Brief Announcement: Non-Uniform Content-Oblivious Leader Election on Oriented Asynchronous Rings

Jérémie Chalopin
Aix Marseille Univ, CNRS, LIS, Marseille, France

Yi-Jun Chang
National University of Singapore, Singapore

Lyuting Chen
National University of Singapore, Singapore

Giuseppe A. Di Luna
DIAG, Sapienza University of Rome, Italy

Haoran Zhou
National University of Singapore, Singapore

Abstract

In this paper, we study the leader election problem in oriented ring networks under content-oblivious asynchronous message-passing systems, where an adversary may arbitrarily corrupt message contents.

Frei et al. (DISC 2024) recently presented a uniform terminating leader election algorithm for *oriented* rings in this setting, with message complexity $O(n\mathsf{ID}_{\max})$ on a ring of size n, where $\mathsf{ID}_{\max}$ is the largest identifier in the system.

In this paper, we investigate the message complexity of leader election in this model, showing that no uniform algorithm can solve the problem if each process is limited to sending a constant number of messages in one direction.

Interestingly, this limitation hinges on the *uniformity* assumption. In the *non-uniform* setting – where processes know an upper bound $U \geq n$ on the ring size – we present an algorithm with message complexity $O(nU\mathsf{ID}_{\min})$, in which each process sends $O(U\mathsf{ID}_{\min})$ messages clockwise and only three messages counter-clockwise. Here, $\mathsf{ID}_{\min}$ is the smallest identifier in the system. This dependence on the identifiers compares favorably with the dependence on $\mathsf{ID}_{\max}$ of Frei et al. (DISC 2024).

We also show a non-uniform algorithm where each process sends $O(U \log \mathsf{ID}_{\min})$ messages in one direction and $O(\log \mathsf{ID}_{\min})$ in the other. The factor $\log \mathsf{ID}_{\min}$ is *optimal*, matching the lower bound of Frei et al. (DISC 2024).

Finally, in the anonymous setting, we propose a randomized algorithm where each process sends only $O(\log^2 U)$ messages, with a success probability of $1 - U^{-c}$.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Content-Oblivious Networks, Leader Election, Oriented Rings, Asynchronous Systems

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.51

Related Version *Full Version*: https://arxiv.org/abs/2509.19187

Funding This work has been partially supported by ANR project DUCAT (ANR-20-CE48-0006), by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (24-1323-A0001), and by the Italian MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU through projects SERICS (PE00000014).

1 Introduction

The field of reliable distributed computing is characterized by the study of a wide range of failure models, from failures affecting specific processes [4, 17] to failures affecting communication channels, such as message omission failures, message addition failures or message corruptions [19]. In a recent paper, Censor-Hillel et al. [7] introduced a new interesting failure model for channels: the fully-defective asynchronous network. In this model, processes do not fail, but all messages in transit may be arbitrarily corrupted by an adversary. While the adversary can alter the content of any message, it cannot create or destroy messages.

This model is content-oblivious, meaning that messages carry no reliable content, not even the identity of the sender. The only information a message conveys is its existence.

Surprisingly, Censor-Hillel et al. [7] showed that if a leader process is initially known and the topology is 2-edge-connected, it is possible to simulate a reliable, uncorrupted message-passing network, even in asynchronous settings, within this fully-defective model.

This result makes studying leader election algorithms for this model particularly compelling: if a leader can be elected in a certain topology, with an algorithm that is terminating and where it is possible to distinguish in some way the messages of the leader election algorithm, then on such topology any asynchronous algorithm can be simulated even if all messages are arbitrarily corrupted. In this regard, Censor-Hillel et al. [7] conjectured that electing a leader was impossible in content-oblivious model and thus that having a leader was a necessary requirement. We stress that electing a leader is a fundamental problem in the classical message-passing setting, and it has been extensively studied in systems with unique identifiers [1, 12, 16, 18] and in anonymous networks [2, 3, 8, 21].

Frei et al. [13] disproved the aforementioned conjecture for a notable family of 2-edge-connected networks: oriented ring topologies. In an oriented ring, processes have a common notion of clockwise and counter-clockwise direction. More specifically, Frei et al. [13] proposed a quiescently terminating leader election algorithm with a message complexity of $O(n\mathsf{ID}_{\max})$, where $\mathsf{ID}_{\max}$ is the maximum identifier in the system. The quiescent termination of the algorithm allows it to be composed with the simulator of Censor-Hillel et al. [7] and thus implies that the presence of unique identifiers is enough to simulate any asynchronous algorithm on a content-oblivious oriented ring.

