Fast Leader Election in Anonymous Rings with Bounded Expected Delay

Rena Bakhshi[a,*], Jörg Endrullis[a], Wan Fokkink[a], Jun Pang[b]

[a] *Vrije Universiteit Amsterdam, Department of Computer Science, The Netherlands*
[b] *Université du Luxembourg, Computer Science and Communications, Luxembourg*

Abstract

We propose a probabilistic network model, called asynchronous bounded expected delay (ABE), which requires a known bound on the expected message delay. In ABE networks all asynchronous executions are possible, but executions with extremely long delays are less probable. Thus, the ABE model captures asynchrony that occurs in sensor networks and ad-hoc networks.

At the example of an election algorithm, we show that the minimal assumptions of ABE networks are sufficient for the development of efficient algorithms. For anonymous, unidirectional ABE rings of known size n we devise a probabilistic election algorithm having average message and time complexity $O(n)$.

Key words: distributed computing, design of algorithms, analysis of algorithms

---

1. Introduction

The two commonly used network models are synchronous and asynchronous. In synchronous network all nodes proceed simultaneously in global rounds. While this model allows for efficient algorithms, the assumptions are typically too strict for practical applications. The (fully) asynchronous network model, on the other hand, requires only that every message will eventually be delivered. The assumptions of this model are generally too weak to study the time complexity of algorithms.

For the development of practically usable, efficient algorithms we need to find a golden mean between synchronous and asynchronous networks. A possible approach is asynchronous bounded delay (ABD) networks [7, 21], where a fixed bound on the message delivery time is assumed. ABD networks are generally closer to synchronous than to fully asynchronous networks. The ABD model is a nice theoretical framework, but the assumption of a bounded message delay is often hard to satisfy in real-life networks.

We propose a probabilistic model, that we call *asynchronous bounded expected delay (ABE)* networks. The ABE network model requires a known bound on the *expected*

*Corresponding author

Email addresses: rbakhshi@few.vu.nl (Rena Bakhshi), joerg@few.vu.nl (Jörg Endrullis), wanf@cs.vu.nl (Wan Fokkink), jun.pang@uni.lu (Jun Pang)

Preprint submitted to Information Processing Letters *June 8, 2011*

message delay. Thus, we strengthen the asynchronous network model with a minimal requirement for analysing the time complexity of algorithms.

We elaborate on the advantage of ABE over the ABD network model. A strong point in favour of the ABE network model is its probabilistic nature. A probabilistic treatment of the message delay is for example crucial to analyse protocols that employ a *timeout mechanism* like in the TCP/IP protocol. The assumption of the known bound on the expected message delay allows for deriving a lower bound on the probability that a message will arrive within a given time limit (before the timeout). In contrast, it is impossible to evaluate algorithms with timeout mechanism on the basis of the ABD network model. The only assumption of the ABD network model is a fixed bound on the message delay. If the timeout is greater or equal to this bound, then all messages will arrive in time, rendering the timeout mechanism useless. If the timeout is smaller than the bound on the message delay, it is impossible to estimate how many messages will arrive before the timeout (all messages could arrive after the timeout).

Moreover, messages sent via a physical channel may get lost or corrupted, for example, due to material imperfections or signal inferences. Since message transmission is unreliable, all we can settle for is a probability p of successful transmission. To ensure that a message arrives at its destination, it may need to be retransmitted (possibly multiple times) until the transmission has been successful. The number of necessary retransmissions for a message cannot be bounded: with probability $(1-p)^k$ a message requires more than k retransmissions, and thus the message delay is unbounded. While the message delay cannot be bounded, from the probability p we can derive the average number of needed retransmissions and thereby the average message delay. In particular, the average number of transmissions is $k_{avg} = \sum_{k=0}^{\infty}(k+1)\cdot(1-p)^k \cdot p = \frac{1}{p}$. Assuming that a successful transmission takes one time unit, the average message delay is $\frac{1}{p}$ as well. Assuming that we know the exact value of p for physical channels may already be an unrealistic assumption. However, frequently a lower bound $p_{low} \leq p$ on p can be derived from material properties in combination with the maximum strength of inference signals in a given environment. Such a lower bound on p is sufficient to derive an upper bound on the expected message delay, and this suffices for ABE networks.

Although the assumptions of ABE networks are minimal, it is possible to devise efficient algorithms. We demonstrate this on an example of an election algorithm for anonymous, unidirectional ABE rings having (average) linear time as well as message complexity. So its efficiency is comparable to the most optimal election algorithms known for anonymous, synchronous rings [16].

