Network Computing and Efficient Algorithms

Leader Election

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

September 1, 2021

Leader Election

- We concentrate the ring topology.
- Many interesting challenges in distributed computing already reveal the root of the problem in the special case of the ring.
- Paying attention to the ring also makes sense from a practical point of view as some real world systems are based on a ring topology.
- Problem 3.1 Each node eventually decides whether it is a leader or not, subject to the constraint that there is exactly one leader.
- More formally, nodes are in one of three states:
  - Undecided
  - Leader
  - Not leader
- Initially every node is in the undecided state.

Some Definitions

Definition (Anonymous)

A system is anonymous if nodes do not have unique identifiers.

Definition (Uniform)

An algorithm is called uniform if the number of nodes n is not known to the algorithm (to the nodes, if you wish). If n is known, the algorithm is called non-uniform.

- Whether a leader can be elected in an anonymous system depends on whether the network is symmetric or asymmetric.
  - Symmetric network: ring, complete graph, complete bipartite graph, etc.
  - Asymmetric network: star, single node with highest degree, etc.

Anonymous Leader Election

Lemma

After round k of any deterministic algorithm on an anonymous ring, each node is in the same state s_k.

Proof.

Proof by induction: All nodes start in the same state. A round in a synchronous algorithm consists of the three steps sending, receiving, local computation (see Definition 1.8). All nodes send the same message(s), receive the same message(s), do the same local computation, and therefore end up in the same state. □

Anonymous Leader Election

Theorem (Anonymous Leader Election)

Deterministic leader election in an anonymous ring is impossible.

Proof.

(with above Lemma): If one node ever decides to become a leader (or a non-leader), then every other node does so as well, contradicting the problem specification for $n > 1$. This holds for non-uniform algorithms, and therefore also for uniform algorithms. Furthermore, it holds for synchronous algorithms, and therefore also for asynchronous algorithms. □

Anonymous Leader Election

- Theorem also holds for other symmetric network topology (*e.g.*, complete graphs, complete bipartite graphs).
- Theorem does generally not hold for randomized algorithms.
- randomization does not always help.

Asynchronous Ring

- Assumption
  - Non-anonymity: each node has a unique identifier.
- Trivial leader election algorithm
  - Simply elect the node with, e.g., the highest ID

