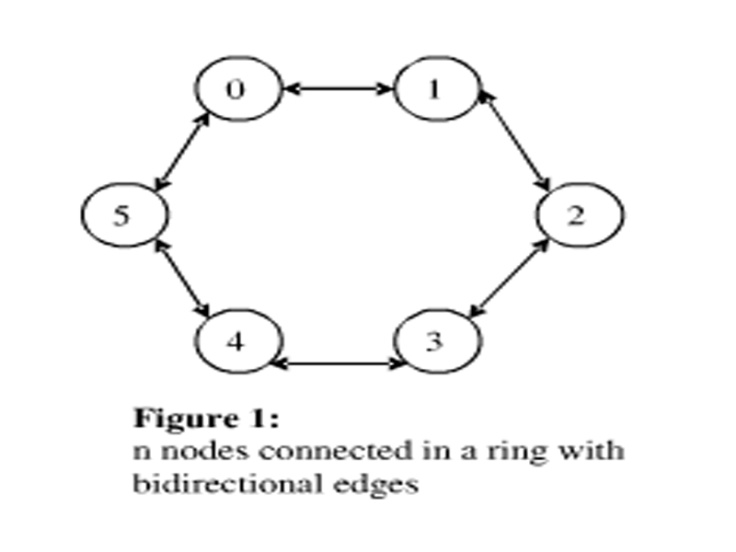
**HS Leader Election Algorithm**

**Leader Election**

The problem originally arose in the study of token ring networks. The idea is that one token will circulate around the ring and only the owner of the token has the sole right to initiate communication. In a network, if multiple nodes (or processes) simultaneously attempt to communicate, the communications can potentially interfere with each other. Sometimes the token may be lost or the owner of the node has malfunctioned. In this case, the active nodes of the network will run a leader election algorithm to elect a leader. The leader will be the owner of the token. In the remainder of the paper, node s and processes will be used interchangeably.

**Leader Election**

The network is arranged in a ring. The communication links (edges) are bidirectional. This means if node I has an edge to node j, then that implies node jhas and edge to node i. Each node is assigned a positive unique identification number (UID). The nodes of the ring are assigned indices from 0 to n - 1. Each node only has access to its own local information. A node I does not have access to node j’s local variables. Each node is able to distinguish its left neighbor from its right neighbor.



**Asynchronous Mode**

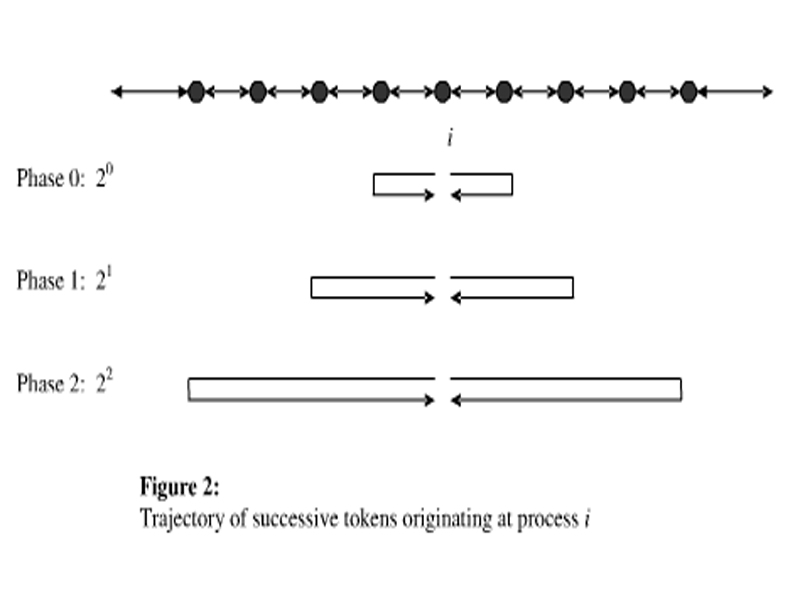
* **Nodes:** In this model, we assume each node of the network will take steps at arbitrary order and at arbitrary relative speeds. By arbitrary order, it means that the nodes do not have to follow a specific pattern of execution, such as taking turns. At any given instance any node can possibly execute. By arbitrary relative speeds, it means that not all nodes are identical. Some may operate faster than others (i.e. different processor speeds).
* Channels: In the asynchronous model, message delivery time is also arbitrary, but we know that every message sent is eventually delivered. The channels are known as Reliable FIFO channels. Reliable in that there is no message lost, and messages in the channel are treated in a FIFO manner.