Election is the problem of determining a unique leader in a network, in the sense that the leader (process or node) knows that it has been elected and the other processes know that they have not been elected. This is a fundamental problem in distributed computing and has many applications. For example, it is an important tool for breaking symmetry in a distributed system. By choosing a process as the leader it is possible to execute centralised protocols in a decentralised environment. Election can also be used to recover from token loss for token-based protocols, by making the leader responsible for generating a new token when the current one is lost. There exists a broad range of election algorithms; see e.g. the summary in [21, 17]. These algorithms have different message complexity in the worst and/or average case. Furthermore, they vary in communication mechanism (*asynchronous vs. synchronous*), process names (*unique identities vs. anonymous*), and network topology (e.g. *ring, tree, complete graph*).

Classical (deterministic) election algorithms are [14, 11, 20, 8] for asynchronous rings with worst-case message complexity $O(n \log n)$, and [12] for synchronous unidirectional rings with worst-case message complexity $O(n)$. Without additional assumptions, $\Omega(n \log n)$ is the lower bound on the average message complexity for asynchronous rings [19, 6].

In an *anonymous network*, processes do not carry an identity. As the number of processes in a network increases, it may become difficult to keep the identities of all processes distinct, or a network may accidentally assign the same identity to different processes. In some situations, transmitting identities may be too expensive (e.g., FireWire bus, cf. [18]). Since deterministic election is impossible in an anonymous network [1]; randomisation is used to break the symmetry.

If the network size is known, it is possible to construct a randomised election algorithm that terminates with probability one, e.g. [15]. It exhibits infinite traces, but the probability that such an infinite trace is executed is zero. For unknown network size, the presence of an *oracle* for leader detection is required, e.g. [9]. In the absence of an oracle, there are several impossibility results for anonymous rings. No randomised algorithms can elect a leader in an anonymous ring if the ring size is known only within a factor of two [1]. Furthermore, algorithms for computing the ring size always have a positive probability of computing the wrong result [21]. Thus, there is no randomised algorithm that can elect a leader in an anonymous ring of unknown size.

We study the problem of election in anonymous, asynchronous rings. For such rings, the best known election algorithms are [15, 10, 13, 5] with average message complexity $\Theta(n \log n)$. Itai and Rodeh [16] have proposed an algorithm for synchronous unidirectional anonymous rings; its average message complexity is $O(n)$. The algorithm strongly depends on the notion of rounds, even though the idea of the activation parameter in this algorithm is similar to ours. The algorithm proceeds multiple elections, each of which consist of exactly n rounds. Both central ideas of the algorithm crucially depend on synchronous networks: the synchronous sending of the messages in the beginning of the election as well as determining the number of active nodes by counting the messages passed through in n rounds. Thus, the similarities between our algorithm and that of Itai-Rodeh are limited to the activation parameter, which is a very natural choice in probabilistic systems. Hence, the Itai-Rodeh algorithm cannot be adapted for ABE networks; however, our algorithm works in synchronous networks.

Contribution and outline. The ABD model assumes that there is a fixed bound on message delay. In Sec. 2 we propose a probabilistic network model, called asynchronous bounded expected delay (ABE) networks, that allows for an unbounded message delay, and assumes a known bound on expected message delay. This model is closer to the fully asynchronous network model than ABD networks.

For anonymous, unidirectional ABE rings of known ring size n, we devise a probabilistic election algorithm with average message complexity $O(n)$; see Sec. 3. Previously, election algorithms with linear time and message complexity have only been known for network models with a strict bound on the message delay, i.e., synchronous networks and ABD networks. In Sec. 4 and 5, we prove the correctness, the average linear complexity of our algorithm, and optimize the activation parameter.

2. Asynchronous Bounded Expected Delay Networks

We describe the model of ABE networks, briefly announced in [4], which strengthens asynchronous networks with the assumption of a known bound on the expected message delay. This strengthening allows one to analyse the (average) time complexity of algorithms.

At first glance it may appear superfluous to consider a bound on the expected delay, instead of the expected delay itself. We briefly motivate our choice. First, when determining the expected delay for real-world networks, one needs to take into account parameters such as material properties, environmental radiation, electromagnetic waves, etc. Frequently, these values change over time, or cannot be calculated precisely. Thus we have to cope with ranges for each of these parameters, and consequently, the best we can deduce is an upper bound on the expected message delay. Second, the links in a network are typically not homogeneous and often have different expected delays. Then the maximum of these delays can be chosen as an upper bound, instead of having to deal with different delays for every link.