Additionally, Frei et al. [13] proved that, for any uniform algorithm, there exists an identifier assignment under which the algorithm must send $\Omega\left(n \log \frac{\mathsf{ID}_{\max}}{n}\right)$ messages. Although originally established for the uniform setting, this lower bound can be extended to the non-uniform setting, even when the *exact* value of n is known; see the full version of the paper for details.

In anonymous systems, where processes lack identifiers, Frei et al. [13] showed a randomized leader election algorithm using $n^{O(c^2)}$ messages and succeeding with probability $1 - n^{-c}$, but without explicit termination. In fact, Itai and Rodeh [15] proved that no uniform algorithm can both elect a leader and guarantee explicit termination in anonymous systems without additional assumptions. We stress that the lack of explicit termination does not allow the composition of this algorithm with the simulator of Censor-Hillel et al. [7].

Finally, we want to mention the related line of research of the beeping model [5, 6, 9, 20], where in each round a process may either emit a beep to its neighbors or listen. This model is inherently synchronous. Therefore, the techniques developed for leader election in the beeping model [10, 11, 14] cannot be applied in content-oblivious networks.

1.1 Contributions

We shed new light on the message complexity of leader election in the content-oblivious model. Specifically, we show that a uniform algorithm, even on oriented rings, cannot elect a leader while sending only a constant number of messages in a single direction.

Interestingly, we show that non-uniform algorithms, where processes know an upper bound U on the network size n, can circumvent this impossibility result.

In particular, we present an algorithm for oriented rings (Section 3) that elects a leader by sending $O(nU\mathsf{ID}_{\min})$ messages in the clockwise direction and where each process sends just three messages in the counter-clockwise direction. Here ID_{min} is the minimum identifier in the system. We also present a non-uniform algorithm for oriented rings (Section 4) that sends $O(nU \log \mathsf{ID}_{\min})$ messages in one direction and $O(n \log \mathsf{ID}_{\min})$ messages in the other. We stress that the factor $\log \mathsf{ID}_{\min}$ of our algorithm is *optimal*. This follows from the aforementioned lower bound of Frei et al. [13], which shows the necessity of logarithmic dependency on the values of the identifiers. Our result implies that $\Theta(\log \mathsf{ID}_{\min})$ is the *tight* message complexity for constant-size instances.

The key technical novelty in our non-uniform algorithms is the design of a global synchronization mechanism that leverages the known upper bound on the network size to keep processes partially synchronized across algorithmic phases. This mechanism enables processes to compare their identifiers bit by bit, leading to an algorithm whose complexity depends only on the size of the identifiers (and not on their values).

Regarding anonymous networks, with non-uniformity at hand, the aforementioned impossibility result of Itai and Rodeh [15] no longer applies. This allows us to create a randomized algorithm (Section 5) that, with probability at least $1 - U^{-c}$, is quiescently terminating. The algorithm has message complexity $O(n \log^2 U)$. In contrast to the randomized approach of Frei et al. [13], whose message complexity has an inherent exponential dependence on c, our algorithm achieves near-linear message complexity with only a multiplicative dependence on c, which is hidden in the $O(\cdot)$ notation.

For space reasons, in this brief announcement, we just announce our results and describe the high-level ideas; see the full version of this paper for technical details.

2 System Model

We consider a system composed of a set of n processes $P = \{p_0, p_1, \ldots, p_{n-1}\}$ that communicate on a ring network by sending messages to each other. More precisely, a process p has two local communication ports: port 0 and port 1. By means of a specific port, a process is able to send messages to and receive messages from one of its neighbors.

When the system is not anonymous, processes have unique identifiers that are arbitrarily selected from $\mathbb{N}$.

We assume an *asynchronous system*; that is, each message has an unpredictable delay that is finite but unbounded. A message cannot be lost. Messages are buffered, in the sense that if several messages are delivered to a process at the same time, they are stored in a local buffer and can then be retrieved by the destination process at its discretion. The local computation time at each process may be arbitrary but bounded, and processes do not share a notion of common time.

Processes are correct and do not experience any kind of failure.

