#ASLI ENGINEERING

HS Algorithm for Leader Election

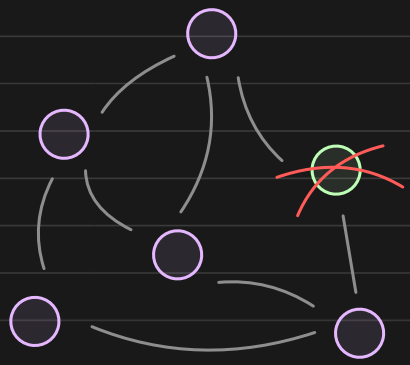
SWIPE

BY

ARPIT BHAYANI

HS algorithm for leader Election

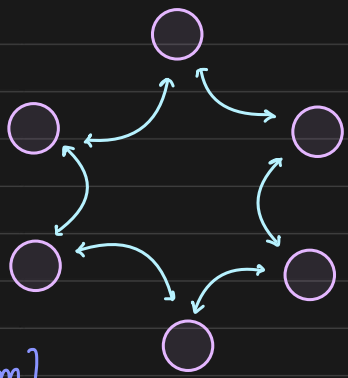
leader Election is an automated way of System Recovery. When the leader node is down, the leader Election algorithm is triggered which elects the new leader thus restoring the system



The HS algorithm

HS algorithm is also a synchronous algorithm for leader election, but it has a communication complexity ie. # messages for election is O(nlogn)

It works with any network that is arranged in a bidirectional ring. [bare minimum]



Algorithm works even when the total number of nodes are unknown

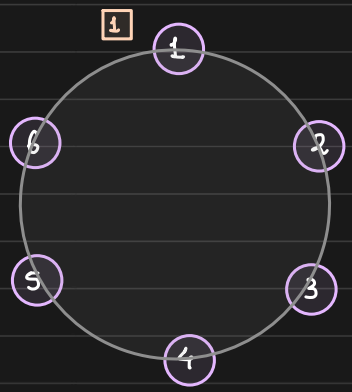
Assumptions:

- every node has a unique UID
- UIDs are comparable
- every node knows both of its immediate neighbours

The algorithm

Every node participates in the election and pitches itself to be the new leader.

Pitch: create a message with its own UID and send it to its immediate neighbours



In order to reduce the number of messages, poor candidates step out of the election sooner.

CORE IDEA: local maxima proceeds to test if global maxima

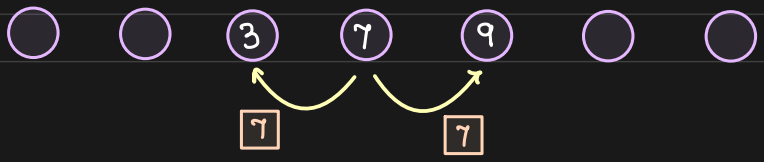
Each node operates in phases $i = 0, 1, 2, 3, \cdots$

After each phase, some nodes are eliminated from the election

Each of the participating node, in each phase i, sends message upto 2^i nodes in both the direction.

Hence, in phase 0,

- every node participates in the election
- every node sends the message with its own ID to its neighbours



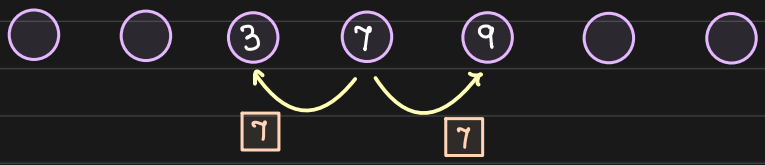
When the neighbouring nodes receive the message, it compares its own UID with the incoming UID