Definition 1. We call a network *asynchronous bounded expected delay (ABE)* if:

1. A bound δ on the expected message delay (network latency) is known.
2. Let t be a real time. We assume that bounds $0 < s_{low} \leq s_{high}$ on the speed of the local clocks are known. That is, for every node A the following holds for the local clock $\mathcal{C}_A$: $s_{low} \cdot (t_2 - t_1) \leq |\mathcal{C}_A(t_2) - \mathcal{C}_A(t_1)| \leq s_{high} \cdot (t_2 - t_1)$.
3. A bound γ on the expected time to process a local event is known.

In comparison with ABD networks, the ABE network model relieves the assumption of a strict bound on the message delay. The assumption is weakened to a bound on the expected message delay. Thereby we obtain a probabilistic network model which, as discussed above, covers a wide range of real-world networks to which the ABD network model is not applicable. For this reason, we advocate the model of ABE networks as a natural and useful extension of the fully asynchronous network model.

Example 1. *The known upper bound on the expected message delay δ allows for deriving a lower bound $p(t)$ on the probability that a message will be delivered with a delay less or equal to t. From $\delta \geq t \cdot (1-p(t))$ it follows that $p(t) \geq 1-\delta/t$ for $t > \delta$. If $\delta = 1$, then $p(2) = 0.5$, $p(5) = 0.8$, etc. As a consequence we also obtain that long message delays are less probable (but possible).*

To conclude this section, we discuss synchronisers for ABE networks. A synchroniser is an algorithm to simulate a synchronous network on another network model. A well-known impossibility result [2] states that fully asynchronous networks cannot be synchronised with fewer than n messages per round (every node needs to send a message every round). This of course destroys the message complexity when running synchronous algorithms in a fully asynchronous network. The more efficient ABD synchroniser by Tel et al. [22] relies on knowledge of the bounded message delay. As in fully asynchronous networks the message delay in ABE networks is unbounded (although we assume a bound on the expected delay). In a slogan: every execution of a fully asynchronous network is also an execution of an ABE network. The difference is that huge message delays in ABE networks are less probable. Hence, the impossibility result [2] for fully asynchronous networks carries over to ABE networks as follows:

Corollary 1. *ABE networks of size n cannot be synchronised with fewer than n messages per round.*

Hence, we cannot run synchronous algorithms in ABE networks without losing the message complexity. Although ABE networks are very close to fully asynchronous networks, it turns out the model allows for the development of efficient algorithms. We show this at the example of a surprisingly robust and efficient leader election algorithm with $\Theta(n)$ average time and message complexity.

3. Fast Leader Election with Bounded Expected Delay

We present an election algorithm for anonymous, unidirectional ABE rings. The algorithm is parameterised by a *base activation parameter* $\mathcal{A}_0 \in (0, 1)$. The order of messages is arbitrary between any pair of nodes. For simplicity, we assume that the expected time to process a local event is 0, that is, $\gamma = 0$. However, all results presented in this paper can be generalised straightforwardly for expected value $\gamma > 0$.

The algorithm presented below actually does not require continuous clocks. It suffices that every node has a local timer *ticking* once per (local) time unit. Obviously such a discrete timer can be simulated using continuous clocks, thus, w.l.o.g. we assume that every node has a timer in the sequel.

During execution of the algorithm every node is in one of the following states: idle, active, $\mathsf{passive}$ or leader; in the initial configuration all nodes are idle. Moreover, every node A stores a number $d(A)$, initially 1. The messages sent between the nodes are of the form $\langle \mathsf{hop} \rangle$, where $\mathsf{hop} \in \{1, \dots, n\}$ is the hop-counter of the message. Every node A executes the following algorithm:

- If A is idle, then at every clock tick, with probability $1 - (1 - \mathcal{A}_0)^{d(A)}$, A becomes active, and in this case sends the message $\langle 1 \rangle$.
- If A receives a message $\langle \mathsf{hop} \rangle$, it sets $d(A) = \max(d(A), \mathsf{hop})$. In addition, depending on its current state, the following actions are taken:
  - (i) If A is idle, then it becomes $\mathsf{passive}$ and sends the message $\langle d(A) + 1 \rangle$.
  - (ii) If A is $\mathsf{passive}$, then it sends the message $\langle d(A) + 1 \rangle$.
  - (iii) If A is active, then it becomes leader if $\mathsf{hop} = n$, and otherwise it becomes idle, purging the message in both cases.

In other words, messages travel along the ring and 'knock out' all idle nodes on their way. That is, idle and $\mathsf{passive}$ nodes *forward* messages; by forwarding a message, idle nodes are turned $\mathsf{passive}$. If a message has knocked out an idle node (at any point during its lifetime), we refer to the message as *knockout message*. If a message hits an active node, then it is purged, and the active node becomes idle, or is elected leader if $\mathsf{hop} = n$ (that is, if the node itself is originator of the message). We say that a node has *woken up* when it transits from the idle to the active state.