Content-Oblivious Algorithms. Since we are interested in *content-oblivious algorithms*, we assume that messages do not carry any information apart from their existence. That is, each message is an empty string (a *pulse* [13]) and a process only observes the local port number on which the message is received.

Oriented Rings. We consider *oriented* rings. A ring is oriented if processes share a common notion of *clockwise* (CW) and *counter-clockwise* (CCW) orientation. More precisely, consider a ring $(p_0, p_1, \ldots, p_{n-1})$ for each $j \in [0, n-1]$, at process p_j, port 0 leads to p_{j+1} and port 1 leads to p_{j-1} (where indices are taken modulo n). We will say that a message is traveling in the clockwise direction if it is sent on port 0 and received on port 1 (i.e., a message goes from p_i to p_{i+1}); conversely, a message travels in the counter-clockwise direction if it is sent on port 1 and received on port 0.

Uniform and Non-Uniform Algorithms. An algorithm is *uniform* if it works for all possible network sizes; i.e., processes do not know the total number of processes n or a bound on it. An algorithm is *non-uniform* if processes know an upper bound $U \geq n$.

Message Complexity. We measure the performance of our algorithms by their *message complexity*. That is the number of all messages sent. We further discriminate between the numbers of clockwise messages and counter-clockwise messages sent by each process.

Leader Election Problem. In this problem, processes must elect one of them as the *leader* and all others are designated as *non-leader* processes. We consider *terminating leader election*, in the sense that every process must eventually enter either a `Leader` or `Non-Leader` state, and both of these states are final – meaning no further state changes are possible, and the process terminates the execution of the algorithm.

Moreover, we require that once the last process enters its final state, there are no messages in transit in the network. This is known as *quiescent termination* [13]. In our algorithms, the last process to enter its final state is always the unique leader. This property enables easy composability of algorithms: once the leader enters the final state, it knows that the network is "at rest" and can begin sending messages for a subsequent algorithm [13, Section 1.1].

3 Constant Number of Messages in One Direction

We now describe an algorithm where each process sends $O(U\mathsf{ID}_{\min})$ messages in the clockwise direction, and just 3 messages in the counter-clockwise direction. This algorithm is of interest as it shows that our claim below needs the uniformity assumption:

▶ **Theorem 1.** *For any uniform leader election algorithm and any bound $b \in \mathbb{N}$ there exists an oriented ring where at least one process sends more than b messages in each direction.*

The Constant Direction Algorithm, pseudocode in Algorithm 1, starts in a competing phase. In the competing phase, blue lines, a process p with identifier ID performs a loop of $U\mathsf{ID}$ iterations. In each iteration, the process sends a message in the CW direction and waits for a message from the CW direction. This phase synchronizes the processes so that the indices of any two processes in the phase loop can differ by at most $n-1$. The process with the minimum identifier will be the only one to end the competing phase by completing the prescribed number of iterations.

Algorithm 1 Constant Direction Algorithm(ID,boundOnSize).

```
1  for i ← 1 to boundOnSize ∗ ID do
2      send a message on port 0
3      receive a message on port q      // Wait for a message and store the port number in
         variable q
4      if q = 0 then break
5  if q = 0 then
       /* if a process enters this branch, then it is not the process with minimum ID
          */
6      send a message on port 1
7      receive a message on port 1                    // Wait for a message on port 1.
8      repeat
9          receive a message on port q
10         send a message on port 1 − q
11     until received two messages on port 0
12     return Non-Leader
13 else
       /* only the process with minimum ID enters this branch                    */
14     send a message on port 1
15     receive a message on port 0
16     send a message on port 1
17     repeat
18         receive a message on port q
19         if q = 1 then
20             send a message on port 0
21     until q = 0
22     send a message on port 1
23     receive a message on port 0
24     return Leader
```

This process then enters the termination detection phase (green lines). In this phase, the process first sends a message CCW to inform all other processes that they are not leaders. Once this message returns, it sends a second CCW message while starting to relay CW messages; when this second message will be received back the process will know that all CW messages in the network are eliminated. Finally, once the second message is received, it sends a third and final CCW message to terminate the algorithm.