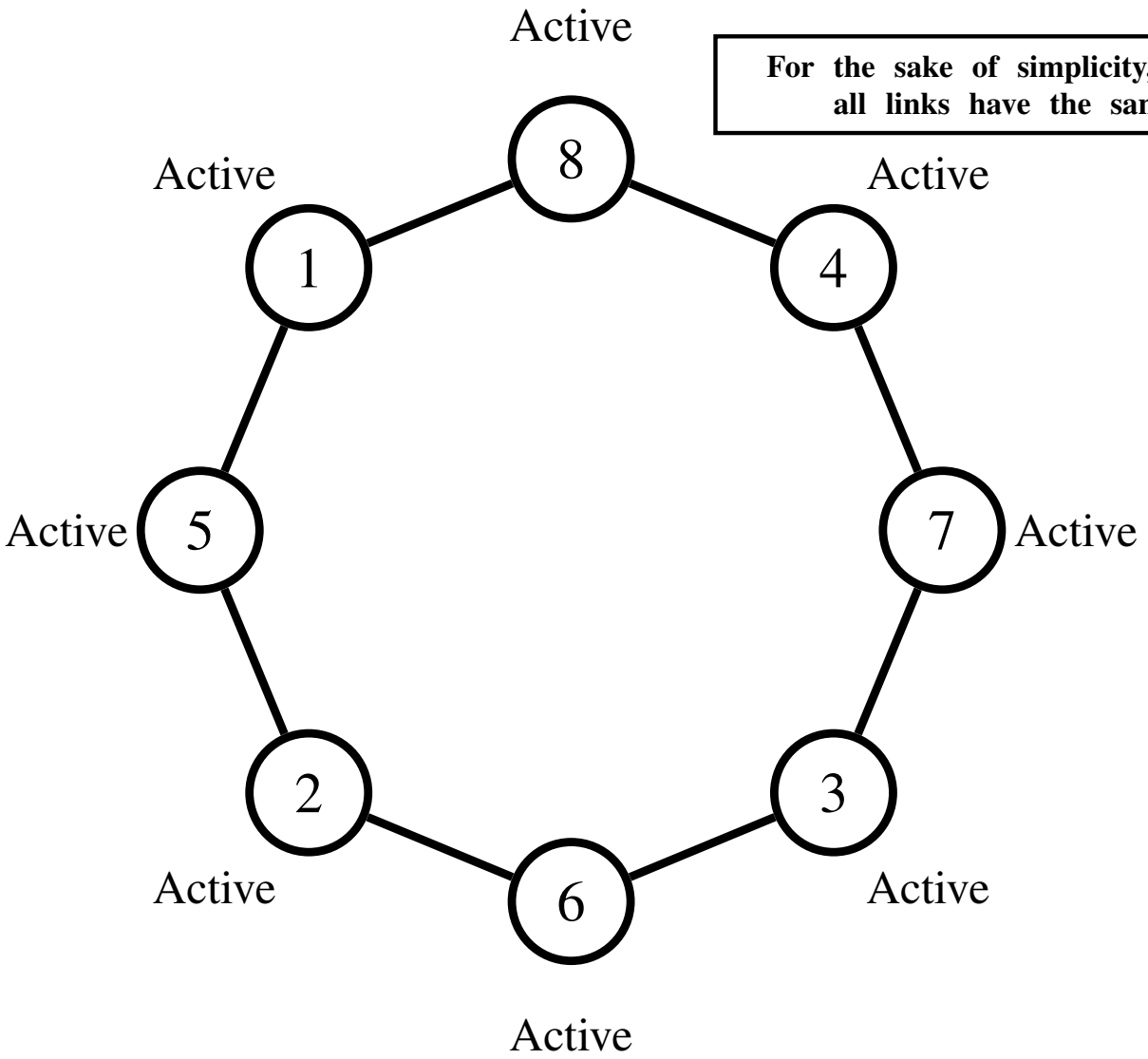
Clockwise Leader Election

- Node v wakes up, then executes.
- Node v receives a message $\longrightarrow$ wakes up.

ALGORITHM 3.6: CLOCKWISE LEADER ELECTION()

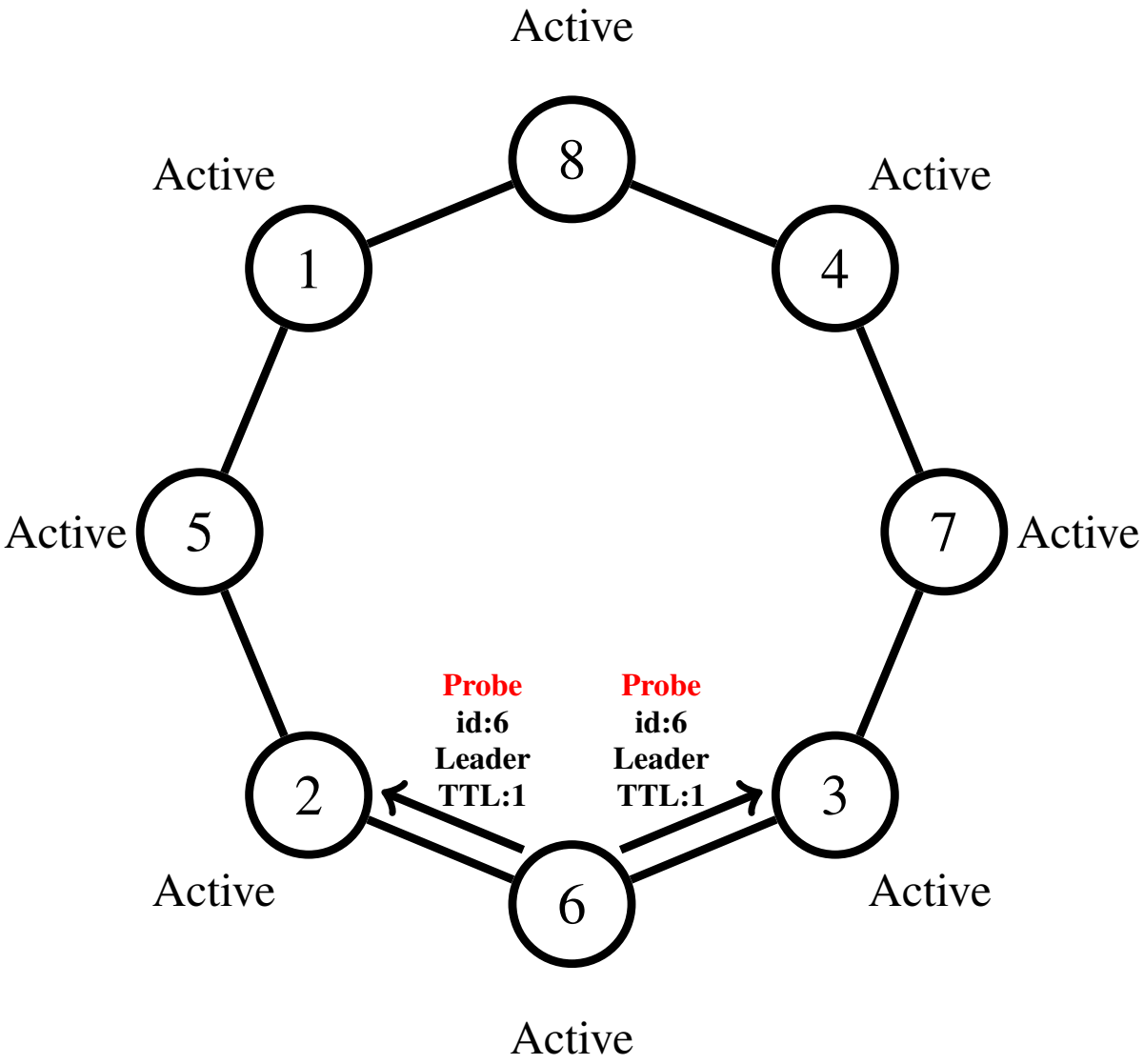
1: **Each node** v executes the following code:
2: v sends a message with its identifier (for simplicity also v) to its clockwise neighbor.
3: v sets $m := v$ {the largest identifier seen so far }
4: **if** v receives a message w with $w > m$ **then**
5: v forwards message w to its clockwise neighbor and sets $m := w$.
6: v decides not to be the leader, if it has not done so already.
7: **if** v receives its own identifier v **then**
8: v decides to be the leader.