The value $d(A)$ stores the highest received hop-count for every node. It indicates that $d(A) - 1$ predecessors are $\mathsf{passive}$. A higher value of $d(A)$ increases the probability that a node A becomes active. By taking $1-(1-\mathcal{A}_0)^{d(A)}$ as wake-up probability for nodes A, we achieve that the overall wake-up probability for all nodes stays constant over time. This ensures that the algorithm has linear time and message complexity.

Note that we forward messages $\langle \mathsf{hop} \rangle$ as $\langle d(A)+1 \rangle$ instead of $\langle \mathsf{hop}+1 \rangle$. This is used since the channels exhibit non-FIFO behaviour. Consider the following scenario. A message $\langle h \rangle$ with high hop-count overtakes a message $\langle \ell \rangle$ with low hop-count, and then $\langle h \rangle$ is purged by an active node A. Then $d(A)=h$ and when $\langle \ell \rangle$ passes by A its hop-count will be increased to $h+1$ (as if the overtaking would never have taken place). Using $\langle \mathsf{hop}+1 \rangle$ instead of $\langle d(A)+1 \rangle$, there exist scenarios where all nodes are passive except for one idle node B, and $d(B)=2$. That is, $d(B)$ is much lower than the actual number of passive predecessors of B, and as a consequence the overall wake-up probability would not stay constant over time.

We briefly elaborate on why the framework of ABE networks is essential for this election algorithm. The bound on the expected delay is necessary for proving that the algorithm terminates with probability one, and that the average time and message complexity are $\Theta(n)$. To the best of our knowledge, the algorithm is the first leader algorithm with the linear average time and message complexity in the settings of asynchronous anonymous rings without a fixed bound on the message delay.

4. Correctness

Our election algorithm has terminated when all nodes are either passive or leader, and no messages are in transit. Our algorithm satisfies the following invariants:

Lemma 2. *For every node A at least $d(A)-1$ predecessors are* passive*.*

Proof. Initially, the claim holds since $d(A)=1$. Assume the claim would be wrong, then consider the first event invalidating the claim. By definition of the algorithm, $d(A)$ is the maximum hop-count of all messages received by A, and passive nodes stay passive forever. Therefore, we can restrict attention to the case that a node A receives a message $\langle x \rangle$, but fewer than $x-1$ predecessors of A are passive. The message $\langle x \rangle$ must have been sent by the predecessor B of A. The case of B being non-passive is trivial, since then it must have sent the message $\langle 1 \rangle$. If B is passive, then $x \le d(B)+1$, and since the invariant holds for B, $d(B)-1$ predecessors of B are passive. Then $d(B)=x-1$ predecessors of A are passive. □

Lemma 3. *When a* leader *node is elected, all other nodes are* passive*.*

Proof. According to the algorithm, an active node A is elected leader, when it receives the message $\langle n \rangle$. Then $d(A)=n$, so by Lem. 2, all $n-1$ other nodes are passive. □

Lemma 4. *There are always as many messages in the ring as* active *nodes.*

Proof. Initially all nodes are idle and the lemma holds. Let us consider all possible scenarios. If an idle or passive node receives a message $\langle \mathsf{hop} \rangle$, it will relay the message further. Thus, the number of messages and active nodes remains unchanged. If an active node receives a message, it changes its state to either idle or leader. In both cases, the message is purged. Thus, both messages and active nodes decrease by 1. If an idle node becomes active, it sends out a message. Thus, the number of messages and the number of active nodes both increase by 1.Finally, note that when an active node receives the message $\langle n \rangle$, and becomes leader, by Lem. 3, all other nodes are passive, so that there are no other messages in the ring. Hence, in all cases the invariant is preserved. □

Lemma 5. *Always at least one node is not* passive*.*

Proof. Only idle nodes A can become passive, after receiving a message $\langle \mathsf{hop} \rangle$. This message will be passed on as $\langle d(A)+1 \rangle$. Hence, there is at least one message travelling in the network, and, by Lem. 4, at least one active node in the network. □

Using these invariants, we can show that our algorithm is correct.

Theorem 6. *Upon termination, exactly one leader has been elected.*

Proof. Termination without elected leader is not possible, since by Lem. 5, there is always a non-passive node A (if A would be active there would be a message travelling by Lem. 4). By Lem. 3, if a leader has been elected, all other nodes are passive. Hence, upon termination there is a unique leader. □

Theorem 7. *The election algorithm terminates with probability one.*

Proof. There exist only a finite number of network configurations C. For every non-terminated configuration $c \in C$, there is a probability $P(c) > 0$ such that for every possible non-deterministic choice the probability of the next scenario is at least $P(c)$:

- no idle node becomes active until all messages in the network are forwarded and received by active nodes (by Lem. 4 there are as many messages as active nodes in the network);
- next, exactly one idle node A (which exists by Lem. 5) becomes active, and its message travels around the whole ring without any other node becoming active. When A receives its own message, it is elected leader and we have termination.

