Distributed Algorithms

Lecture 5

Lecture by Dr. Gadi Taubenfeld
Typeset by Steven Karas

2017-05-03
Last edited 20:46:22 2017-05-03

Disclaimer These lecture notes are based on the lecture for the course Distributed Algorithms, taught by Dr. Gadi Taubenfeld at IDC Herzliyah in the spring semester of 2017. Sections may be based on the lecture slides and accompanying book written by Dr. Gadi Taubenfeld.

Agenda

- Lower bound
- Random algorithm
- Synchronous model

1 Leader Election

1.1 Lower bound

First proven by J.E. Burns in 1980. We will prove that the message complexity of electing a leader in a ring is $\Omega(n \log n)$.

Model

- Asynchronous network
- Bidirectional ring (implies unidirectional)
- # of processes is not known (does not imply known #)
- FIFO links
- processes startup spontaneously or upon receiving a message

Weaker proof The message complexity of electing a leader in a ring is $\frac{1}{4} n \log n$ where n is a power of 2 ($n = 2^i$). We'll prove this in the future.

Proof The message complexity of electing a leader in a ring is $\frac{1}{8} n \log \frac{n}{2}$ for any n.

There exists some $i \in \mathbb{N}$ such that $\frac{n}{2} \leq 2^i \leq 2^n$.

From the above, we know that the lower bound for this smaller ring is as above. It follows that an lower bound can be found for a generic $\frac{n}{2}$.

$$\frac{1}{4} \cdot \frac{1}{2} \cdot n \cdot \log \frac{n}{2} \leq \frac{1}{4} \cdot 2^i \cdot \log 2^i$$

Notation: Ring

Notation: Line A leader election that works on a ring that is executed on a line may either elect exactly one leader, or not successfully elect a leader. An asynchronous ring can simulate the behavior of a line by delaying the delivery of messages across both directions of a link.

Lemma 1 There is an infinite number of lines (and rings) of size 1 on which at least one message is sent (when P is executed).

Lemma 2 For every $i \geq 0$, there is an infinite set of lines L_i such that for every line $L \in L_i$, it holds that:

1. $|L| = 2^i$
2. $\text{message}(L) \geq 1 + \frac{1}{4} \cdot 2^i \log 2^i$

Proof by induction, where the base case is $i = 0$, which holds by Lemma 1. Induction step proven using lemma 3.

Lemma 3 From any 4 lines in L_{i-1}, it is possible to compose (by connecting 2 lines) one line L such that:

1. $|L| = 2^i$
2. $\text{message}(L) \geq 1 + \frac{1}{4} 2^i \log 2^i$

Consider the lines $A, B, C, D \in L_{i-1}$. From these, compose AB, BA, CD, DC as candidates. Execute P on each of the line segments. Before we release the messages between the segments, at least $2(1 + \frac{1}{4}2^{i-1} \log 2^{i-1})$ messages have been sent. Assume the negative, such that $2(1 + \frac{1}{4}2^{i-1} \log 2^{i-1}) + \frac{1}{4}2^i > 1 + \frac{1}{4}2^i \log 2^i$.

However, this means that the information after the messages between the segments are released has propagated at most across one half of each segment. But, this would imply that if we were to connect all four segments into a larger ring, it would be possible for them to elect two leaders.[1]

1.2 A Randomized Algorithm

Presented by A. Itai and M. Rodeh in 1997. We present a randomized algorithm with identical processes for electing a leader in a ring, where on the average $O(n \log n)$ bits are transmitted.

Model

- Identical processes
- Asynchronous network
- Unidirectional ring
- # of processes is known
- FIFO links
- No failures (processes, links, messages)
- Start spontaneously or on message receipt

Algorithm Each active process chooses 0 or 1 with equal probability, and sends it to the next active process. An active process becomes inactive if it chose 0 and the preceding process chose 1. Each process checks if it is the only active process, and becomes the leader if it is.

Analysis The probability that an active process becomes inactive is $\frac{1}{4}$. The expected number of processes that become inactive each round is $\frac{1}{4}n$. The expected number of rounds until one process remains is $\log_{4/3} n \approx 2.4 \log_2 n$. The probability that more than one process remains active after $5 \log n$ is very small.

1.3 Leader election in a synchronous ring

The worst case message complexity of electing a leader in a synchronous ring is $O(n \log n)$ for comparison based algorithms and $O(n)$ for non-comparison based algorithms. Lower bounds proven by G.N. Fredericksson in 1987. However, the time complexity for these is unbounded, so they are not practical.

Model

- Unique identifiers
- Synchronous network
- Unidirectional ring
- # of processes is known
- No need to assume FIFO links
- All processes start at the same time

[1]The drawings in the slides do much better justice to the elegance of this proof, and I strongly recommend going over them as well.

1.4 Time Slice algorithm

Synchronous model as above. Chunk message rounds of size n. If the chunk number equals the process id, then self-elect as the leader. Takes $n \cdot \min id$ time.

Variant: Maximum id election Elect the minimum node, then have it poll for the max id, then elect it as the leader.

1.5 Variable speeds algorithm

Synchronous model as above, but the number of processes is unknown. Each process initiates a token which travels along the ring. A token v travels at the rate of 1 message every 2^v rounds. Each process tracks the smallest token it has seen and discards any larger ones. If a token returns to its originator, it is elected.

Analysis The number of messages is less than $2n$. This follows from $\sum_{i=\min id}^{\max id} \frac{n}{i-\min id} \leq 2n$. The time complexity is $n \cdot 2^{\min id}$.

1.6 Modified variable speeds algorithm

Same model as above, but with variable start times.

Starter processes are ones that start spontaneously. Non-starters are ones that started by receiving a message.

Algorithm Starters create a token and pass it on. Non-starters act as relays. Perform the regular variable speed algorithm.

Message complexity of $4n$. Time complexity of $n + n \cdot 2^{\min starter.id}$

2 Next time

Minimum Spanning Tree and general network leader election.