Example

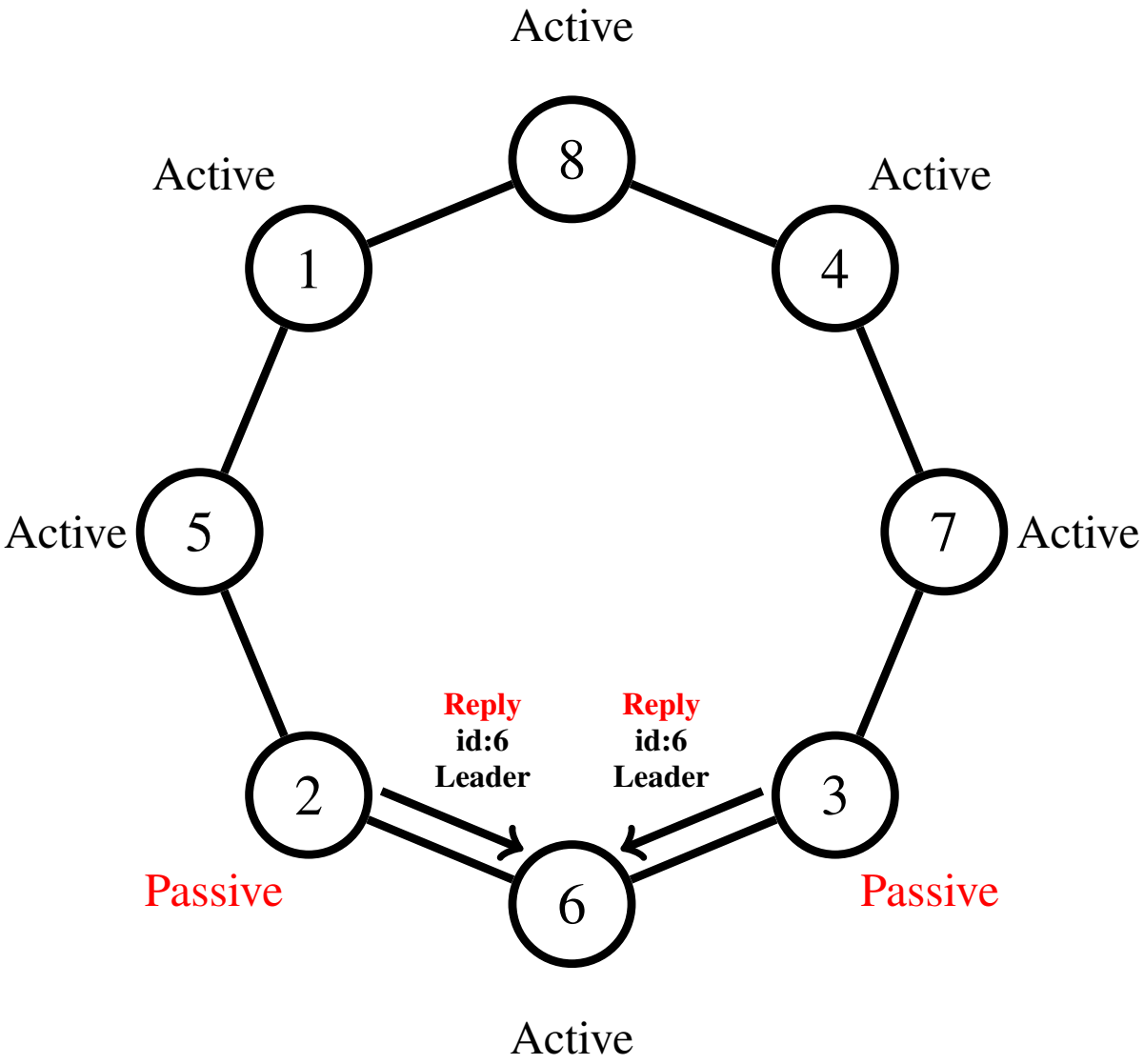


For the sake of simplicity, we assume all links have the same rate.