Taking $\zeta = \min\{P(c) \mid c \in C\}$ we obtain that from every possible non-terminated configuration the probability of termination is at least $\zeta > 0$. Hence the algorithm terminates with probability one. □

5. Complexity

In this section, we show that our algorithm has linear time and message complexity. First, we give an intuition behind the linear complexity of the algorithm. The crux is the choice of a suitable activation parameter $\mathcal{A}_0(n)$. To achieve a linear complexity it suffices to take (the non-optimal) $\mathcal{A}_0(n) = 1 - \sqrt[n]{(n-1)/(n+1)}$. We briefly elaborate on this choice. For simplicity we assume $s_{low} = s_{high} = 1$ and $\delta = 1$ for this sketch. If all nodes gamble once, the probability that any of the nodes wakes up is $1-(1-\mathcal{A}_0(n))^n = 2/(n+1)$. The bound δ on the expected message delay allows us to derive a bound $R(n)$ on the expected time for a message to travel through the whole ring: $R(n) = \delta \cdot n = n$. Then the probability $W(n)$ of any node waking up during $R(n)$ time is $1-(1-2/(n+1))^n$ which converges to: $W(n) \to 1-e^{-2}$, for $n \to \infty$.

This is the crucial observation yielding linear time and message complexity: the time $R(n)$ for a round trip of a message is linear in n, and the probability $W(n)$ of any node waking up during this time is constant (i.e. asymptotically independent of n).

Omitting the simplification $s_{low} = s_{high} = \delta = 1$, we have $R(n) = \delta \cdot n = n$, and we employ the lower (upper) bound s_{low} (s_{high}) on the clock speed to derive a lower (upper) bound on the probability of any node waking up during the time.

We define the *activation count* $w(A)$ as the number of times node A has woken up. We now show the properties of the activation count.

Lemma 8. *Let A and B be* active *or* idle *nodes such that the path from A to B visits only passive nodes. Then the number of messages between A and B is*

$$w(A) - w(B) + \mathsf{active}(B)$$

where $\mathsf{active}(B)$ *is* 1 *if B is* active*, and* 0 *otherwise.*

Proof. Initially all nodes are idle and the lemma holds. That is, $\mathsf{active}(B) = 0$; moreover, $w(A) = w(B) = 0$ and there are no messages in the ring. For a node A we denote $p(A)$ and $s(A)$ as the first active or idle predecessor and successor of A, respectively. First, consider the case: $A = B$. Then all nodes except for A are passive, and by Lem. 4 there is a message in the ring iff A is active. For the remainder of the proof we assume $A \neq B$, that is, $A \neq s(A)$ and $A \neq p(A)$. We consider all possible events. If a passive node receives a message $\langle \mathsf{hop} \rangle$, it relays the message further. Thus, the number of messages between active or idle nodes remains unchanged.

If an idle node A receives a message $\langle \mathsf{hop} \rangle$, it becomes passive and relays the message further. Then, the number of messages between $p(A)$ and $s(A)$ is:

$$\begin{aligned}(w(p(A)) - w(A) + 0) + (w(A) - w(s(A)) + \mathsf{active}(s(A)))\\ = w(p(A)) - w(s(A)) + \mathsf{active}(s(A))\end{aligned}$$

If an idle node A becomes active, it sends out a message. The same holds for the number of messages between A and $s(A)$. Then $w(A)$, and so $w(A) - w(s(A)) + \mathsf{active}(s(A))$, increases by 1. The number of messages between $p(A)$ and A remains unchanged: $w(p(A)) - w(A) + \mathsf{active}(A)$, both $w(A)$ and $\mathsf{active}(A)$ increase by 1 and thereby equal each other out.

Finally, if an active node A receives a message, it changes its state to either idle or leader. In both cases, the message is purged. The number of messages between A and $s(A)$ remains unchanged: $w(A) - w(s(A)) + \mathsf{active}(s(A))$. The number of messages between $p(A)$ and A decreases by 1. The same holds for $w(p(A)) - w(A) + \mathsf{active}(A)$, since $\mathsf{active}(A)$ decreases by 1. Hence, in all cases the lemma holds. □

Lemma 9. *Let A and B be nodes such that A is not* passive*, the path from A to B visits only passive nodes, and there are no knockout messages between A and B. Then the number of nodes between A and B (not counting B) is $d(B) - 1$. (Note, if $A = B$, we assume the path around the whole ring)*

Proof. Let $\rho(C)$ be the number of passive predecessors of C (not counting C). We say that a node C is *informed* if $d(C) - 1 = \rho(C)$. Likewise, a message M with hop-count h is called *informed* if exactly $h - 1$ nodes preceding M are passive.