A process whose identifier is not the minimum will enter the relay phase (red lines) upon receiving and relaying a message from the CCW direction. In the relay phase, the process first waits to receive a CW message, in order to eliminate the CW message it sent during the competing phase, and afterwards it relays both CCW and CW messages it receives. A non-leader process terminates after receiving three CCW messages.

▶ **Theorem 2.** *There exists a quiescently terminating leader election algorithm for oriented rings in which each process sends $O(U\mathsf{ID}_{\min})$ messages clockwise and three messages counterclockwise, for a total of $O(nU\mathsf{ID}_{\min})$ messages.*

4 An algorithm that sends $O(nU \log \mathsf{ID}_{\min})$ messages

We now describe an algorithm in which each process sends $O(U \log \mathsf{ID}_{\min})$ messages. The algorithm uses an encoding of identifiers such that the smallest identifier is not a prefix of any other identifier, and the last bit of every encoded identifier is 0. To achieve this, given an identifier ID of bit length ℓ, we encode it as $1^\ell 0 \cdot \mathsf{ID} \cdot 0$.

The algorithm elects the process with the minimum identifier and proceeds in elimination rounds. In each round, active processes compare a specific bit of their (encoded) identifiers, eliminating all processes that do not have the minimum bit at that position.

The bound U on the network size is used to synchronize processes across different rounds of the algorithm. When an active process enters a round, it begins the *Synchronization phase*. In this phase, each active process sends and receives messages in the CW direction for U steps. Then, processes with bit 0 send and receive one message in the CCW direction, blocking messages in the CW direction (this is the *Zero Signaling phase*). Meanwhile, processes with bit 1 attempt to continue sending and receiving messages in the CW direction for another U steps. However, they become inactive if they receive a CCW message, indicating the presence of a process with bit 0 (this is the *No-Zero Checking phase*).

The rationale for these phases is to ensure that processes with bit 1 proceed to the next round in the active state if and only if no process with bit 0 exists in the current round. Indeed, when processes with bit 0 stop propagating CW messages, the processes with bit 1 are prevented from completing their *No-Zero Checking phase*.

Conversely, a process with bit 0 in a given round remains active. The *Synchronization phase* described above ensures that once an active process completes this phase for a new round, all active processes have completed the previous round and are in the *Synchronization phase* of the current round.

An inactive process relays messages, and while doing so, it eliminates the first CW message it receives. This guarantees that, once all inactive processes have eliminated one message, the total number of messages in the network equals the number of active processes. Therefore, once all processes except the one with the minimum identifier have been eliminated, only a single message remains in the network.

The algorithm terminates after $|\mathsf{ID}_{\min}| = O(\log \mathsf{ID}_{\min})$ rounds. At this point, the only remaining active process is the one with the minimum identifier. This process can detect this locally and communicates it to all other processes (which are, by construction, inactive) by sending an additional message in the CCW direction (this is the *Termination phase*).

All other processes detect termination when they receive two consecutive messages in the CCW direction: the first sent during the *Zero Signaling phase* by the process with the minimum identifier, and the second sent during the *Termination phase*. Therefore, we have the following theorem.

▶ **Theorem 3.** *There exists a quiescently terminating leader election algorithm for oriented rings in which each process sends $O(U \log \mathsf{ID}_{\min})$ messages clockwise and $O(\log \mathsf{ID}_{\min})$ messages counterclockwise, for a total of $O(nU \log \mathsf{ID}_{\min})$ messages.*

5 Randomized Leader Election

In our randomized algorithm, each process samples its ID uniformly at random from the set $\{0, 1, 2, \ldots, 2^{\lceil c_1 \log U \rceil} - 1\}$. With probability at least $1 - U^{2-c_1}$, all processes receive distinct identifiers. The bit-by-bit comparison algorithm from Section 4 is then executed using these random identifiers.

This sampling strategy yields a useful property: for any fixed bit index i, every maximal consecutive sequence of processes whose bit $\mathsf{ID}[i]$ equals 1 has length at most $\lceil c_2 \log U \rceil$ with probability at least $1 - U^{1-c_2}$. This property reduces the number of steps in the *Synchronization* and *No-Zero Checking* phases of the algorithm in Section 4 from U to $\lceil c_2 \log U \rceil$, thereby improving the message complexity and establishing our final theorem.