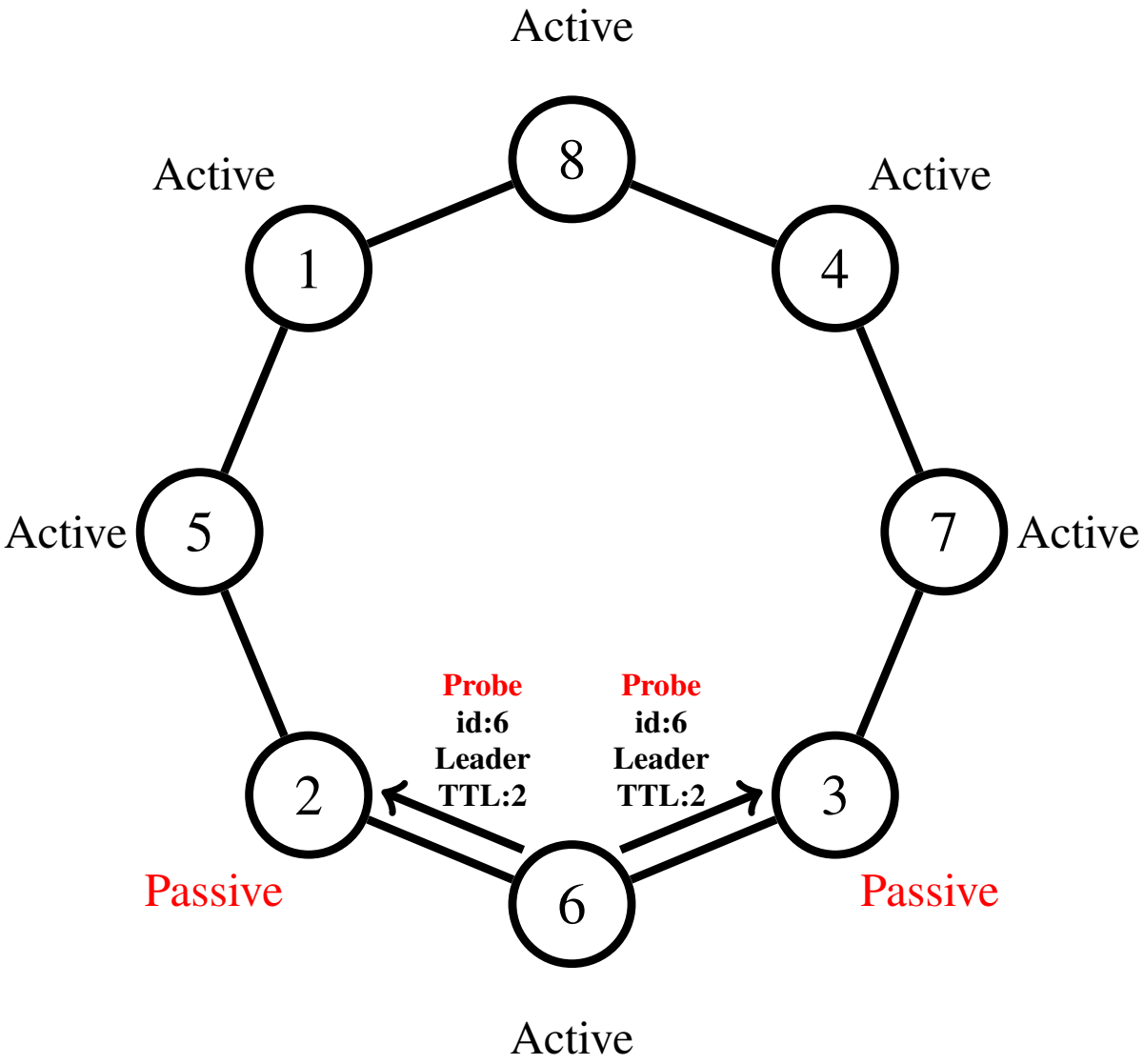
Example


Active
8
Active
Active
1
4
Active
5
7
Active
Probe
id:6
Leader
TTL:1
Probe
id:6
Leader
TTL:1
2
3
Active
Active
6
Active

