Leader Election Proofs

November 16, 2017

The proofs provided herein are taken from the book "Distributed Computing", by Hagit Attiya and Jennifer Welch, with some terminology/notation adaptation.

1 Anonymous rings

Bear in mind that we consider oriented rings, i.e. all processes have consistent notions of left and right.

Theorem 1 There is no anonymous algorithm for leader election in synchronous rings of size 2 or more.

Proof: The proof stems from the fact that without process IDs (and as the ring is oriented) symmetry cannot be broken. Formally, assume there is an anonymous leader election algorithm A for ring R of size 2 or more. We prove by induction that, at the beginning of each round, the states of all processes are the same and the input buffers of all processes contain the same messages.

The base case $k = 0$ is straightforward: in the initial configuration, as processes do not have distinct IDs, all processes start in the same state and there are no
messages in input buffers. Assume the claim holds for all rounds up to \( k - 1 \) and prove for \( k \). From induction hypothesis, all processes are at the same state at the beginning of round \( k - 1 \) and have the same messages in their input buffers. Thus, in round \( k - 1 \), all processes start from the same state and receive the same messages, hence they send the same message \( m_r \) (if any) to the right link, send the same message \( m_l \) (if any) to the left link and shift to the same state. It follows that in the beginning of the \( k \)'s round all processes are again in the same state and have exactly the same messages in their input buffers. It follows that if one process shifts to an elected state all processes do. This is a contradiction.

\[ \tag*{\Box} \]

2 Algorithms for Asynchronous Rings

**Theorem 2** The message complexity of the simple leader election algorithm is \( \Theta(n^2) \).

**Proof:** Clearly the algorithm does not send more than \( n(n + 1) \) messages. As for the other direction, consider a ring in which the processes have IDs \( 0, 1, \ldots, n - 1 \) and are arranged in the ring in a decreasing order (going clockwise). Then the **probe** message of any process \( i \) is sent \( i + 1 \) times. Thus the total number of messages sent in this case is: \( \sum_{i=0}^{n-1} (i + 1) + n = \Theta(n^2) \).

The more efficient algorithm proceeds in **phases**. A \( k \)-**neighborhood** of a process in the ring is the set of processes that are at a distance of no more than \( k \) from it (in both directions). In phase \( l \), each process tries to become the **temporary leader** of its \( 2^l \)-neighborhood. Only processes that succeed in becoming temporary leaders in phase \( l \) continue to the \((l + 1)\)'s phase.
Theorem 3 The message complexity of the second leader election algorithm is $O(n \log n)$.

Proof: The probe distance in phase $k$ is at most $2^k$. Hence the number of messages triggered by a temporary leader in phase $k$ is at most $4 \cdot 2^k$. We now prove by induction that the number of processes that are temporary leaders at the end of phase $k$, for $k \geq 0$, is at most $\frac{n}{2^k+1}$.

If process $p_i$ is a temporary leader at the end of phase $k$, then every participating process in its $k$-neighborhood has a smaller identifier. The closest together that two phase $k$ temporary leaders can be is if the node at the right side of the $k$-neighborhood of one is immediately to the left of the other. Hence the distance between them is at least $2^k$. It follows that the maximum number of phase $k$ temporary leaders is $\frac{n}{2^k+1}$.

Consequently, by the end of phase $\log n - 1$ there is at most a single leader. Hence the total number of messages is at most:

$$4n + \sum_{k=1}^{\log n-1} 4 \cdot 2^k \cdot \frac{n}{2^k-1+1} + n < 8n \log n.$$ 

Note that the first summand above accounts for the number of phase-0 messages and the last summand accounts for a termination message which must be sent from the leader (and is not shown in the code).

\[\blacksquare\]

3 An $n \log n$ Lower Bound

Definition 3.1 An execution $\sigma$ of a ring algorithm $A$ is open if there exists an edge $e$ of the ring such that in $\sigma$ no message is received over $e$ in either direction. We
call e an open edge of $\sigma$.

In the following proofs, we assume that $A$ is a uniform algorithm and that $n$ is an integral power of 2. We will prove the following theorem.