**Problem Specification**

The node with the highest UID will output the decision that it is the leader**.**

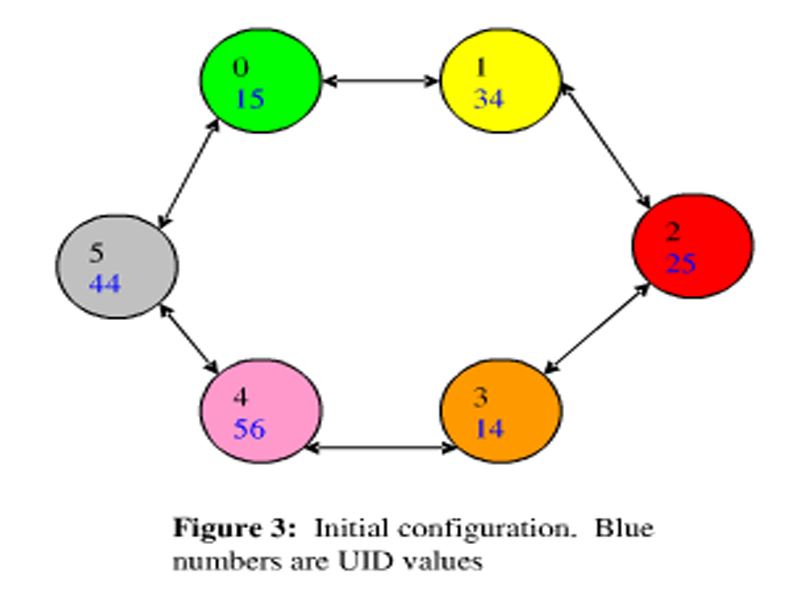
**Informal Description of the algorithm**

Each process I operates in phases 0, 1, 2,....In each phase l, process I sends out "tokens" containing its UID ui in both directions. These are intended to travel distance 2l, then return to their origin i (See figure 2). If both tokens make it back safely, process I continues with the following phase. However, the tokens might not make it back safely. While a uitoken is proceeding in the outbound direction, each other process jon ui's path compares uiwith its own UID uj. If ui<uj, then j simply discards the token, whereas if ui>uj, then j relays ui. If ui= uj, then it means that process j has received its own UID before the token has turned around, so process j elects itself as the leader. All process always relay all tokens in the inbound direction.