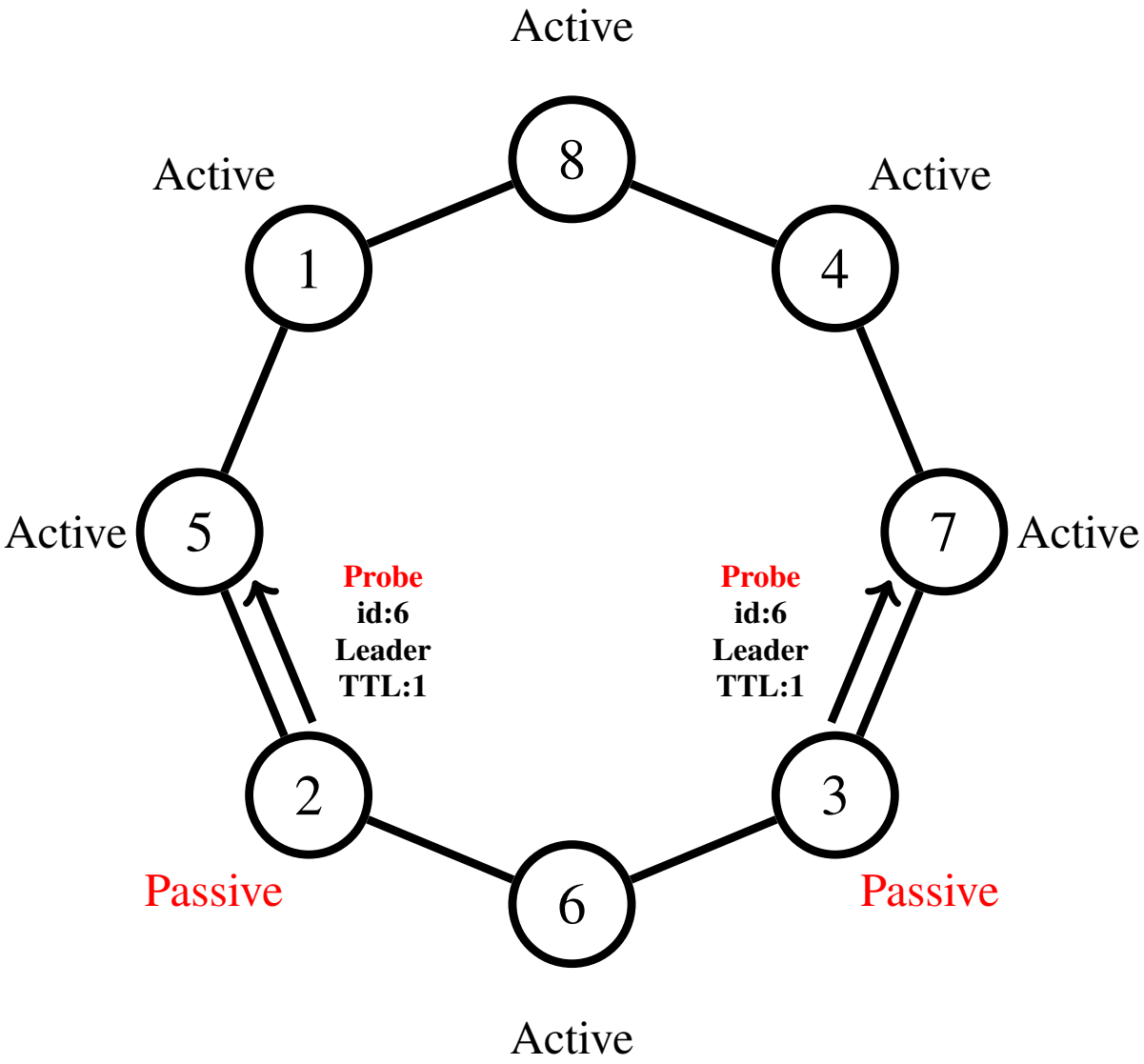
Example


Active
8
Active
Active
1
4
Active
5
7
Active
Reply
id:6
Leader
Reply
id:6
Leader
2
3
Passive
Passive
6
Active