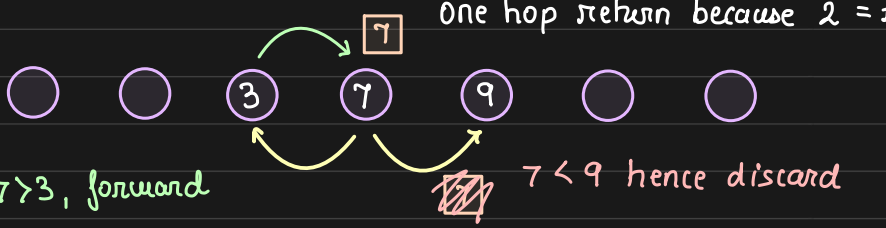
if incoming UID > own UID : forwards UID to its neighbour

if incoming UID < own UID : discard the incoming UID

if incoming UID == own UID : identifies itself as the local leader in 2^i - neighbourhood and continues.

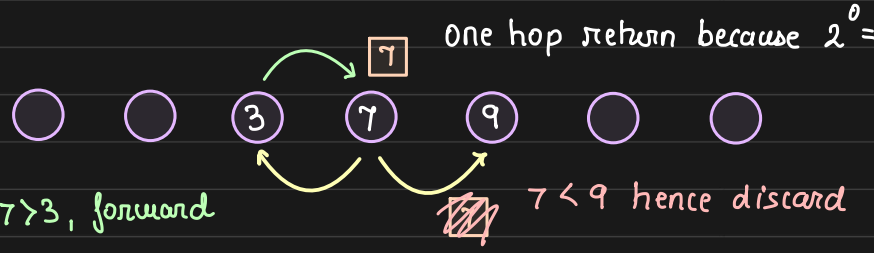
* The message is forwarded only until the desired hop. Once the hops exhaust, they travel back to the origin node





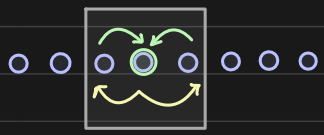
In phase 0, every node participates and it sends message that travels $2^0 = 1$ hops. The nodes upon receiving the message, either discards it or forwards it.

If the message survived for $2^i = 2^0 = 1$ hop, it is sent back to the origin node. if the origin node gets <u>both</u> the messages back it knows that it holds the highest UID in its 2^i neighbourhood



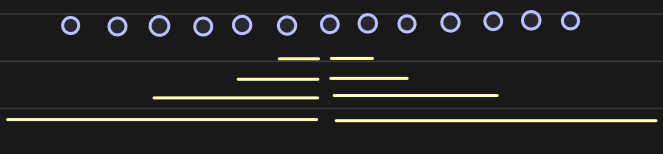
Since 7 is surrounded by 3 and 9, it will get its message back from 3 but not from 9 and hence 7 knows it is NOT the local maximum in phase $0 = 2^0 = 1$ neighbourhood

Hence, it steps out of the election process.



In the next phase, the surviving nodes send their candidature till 2, 4, 8, 16, ... hops

and this way it knows if it is local maximum in increasing neighbour hood



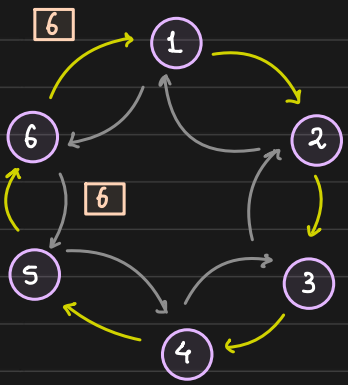
if the node continues to survive the phases it will become a stronger candidate for the new leader, as it may hold the highest UID.

As nodes leave the election, the number of messages decreases.

* Nodes who stepped out of the election, still participate in relaying.

Halting

The node with the highest UID will survive the phase and when it is the only one left, it will receive its own probe message and this tells the node that it has the highest UID and hence, the new leader.



New leader then relays the message across the ring announcing itself as the new leader and the nodes locally updates.

Key Implementation Detail

The message contains <uid, hops, direction>

and this helps node to know

1. when to stop forwarding
2. in which direction to reply