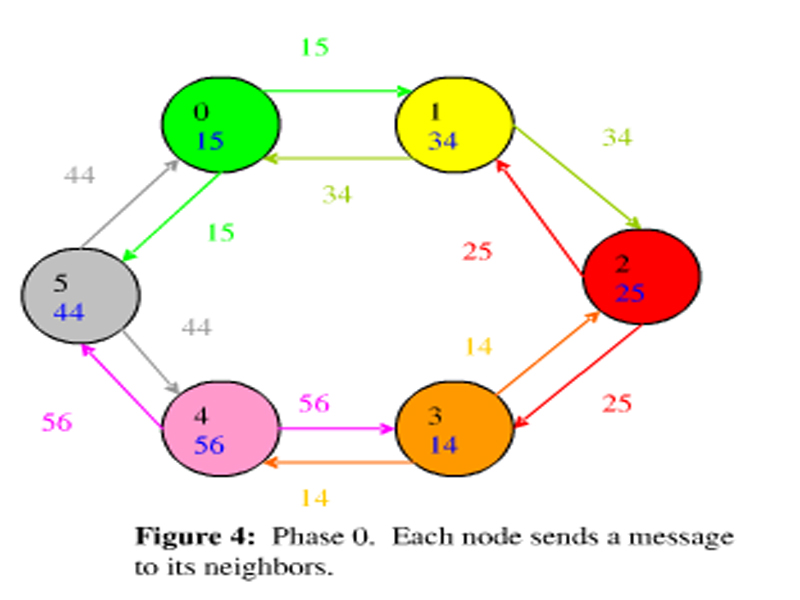


**Sample execution**

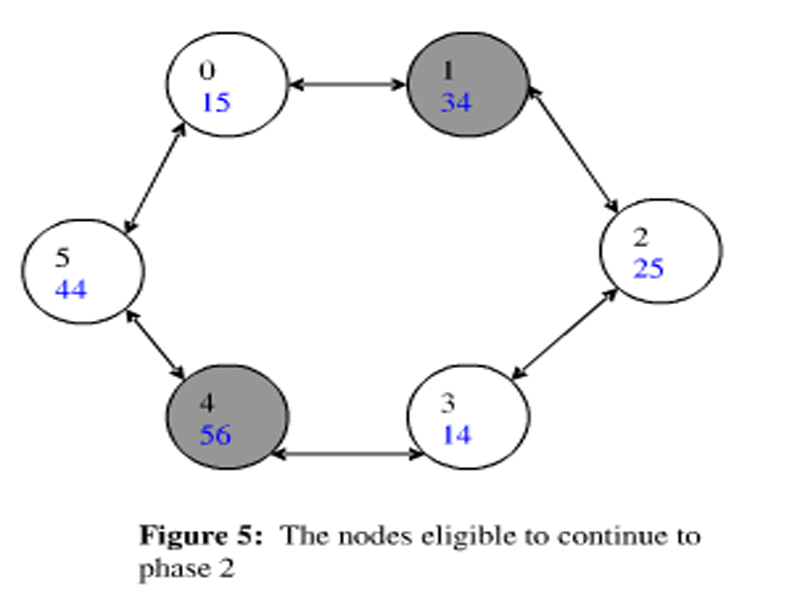
* For demonstration purposes, we will ignore the asynchronous nature of the system and assume that all nodes operate at the same speed and at the same time.
* Initial network configuration with UID values in blue. (Figure 3)

****

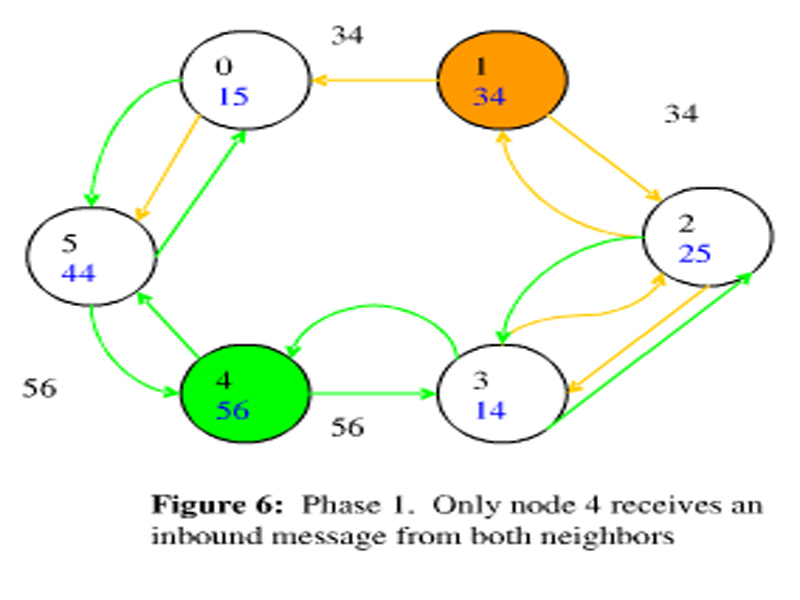
* The algorithm starts in phase 0, and each node will send an outbound message to its neighbors. The message is intended to travel a distance of 20 hops. (Figure 4)



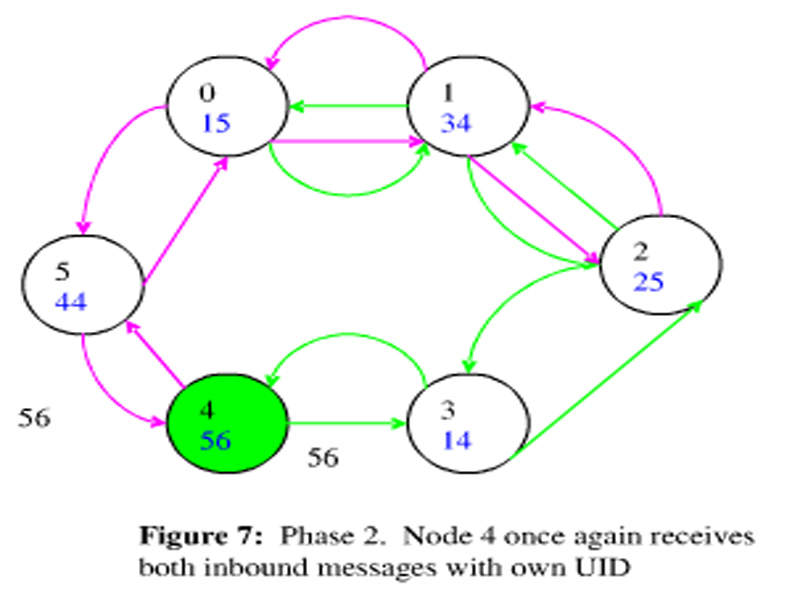
* When node 0 receives UID 44, it will compare with its own UID. Since 44 > 15, and the message already traveled 20hops, it will reverse the direction of the message and send UID 44 inbound back to node 5. Then node 0 will receive UID 34 from node 1, compare 34 > 14, and it will relay UID 34 inbound back to node 1.
* When node 1 receives UID 15 from its left neighbor, it will compare with its own UID. Since 15 > 34 is false, it will simply discard UID 15. Node 1 receives a UID 25 from node 2. 25 > 34 is false, and the message is discarded.
* Similar comparisons are made by all the nodes. This is what happened:
* Node 2 relays UID 34 back to node 1. Node 2 discards UID 14 from node 3
* Node 3 relays UID 25 back to node 2. It also relays UID 56 back to node 4
* Node 4 discards UID 14 and UID 44
* Node 5 relays UID 56 back to node 4, ad discards UID 15
* The nodes that received an inbound message with its OWN UID from both neighbors are qualified to proceed to the next phase. They are indicated in gray. (Figure 5)