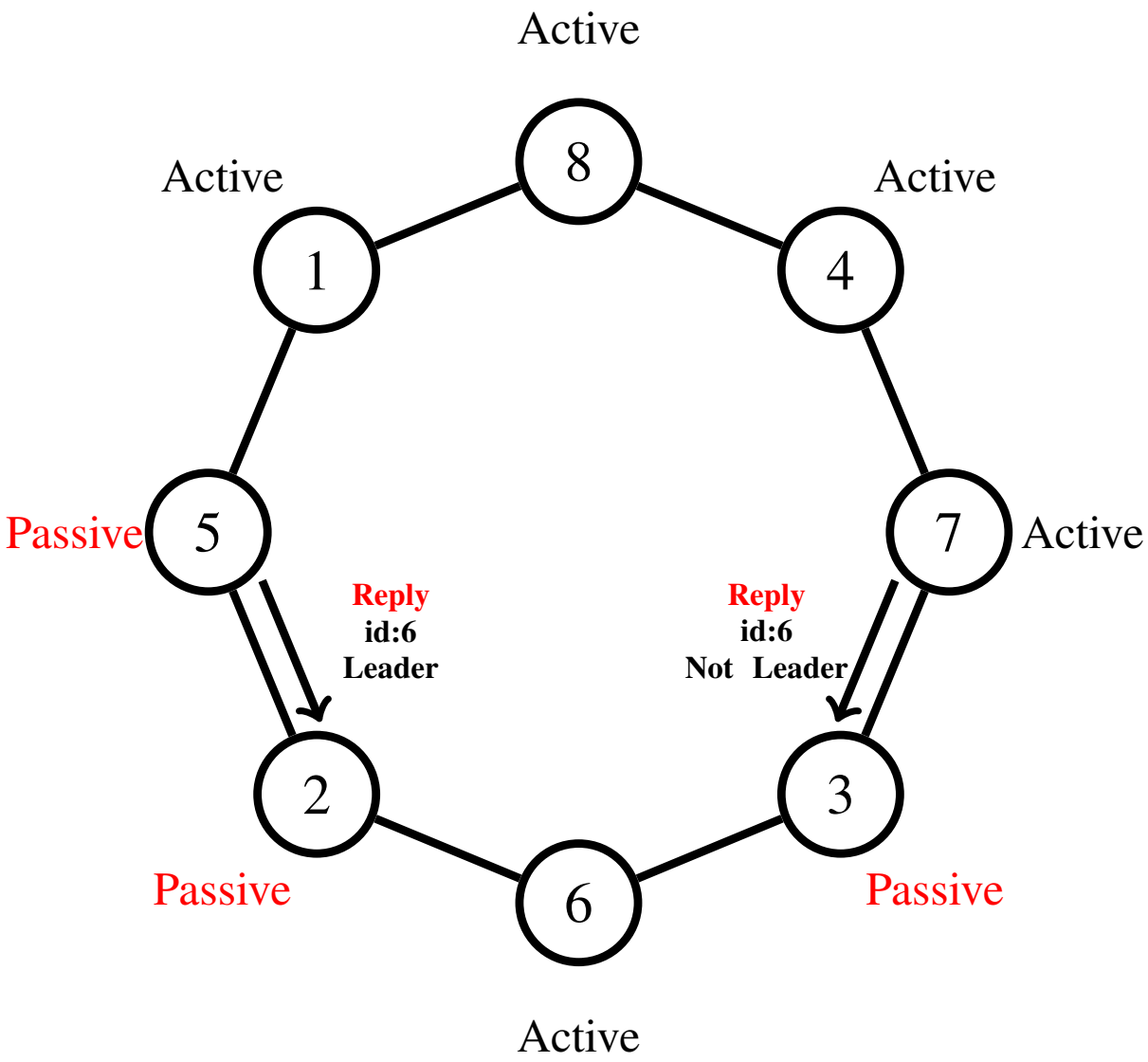
Example


Active
8
Active
Active
1
4
Active
5
7
Active
Probe
id:6
Leader
TTL:2
Probe
id:6
Leader
TTL:2
2
3
Passive
Passive
6
Active