**Theorem 4** For every $n$ and every set of $n$ (unique) identifiers, there is a ring using those identifiers that has an open execution of $A$ in which at least $M(n)$ messages are received, where $M(2) = 1$ and $M(n) = 2M(n/2) + \frac{1}{2}(n^2 - 1)$ for $n > 2$.

For simplicity, we assume in the proofs that follow that 1) the process with the maximum ID is selected as leader, and 2) all processes must learn the ID of the leader. These assumptions can be easily lifted.

The following lemma proves the base case of the theorem.

**Lemma 5** For every set consisting of 2 identifiers, there is a ring $R$ using those two identifiers that has an open execution of $A$ in which at least one message is received.

**Proof:** Assume $R$ contains processes $p_0, p_1$ and that $ID(p_0) > ID(p_1)$. Let $\alpha$ be an execution of $A$ on the ring. Then $p_1$ must eventually write $p_0$’s ID. Thus at least one message from $p_0$ must be received by $p_1$. Let $\sigma$ be the shortest prefix of $\alpha$ that includes the first step in which a message is received. Then the other edge is an open edge of $\alpha$. 

The following lemma proves the inductive step of the theorem.

**Lemma 6** Choose $n > 2$. Assume that for every set of (unique) $n/2$ identifiers, there is a ring using those identifiers that has an open execution of $A$ in which at least $M(\frac{n}{2})$ messages are received. Then for every set of (unique) $n$ identifiers,
there is a ring using those identifiers that has an open execution of A in which at least $2M\left(\frac{n}{2}\right) + \frac{1}{2}\left(\frac{n}{2} - 1\right)$ messages are received.

**Proof:** Let $S$ be a set of $n$ identifiers. We partition $S$ into two sets $S_1$ and $S_2$ of size $\frac{n}{2}$ each. From assumption, there exists a ring $R_1$, with processes using the identifiers of $S_1$, that has an open execution $\sigma_1$ of $A$ in which at least $M\left(\frac{n}{2}\right)$ messages are received. Similarly, there exists a ring $R_2$, with processes using the identifiers of $S_2$, that has an open execution $\sigma_2$ of $A$ in which the same number of messages is received.

Let $e_1$ and $e_2$ be open edges of $R_1$ and $R_2$, respectively. Denote the processes adjacent to $e_1$ by $p_1$ and $q_1$, and the processes adjacent to $e_2$ by $p_2$ and $q_2$. We paste $R_1$ and $R_2$ by deleting edges $e_1$, $e_2$ and connecting $p_1$ to $p_2$ with $e_p$ and $q_1$ to $q_2$ with $e_q$. Let the resulting ring be $R$ (see illustration 1. in the powerpoint presentation).

We note that $\sigma_1\sigma_2$ is an execution of $R$. This is because processes that participate in $\sigma_1$ (respectively $\sigma_2$) have no way to distinguish between their execution in $R_1$ (respectively $R_2$) and in $R$. Clearly, the number of messages received in $\sigma_1\sigma_2$ is at least $2M\left(\frac{n}{2}\right)$.

We now show that by unblocking exactly one of the edges $e_p$, $e_q$ we can force $A$ to receive $\frac{1}{2}\left(\frac{n}{2} - 1\right)$ additional messages.

Consider all executions $\sigma_1\sigma_2\sigma_3$ in which both $e_p$ and $e_q$ remain open. If there is such an execution $\sigma_3$ in which at least $\frac{1}{2}\left(\frac{n}{2} - 1\right)$ messages are received then we are done. Assume otherwise. Then there is an execution $\sigma_1\sigma_2\sigma_3$ that results in a quiescent configuration, in which no process will send more messages unless it receives a new message, and no messages are in transit except on $e_p$, $e_q$.