▶ **Theorem 4.** *For any constant $c > 0$, there exists a randomized quiescently terminating leader election algorithm for anonymous oriented rings with a success probability of $1 - U^{-c}$ in which each process sends $O(\log^2 U)$ messages clockwise and $O(\log U)$ messages counterclockwise, for a total of $O(n \log^2 U)$ messages.*

References

1 H. Attiya and J. L. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley, 2004.

2 P. Boldi, S. Shammah, S. Vigna, B. Codenotti, P. Gemmell, and J. Simon. Symmetry breaking in anonymous networks: Characterizations. In *ISTCS 1996*, pages 16–26, 1996.

3 P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Mathematics*, 243(1):21–66, 2002. doi:10.1016/S0012-365X(00)00455-6.

4 C. Cachin, R. Guerraoui, and L. Rodrigues. *Introduction to Reliable and Secure Distributed Programming*. Springer, 2011.

5 A. Casteigts, Y. Métivier, J.M. Robson, and A. Zemmari. Counting in one-hop beeping networks. *Theoretical Computer Science*, 780:20–28, 2019. doi:10.1016/J.TCS.2019.02.009.

6 A. Casteigts, Y. Métivier, J.M. Robson, and A. Zemmari. Design patterns in beeping algorithms: Examples, emulation, and analysis. *Information and Computation*, 264:32–51, 2019. doi:10.1016/J.IC.2018.10.001.

7 K. Censor-Hillel, S. Cohen, R. Gelles, and G. Sela. Distributed computations in fully-defective networks. *Distributed Computing*, 36(4):501–528, 2023. doi:10.1007/S00446-023-00452-2.

8 J. Chalopin, E. Godard, and Y. Métivier. Election in partially anonymous networks with arbitrary knowledge in message passing systems. *Distributed Computing*, 25(4):297–311, 2012. doi:10.1007/S00446-012-0163-Y.

9 A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *DISC 2010*, volume 6343 of *Lecture Notes in Comput. Sci.*, pages 148–162. Springer, 2010. doi:10.1007/978-3-642-15763-9_15.

10 A. Czumaj and P. Davies. Leader election in multi-hop radio networks. *Theoretical Computer Science*, 792:2–11, 2019. doi:10.1016/J.TCS.2019.02.027.

11 F. Dufoulon, J. Burman, and J. Beauquier. Beeping a deterministic time-optimal leader election. In *DISC 2018*, volume 121 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.DISC.2018.20.

12 P. Flocchini, E. Kranakis, D. Krizanc, F. L. Luccio, and N. Santoro. Sorting and election in anonymous asynchronous rings. *Journal of Parallel and Distributed Computing*, 64(2):254–265, 2004. doi:10.1016/J.JPDC.2003.11.007.

13 F. Frei, R. Gelles, A. Ghazy, and A. Nolin. Content-oblivious leader election on rings. In *DISC 2024*, volume 319, pages 26:1–26:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.DISC.2024.26.

14 K.-T. Förster, J. Seidel, and R. Wattenhofer. Deterministic leader election in multi-hop beeping networks. In *DISC 2014*, volume 8784 of *Lecture Notes in Comput. Sci.*, pages 212–226. Springer, 2014.

15 Alon Itai and Michael Rodeh. Symmetry breaking in distributed networks. *Inf. Comput.*, 88(1):60–87, July 1990. doi:10.1016/0890-5401(90)90004-2.

16 S. Kutten, W. K. Moses Jr., G. Pandurangan, and D. Peleg. Singularly optimal randomized leader election. In *DISC 2020*, volume 179 of *LIPIcs*, pages 22:1–22:18, 2020. doi:10.4230/LIPICS.DISC.2020.22.

17 M. Raynal. *Fault-Tolerant Message-Passing Distributed Systems: An Algorithmic Approach*. Springer, 2018.

18 N. Santoro. *Design and Analysis of Distributed Algorithms*. John Wiley & Sons, 2007.

19 N. Santoro and P. Widmayer. Time is not a healer. In *STACS 1989*, volume 349 of *Lecture Notes in Comput. Sci.*, pages 304–313. Springer, 1989. doi:10.1007/BFB0028994.

20 R. Vacus and I Ziccardi. Minimalist leader election under weak communication. In *PODC 2025*. ACM, 2025.

21 M. Yamashita and T. Kameda. Computing on anonymous networks: Part I - Characterizing the solvable cases. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):69–89, 1996. doi:10.1109/71.481599.