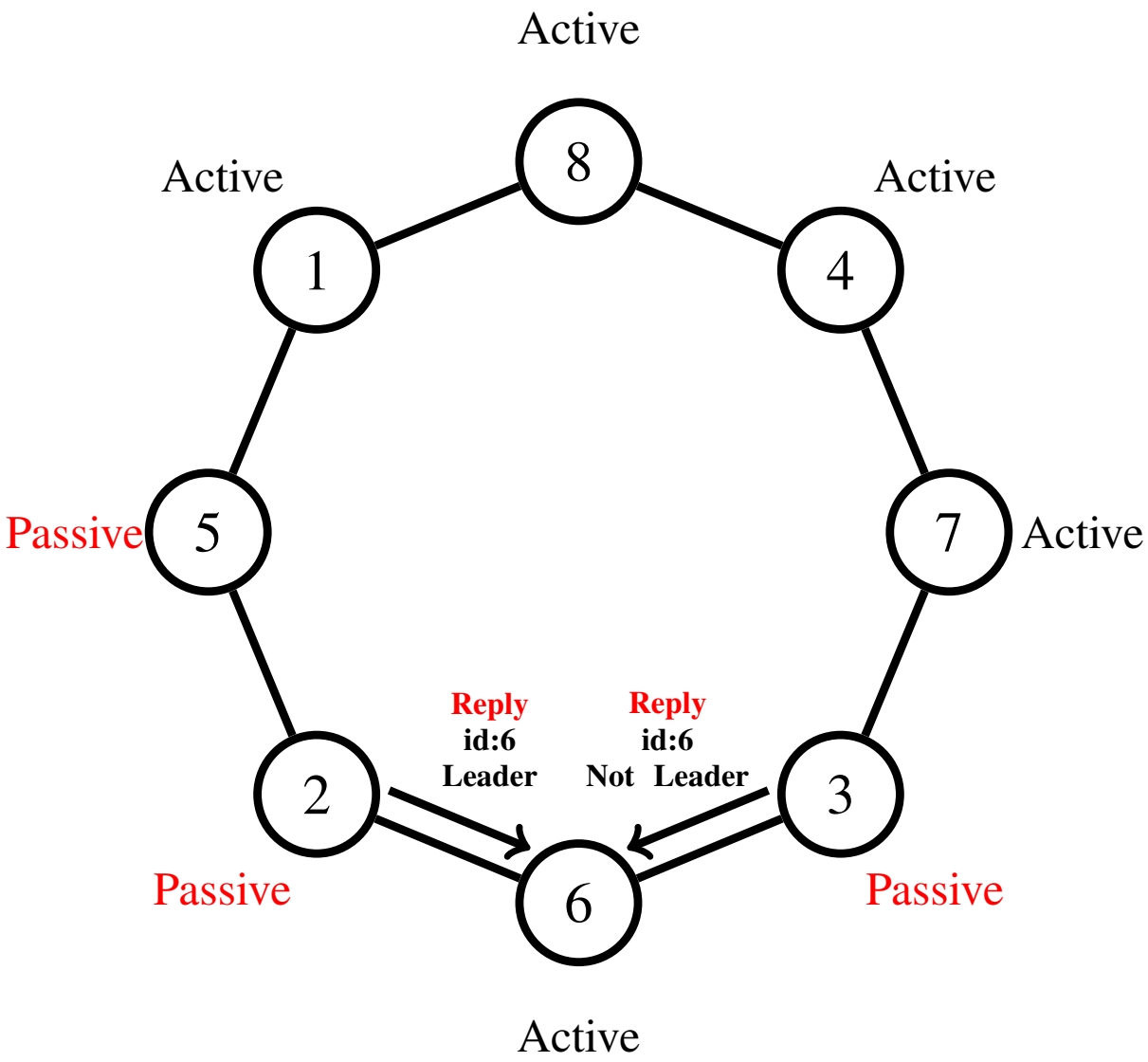
Example


Active
8
Active
Active
1
4
Active
5
7
Active
Probe
id:6
Leader
TTL:1
Probe
id:6
Leader
TTL:1
2
3
Passive
Passive
6
Active