Assume WLOG that the process with the maximum identifier in $R$ belongs to
$R_1$. Then, eventually, all the processes of $R_2$ must receive this process’ identifier. As no messages were received in $\sigma_1 \sigma_2 \sigma_3$ over $e_p$ or $e_q$, none of the processes in $R_2$ have terminated. It follows that in any legal continuation of $\sigma_1 \sigma_2 \sigma_3$, in which the algorithm terminates any process in $R_2$ must receive at least one additional message.

Let $\sigma''_4$ be a legal continuation of $\sigma_1 \sigma_2 \sigma_3$ (occurring after edges $e_p$ and $e_q$ are unblocked, that is, re-connected) during which all messages are received and all processes terminate. Then at least $\frac{n}{2}$ messages are received in $\sigma''_4$. Let $\sigma'_4$ be the shortest prefix of $\sigma''_4$ in which $\frac{n}{2} - 1$ messages are received. Consider the processes in $R$ that received these messages. These are two consecutive sets of processes $P$ and $Q$ such that $P$ (resp. $Q$) contains the processes that received messages due to the unblocking of $e_p$ (resp. $e_q$). (See Illustration 2. in the powerpoint presentation). Since at most $\frac{n}{2} - 1$ messages are received in $\sigma'_4$, $P$ and $Q$ are disjoint and the number of messages received in one of them is at least $\frac{1}{2}(\frac{n}{2} - 1)$. WLOG assume this set is $P$. Let $\sigma_4$ be the subsequence of $\sigma'_4$ that contains only the steps involving processes in $P$. Then $\sigma_1 \sigma_2 \sigma_3 \sigma_4$ is a legal execution in which at least $2M(\frac{n}{2}) + \frac{1}{2}(\frac{n}{2} - 1)$ are received and this execution occurs also when we keep edge $e_q$ disconnected (open), therefore it is an open execution.

\section{The Uniform Synchronous Algorithm is Correct and has Linear Complexity}

\textbf{Lemma 7} The algorithm is correct.

\textbf{Proof:} Follows easily from the fact that only the process with minimum ID recei-
ves its message back.

For bounding the total number of messages, we partition the messages sent by the algorithm to the following 3 categories.

1. First-phase messages.
2. Second-phase messages sent before the message of the eventual leader enters its second phase.
3. Second-phase messages sent after the message of the eventual leader enters its second phase.

**Lemma 8** The total number of messages in the first category is at most $n$.

**Proof:** We prove that at most a single first-phase message is sent over each link. Assume otherwise, then there is some process $p_i$ that receives two first-phase messages, say, $<p_j, 1>$ and $<p_k, 1>$. This implies that both $p_j$ and $p_k$ participate in the algorithm, i.e., they are not relays. Assume also WLOG that $p_j$ is (clockwise) closer to $p_i$ than $p_k$ is. Message $<p_k, 1>$ must pass through $p_j$ en-route $p_i$. Since $p_j$ is not a relay, it does not forward any first-phase messages; if it receives such a message, it turns it into a second-phase message. This is a contradiction. □

**Lemma 9** The total number of messages in the second category is at most $n$.

**Proof:** For simplicity we assume that all the processes that start the algorithm spontaneously do so together. Let $p_i$ be the eventual leader. Clearly, the message of $p_i$ enters its second phase at round $n$ at the latest. It follows that messages from the second category are sent in at most $n$ rounds.
Message $< j, 2 >$ travels at speed $1/2^j$. Thus at most $n/2^j$ messages in this category are sent. It follows that the maximum number of second-category messages is $O(\sum_{j=1}^{n-1} n/2^j) < n.$

Lemma 10 \textit{The total number of messages in the third category is at most $2n$.}

\textbf{Proof:} Let $p_i$ be the eventual leader. Observe that no message is sent after $< i, \ast >$ is received back by $p_i$. Let $p_j$ be another process. Clearly, $i < j$. Since message $< i, \ast >$ travels at speed at least $1/2^i$, at most $n \cdot 2^i$ rounds are required before $< i, \ast >$ returns to $p_i$. Hence messages in the third category are sent only during $n \cdot 2^i$ rounds. It follows that the total number of messages in this category that are sent is at most $\sum_{j=0}^{j=n-1} n/2^j < 2n.$