We prove by induction over the number of events: for all nodes A and B such that A is not passive, and the path from A to B visits only passive nodes, either B is informed, or there is an informed knockout message between A and B. By Lem. 2 we have $d(B) - 1 \leq \rho(B)$. Initially all nodes are informed. We consider all possible events. If an informed (knockout) message is relayed by a passive node, the message stays informed and the node becomes informed.

If a message knocks out an idle node A, then either the node was informed and hence sends out an informed knockout message, or (by induction hypothesis) there must be an informed knockout message M before A. In the latter case, if M was the message received, then the node becomes informed.

If an active node A purges an informed knockout message, it becomes informed.

Finally, if an idle node A becomes active, it sends out a (non-knockout) message.

Hence, in all cases the lemma holds. □

As a direct consequence we obtain the following corollary stating that the overall wakeup probability of all nodes in the ring stays basically constant.

Corollary 10. *Whenever there are no knockout messages travelling in the ring,*

$$1 - (1 - \mathcal{A}_0)^n = 1 - \prod_{A \textit{ is } \mathsf{idle} \textit{ or } \mathsf{leader}} (1 - \mathcal{A}_0)^{d(A)}$$

Proof. There are two possible scenarios, where no knockout messages travel in the ring; (i) there are only idle and passive nodes, or (ii) node A is leader and the other $n - 1$ nodes are passive. Then we can divide the ring in chains of passive nodes followed by an idle or leader node A, and by Lem. 9 we have $d(A)$ is the length of the chain plus 1. As a consequence we obtain: $\sum_{A \text{ is idle or leader}} d(A) = n$. □

In other words, the overall wakeup probability (for idle and active nodes) may only decrease as long as a knockout message travels through the ring. This means that there is a message in the ring that has turned nodes from idle to passive and did not yet encounter an active node (and hence did not complete a round trip). As soon as this message is purged by an active node A, the node updates its counter $d(A)$ representing the number of passive predecessors, thereby restoring the overall wakeup probability.

Lemma 11. *If an* active *node A with activation count $w(A)$ receives a message while all other nodes have activation count smaller than $w(A)$, then A is elected as* leader.

Proof. Assume, towards a contradiction, that there are at least two idle or passive nodes in the ring. By Lem. 8 there are no messages between A and its first active or idle predecessor, unequal to A. Hence, A can receive a message only if all other nodes are passive. By Lem. 9 it follows that after A received this message, $d(A) - 1 = n - 1$. Hence A must have received the message $\langle n \rangle$, and thus is elected leader. □

Theorem 12. *The election algorithm has linear time and message complexity.*

Proof. First, we prove the linear time complexity. Recall that the timer of every node ticks ones per unit of local time. Thus, within $1/s_{low}$ global time the timer of every node ticks at least once. By Lem. 10, $1 - (1 - \mathcal{A}_0)^n$ is a lower bound on the probability for at least one node waking up in $1/s_{low}$ time. Hence, an upper bound F on the expected time until the first node becomes active can be obtained as follows:

$$F = \frac{1}{s_{low}} \cdot \sum_{i=0}^{\infty}(i+1)\cdot(1-\mathcal{A}_0)^{ni}\cdot(1-(1-\mathcal{A}_0)^n) = \frac{1}{s_{low}} \cdot \frac{1}{1-(1-\mathcal{A}_0)^n}$$

The expected time for a message to travel around the entire ring is $R = n \cdot \delta$. An upper bound W on the probability of any node waking up during R time is $W = 1-(1-\mathcal{A}_0)^{n\cdot R\cdot s_{high}}$.

The first node A gets active after expected time F. Then with probability $1-W$ no other nodes wakes up while the message of A travels around the ring, and in this case, after expected time R the node A will be elected as leader. With probability $\leq W$ another node wakes up while the message of A makes a round trip. Then after expected time R all non-passive nodes in the ring will have activation count at least one. This can be seen as follows. The message of A travels along the ring, knocking out all nodes with activation count $< w(A)$, until it is purged by an active node B. Then B must have become active after A, and itself sent out a message. We continue with tracing this message of B, and successively apply the same reasoning until we have finished one round trip, and are back to A.