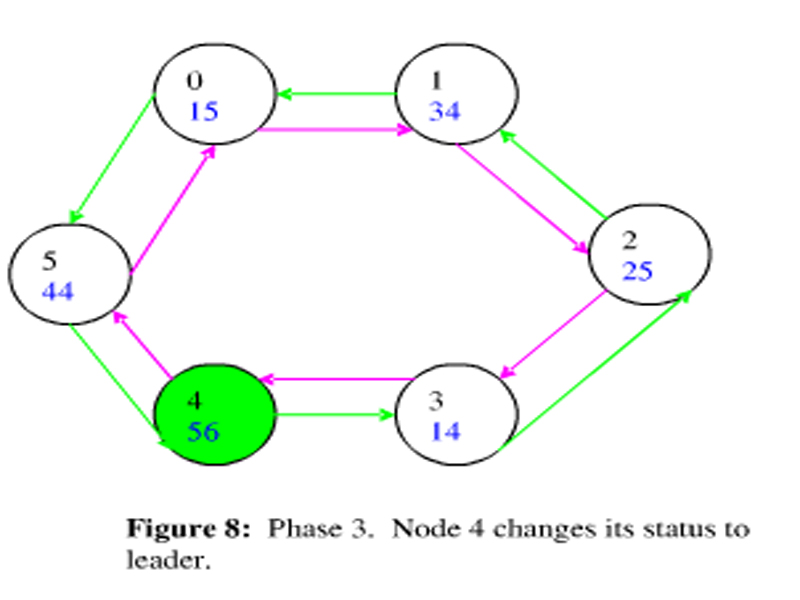
* Now we advance to phase 1, and each active node will send an outbound message to its neighbors. The message this time is intended to travel a distance of 21 hops. (Figure 6). The straight arrows represent outbound and the curved arrows represents inbound.



* Only nodes 1 and 4 are active, only they can initiate messages in this phase
* Node 1 sends UID 34 outbound to node 2 and node 0
* Node 2 compares 34 > 25, and relays it outbound to Node 3.
* Node 3 compares 34 > 14, and since the message has traveled 21 hops, it sends UID 34 inbound to node 2. Node 2 relays UID 34 inbound to Node 1.
* Node 0 compares with UID 34, and relays it outbound to node 5. Node 5 compares and drops the message. Thus, node 1 will not receive a message from its counter-clockwise neighbor.
* Node 4 sends UID 56 to its neighbors and receives inbound messages containing its own UID from both sides. It advances to the next phase.
* Phase 2. Only node 4 is active in phase 2. It is the only node that can initiate a message. The message is intended to travel 22 hops. Outgoing messages sent by node 4 in the clockwise direction are in pink. Outgoing messages in the counter-clockwise direction are in green. Straight lines are outgoing messages, curved lines are incoming. (Figure 7).



* Node 4 once again receives inbound messages containing its own UID. It will advance to the next phase.
* Phase 3. Only node 4 is active in phase 2. It is the only node that can initiate a message. Its messages are intended to travel 23 hops. (Figure 8)



* Node 4’s OUTBOUND message circulates completely around the ring, and reaches itself in the OUTBOUND direction. Thus, it declares itself the leader.

**Some benefits of the HS algorithm**

* Time Complexity: at most 3n or 5n ==> O(n) - matches lower bound
* Communication Complexity: O(n log n) - matches lower bound
* It works for both synchronous and asynchronous model without any modifications
* The Algorithm is uniform (size of network is not required to be known)
* Total number of phases: 1 + [log n]

**References**

* Nancy Lynch, Distributed Algorithm