same style of "code" that we used for the *LCR* algorithm.

3.4. Show that the *LCR* algorithm still works correctly in the version of the synchronous model allowing variable start times. (You might have to modify the code slightly.)

3.8. Consider modifying the *HS* algorithm so that the processes only send tokens in one direction rather than both.

(a) Show that the most straightforward modification to the algorithm in the text does not yield $O(n \log n)$ communication complexity. What is an upper bound for the communication complexity?

(b) Add a little more cleverness to the algorithm in order to restore the $O(n \log n)$ complexity bound.