In case another node has woken up while the message of A was travelling around the ring, the described scenario repeats. For this it is important to observe that by Lem. 10 the overall wakeup probability of all nodes stays constant. As a consequence, after expected time F a node B will wake up, and get the highest activation count $w(B)$. By Lem. 11, after expected time R either (i) B will be elected leader (probability $1-W$), or (ii) all non-passive nodes in the network have at least activation count $w(B)$ (probability W). This leads to the upper bound on the expected time of termination

$$\sum_{i=0}^{\infty}(i+1)\cdot(F+R)\cdot W^i\cdot(1-W) = \frac{1+R\cdot s_{low}-(1-\mathcal{A}_0)^n\cdot R\cdot s_{low}}{(1-(1-\mathcal{A}_0)^n)\cdot(1-W)\cdot s_{low}} \qquad (1)$$

For $\mathcal{A}_0$ we choose the following expression depending on n: $\mathcal{A}_0 = 1-\sqrt[n]{\frac{n-1}{n+1}}$. Using this activation parameter time complexity is linear with respect to the ring size n. The derivation of this parameter will be explained in the next subsection. Substituting the expression for $\mathcal{A}_0$ into (1), we obtain:

$$\frac{\frac{1}{s_{low}}+(1-(1-\mathcal{A}_0)^n)\cdot n\cdot\delta}{(1-(1-\mathcal{A}_0)^n)\cdot(1-\mathcal{A}_0)^{n^2\cdot\delta\cdot s_{high}}} = \frac{\frac{1}{s_{low}}+\frac{2}{n+1}\cdot n\cdot\delta}{\frac{2}{n+1}\cdot(\frac{n-1}{n+1})^{n\cdot\delta\cdot s_{high}}} = \frac{\frac{1}{s_{low}}\cdot\frac{n+1}{2}+n\cdot\delta}{(\frac{n-1}{n+1})^{n\cdot\delta\cdot s_{high}}}$$

Noting that $\lim_{n\to\infty}(\frac{n-1}{n+1})^{n\cdot\delta\cdot s_{high}} = (\frac{1}{e^2})^{\delta\cdot s_{high}}$, we obtain that the time complexity (i.e. the expression above) is $\Theta(n)$.

Now, we prove the linear message complexity. Since the time complexity is linear, here is $\xi > 0$ such that the average execution time of the algorithm $\leq \xi\cdot n$. A lower bound on the expected time until a node becomes active is: $F_{low} = \frac{1}{s_{high}}\cdot\frac{1}{1-(1-\mathcal{A}_0)^n}$. Thus, the expected number of nodes "waking up" during the algorithm execution is $\leq \frac{\xi\cdot n}{F_{low}} = \xi\cdot n\cdot s_{high}\cdot(1-(1-\mathcal{A}_0)^n) = \xi\cdot s_{high}\cdot\frac{2\cdot n}{n+1}$. Since every "wake up" gives rise to at most n messages (once around the ring) and by the expression above, the algorithm has linear message complexity. □

The concept of message complexity differs from bit complexity. The message complexity refers to the expected number of messages until termination, while *bit complexity* is the total number of bits transmitted. Our algorithms has $\Theta(n)$ message complexity, and $\Theta(n \cdot \log n)$ bit complexity as the messages are $\log n$ in size.

Optimal Value for $\mathcal{A}_0$. The crux of the algorithm is the activation parameter $\mathcal{A}_0$ influencing both the time and message complexity. We optimise $\mathcal{A}_0$ with respect to time complexity in dependence on the network size n. For simplicity we assume $\delta = 1$, $s_{low} = s_{high} = 1$. We conclude this section with a discussion of the general case.

The value for $\mathcal{A}_0$ (depending on n) derived from the optimisation is applied in the proof of Thm. 12. This shows that the obtained activation parameter results in linear time and message complexity for arbitrary values for δ, s_{low} and s_{high}, although the analysis does not take these values into account.

For the purpose of optimisation, we consider average-case scenarios instead of worst-case scenarios as used in the proof of linear time and message complexity (Thm. 12). Let $\alpha = 1 - \mathcal{A}_0$. Then, the average number of attempts before a first node becomes active is $(1-\alpha^n) \cdot 1 + (1-\alpha^n) \cdot \alpha^n \cdot 2 + \ldots = \frac{1}{1-\alpha^n}$. The probability β that a message of this first node completes its round-trip is $\alpha^{n-1} \cdot \alpha^{n-2} \cdot \ldots \cdot \alpha = \alpha^{\frac{n(n-1)}{2}}$ since the expected time for a round-trip is $n \cdot \delta = n$. Note that in the proof of Thm. 12 we used an upper bound on the worst case for β, namely α^{n^2}. Thus, the average time required to elect a leader is

$$\beta \cdot \frac{1}{1-\alpha^n} + \quad (1-\beta) \cdot \beta \cdot \frac{2}{1-\alpha^n} + (1-\beta)^2 \cdot \beta \cdot \frac{3}{1-\alpha^n} + \ldots = \frac{1}{\beta \cdot (1-\alpha^n)}.$$

