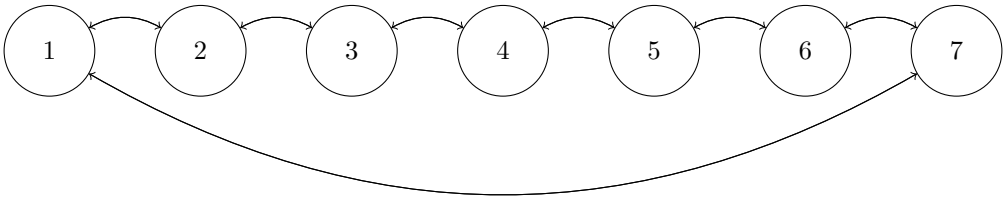
Leader Election in Asynchronous Rings

Tsimafei Baliukonis

January 2025

1 Problem Statement

We have an unknown number of asynchronous processors, each with a unique ID. Each processor is connected to two neighbors in a cyclic topology, forming a ring. Processors can send and receive distinguishable messages to and from their neighbors. However, a processor does not know which neighbor is to its "right" or "left" in the ring.



The goal is to determine which processor has the maximum ID while minimizing the number of messages exchanged.

2 Proposed Algorithm

In 1980, Hirschberg and Sinclair proposed an algorithm that requires $O(n \log n)$ messages, where n is the number of processors in the system.

The algorithm starts by having each processor declare itself as a candidate in the election. The process then proceeds as follows.

At stage i, a candidate checks whether it has the highest ID among 2^i neighbors in both directions. If it does, it moves to the next stage. Otherwise, it withdraws from the election. If, after 2^i steps in one direction, the candidate returns to itself without encountering a higher ID, it declares itself the leader.

Only the processor with the maximum ID will declare itself as the leader, while all other candidates will eventually withdraw.

3 Message Handling

Messages are distinguishable, allowing processors to respond to the sender or forward the message in the same direction.

At stage i, each processor does the following:

- It checks if it is still a candidate.
- If true, it sends the following message to both neighbors:

$$\{\text{from}, \text{process\_id}, 0, 2^i\}$$

- The processor waits for two responses:
  - If it receives a message of the form {no, process_id}, it withdraws.

- If it receives two messages of the form {ok, process_id}, it proceeds to stage $i + 1$.
- If it receives a message of the form {from, process_id, x, 2^i}, that completes a full cycle, it declares itself as the leader.

Additionally, each processor must handle the messages it receives from other processors:

- If a processor receives a message of the form {from, sender_id, x, y}:
  - If sender_id < id, it sends back a {no, sender_id} message.
  - If sender_id > id, it forwards the message further:
    - If $x + 1 = y$, it returns an {ok, sender_id} message.
    - Otherwise, it forwards {from, sender_id, x + 1, y} to the next neighbor.
- If a processor receives a message of the form {ok, adressant_id} or {no, adressant_id} and adressant_id $\neq$ id, it forwards the message in the same direction.

4 Analysis of Complexity

At stage i, all remaining candidates send messages. The maximum number of participants at this stage is approximately $n/(2^i + 1)$. Each message travels a distance of up to 2^i in both directions and returns. Therefore, each candidate sends 2^{i+2} messages at stage i.

The total number of messages sent is bounded by:

$$4 \cdot \left(n + 2 \cdot \left\lfloor \frac{n}{2} \right\rfloor + 4 \cdot \left\lfloor \frac{n}{3} \right\rfloor + 8 \cdot \left\lfloor \frac{n}{5} \right\rfloor + \dots \right)$$

Since each term is not greater than $2n$, and there are at most $\log n$ terms, the total complexity is $O(n \log n)$ messages.

References

[1] D.S. Hirschberg and J.B. Sinclair. "Decentralized Extrema Finding in Circular Configurations of Processors". In: *Communications of the ACM* 23.11 (1980), pp. 627–628.