We take $(1-\beta)$ as the probability of fail trial for the message to make a round-trip. We now derive an optimal value for $\mathcal{A}_0$. Optimal here means that the average time to elect the leader is as low as possible. That is, we minimise $\frac{1}{\beta \cdot (1-\alpha^n)}$ by taking the derivative: $(n-1) \cdot \alpha^{\frac{1}{2}n(n-1)} - (n+1) \cdot \alpha^{\frac{1}{2}n(n+1)} = 0$. Hence, $\mathcal{A}_0 = 1 - \sqrt[n]{\frac{n-1}{n+1}}$. For large ring sizes n, the optimal activation parameter $\mathcal{A}_0$ is approximately $1 - \sqrt[n^2]{\frac{1}{e^2}}$. The optimal value of the activation parameter is supported by simulation experiments, and by model checking [3].

References

[1] D. Angluin. Local and global properties in networks of processors. In *Proc. of STOC*, pages 82–93. ACM, 1980.

[2] B. Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[3] R. Bakhshi, J. Endrullis, W. J. Fokkink, and J. Pang. Asynchronous bounded expected delay networks. Technical Report arXiv:1003.2084, CoRR, 2010.

[4] R. Bakhshi, J. Endrullis, W. J. Fokkink, and J. Pang. Brief announcement: Asynchronous bounded expected delay networks. In *Proc. of PODC*. ACM, 2010.

[5] R. Bakhshi, W. J. Fokkink, J. Pang, and J. C. van de Pol. Leader election in anonymous rings: Franklin goes probabilistic. In *Proc. of TCS*, volume 273 of *IFIP*, pages 57–72. Springer, 2008.

[6] H.L. Bodlaender. New lower bound techniques for distributed leader finding and other problems on rings of processors. *Theor. Comput. Sci.*, 81:237–256, 1991.

[7] C. T. Chou, I. Cidon, I. S. Gopal, and S. Zaks. Synchronizing asynchronous bounded delay networks. *IEEE Trans. Comm.*, 38(2):144–147, 1990.

[8] D. Dolev, M. Klawe, and M. Rodeh. An $O(n \log n)$ unidirectional algorithm for extrema finding in a circle. *J. of Algorithms*, 3(3):245–260, 1982.

[9] M. Fischer and H. Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *Proc. of OPODIS*, volume 4305 of *LNCS*, pages 395–409. Springer, 2006.

[10] W. J. Fokkink and J. Pang. Variations on Itai-Rodeh leader election for anonymous rings and their analysis in PRISM. *J. UCS*, 12(8):981–1006, 2006.

[11] R. Franklin. On an improved algorithm for decentralized extrema finding in circular configurations of processors. *Commun. ACM*, 25(5):336–337, 1982.

[12] G. N. Frederickson and N. A. Lynch. Electing a leader in a synchronous ring. *J. ACM*, 34(1):98–115, 1987.

[13] L. Higham and S. Myers. Self-stabilizing token circulation on anonymous message passing. In *Proc. of OPODIS*, pages 115–128. Hermes, 1998.

[14] D.S. Hirschberg and J.B. Sinclair. Decentralized extrema-finding in circular configurations of processors. *Commun. ACM*, 23(11):627–628, 1980.

[15] A. Itai and M. Rodeh. Symmetry breaking in distributive networks. In *Proc. of FOCS*, pages 150–158. IEEE, 1981.

[16] A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Inform. and Computation*, 88(1):60–87, 1990.

[17] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[18] S. Maharaj and C. Shankland. A survey of formal methods applied to leader election in IEEE 1394. *J. UCS*, 6(11):1145–1163, 2000.

[19] J. K. Pachl, E. Korach, and D. Rotem. Lower bounds for distributed maximum-finding algorithms. *J. ACM*, 31(4):905–918, 1984.

[20] G. Peterson. An $O(n \log n)$ unidirectional algorithm for the circular extrema problem. *ACM Trans. Program. Lang. Syst.*, 4(4):758–762, 1982.

[21] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2000.

[22] G. Tel, E. Korach, and S. Zaks. Synchronizing ABD networks. *IEEE/ACM Trans. Netw.*, 2(1):66–69, 1994.