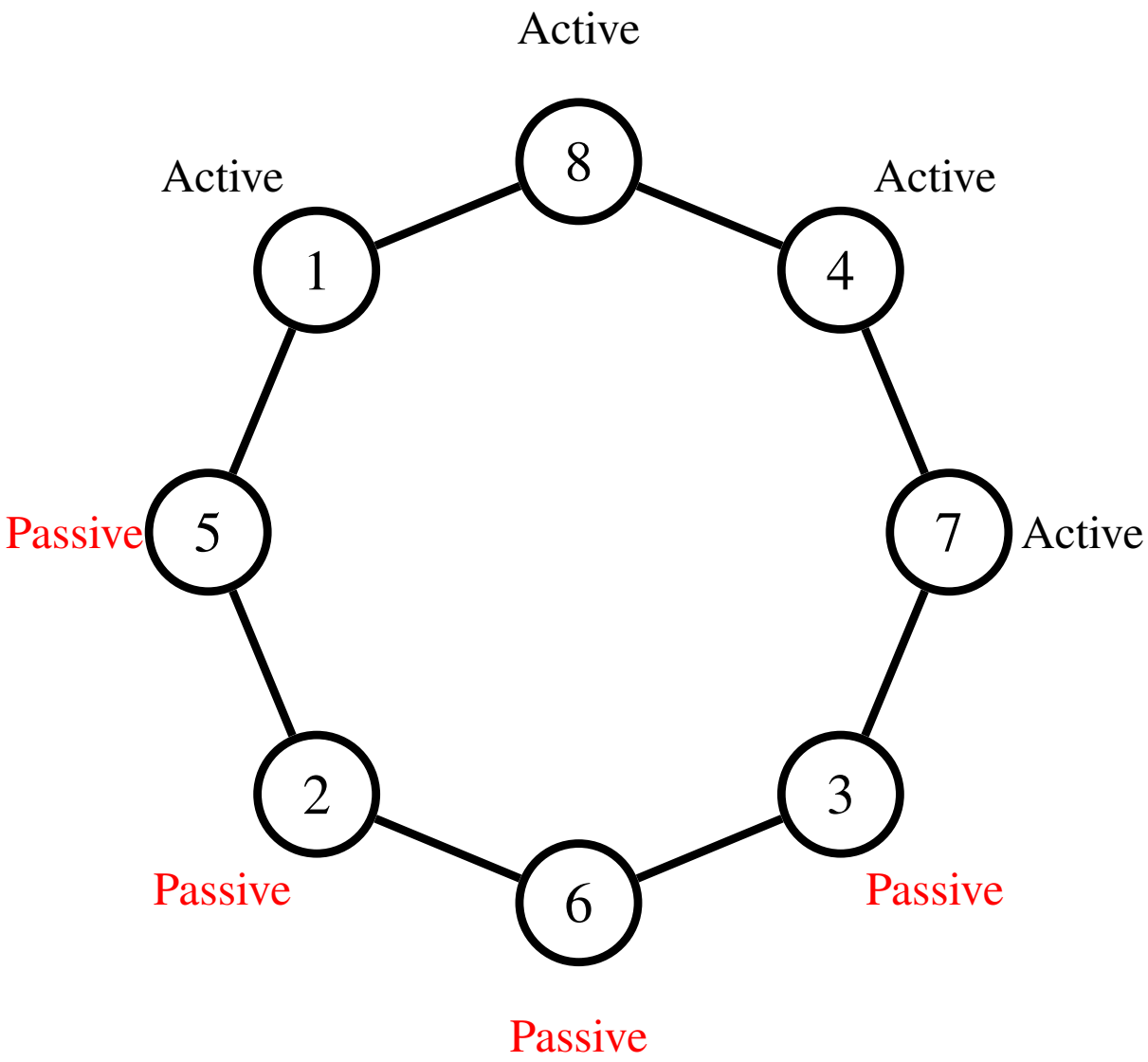
Example


Active
8
Active
Active
1
4
Passive
5
7
Active
Reply
id:6
Leader
Reply
id:6
Not Leader
2
3
Passive
Passive
6
Active

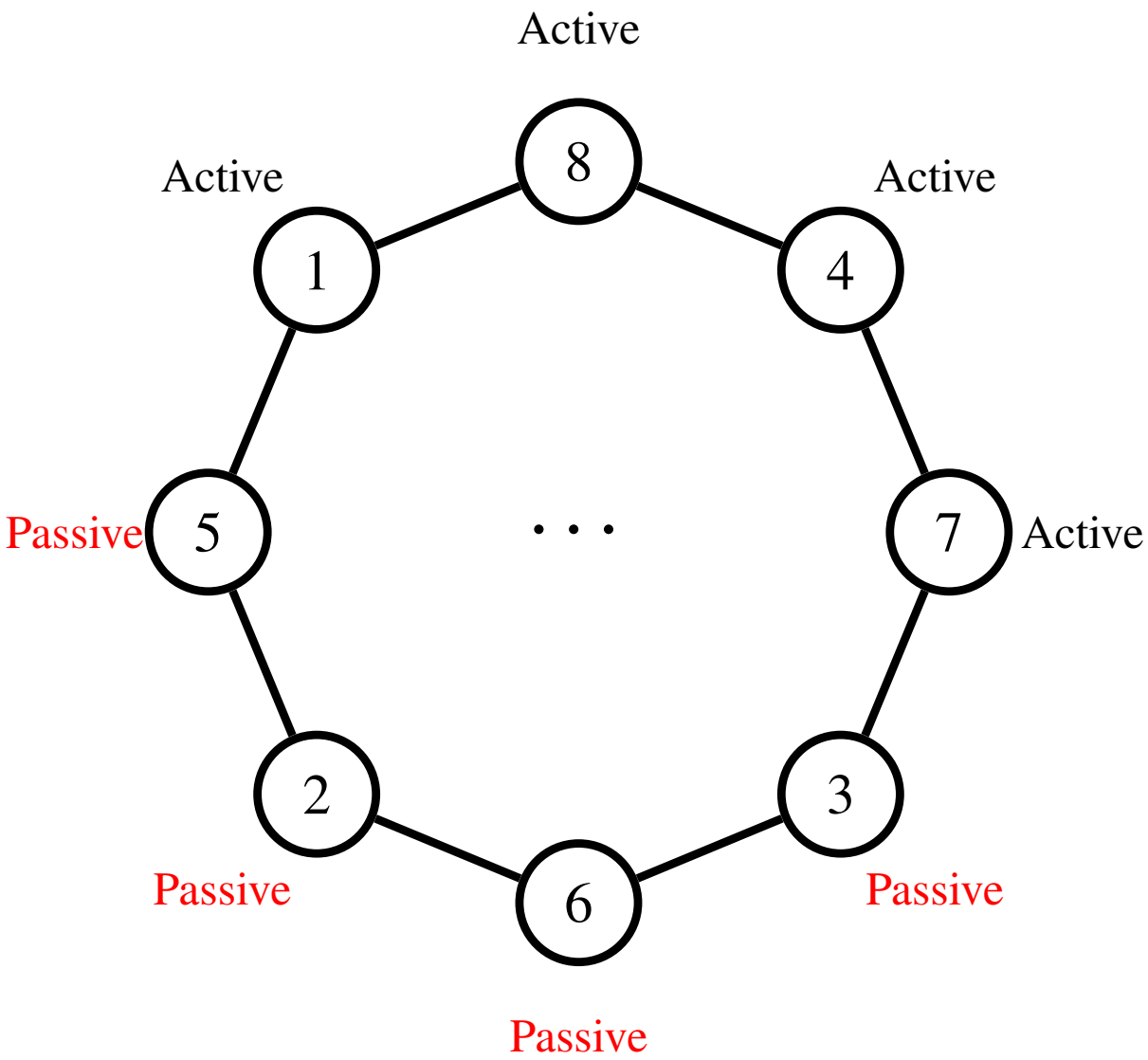
Example


Active
8
Active
Active
1
4
Passive
5
7
Active
Reply
id:6
Leader
Reply
id:6
Not Leader
2
3
Passive
Passive
6
Active

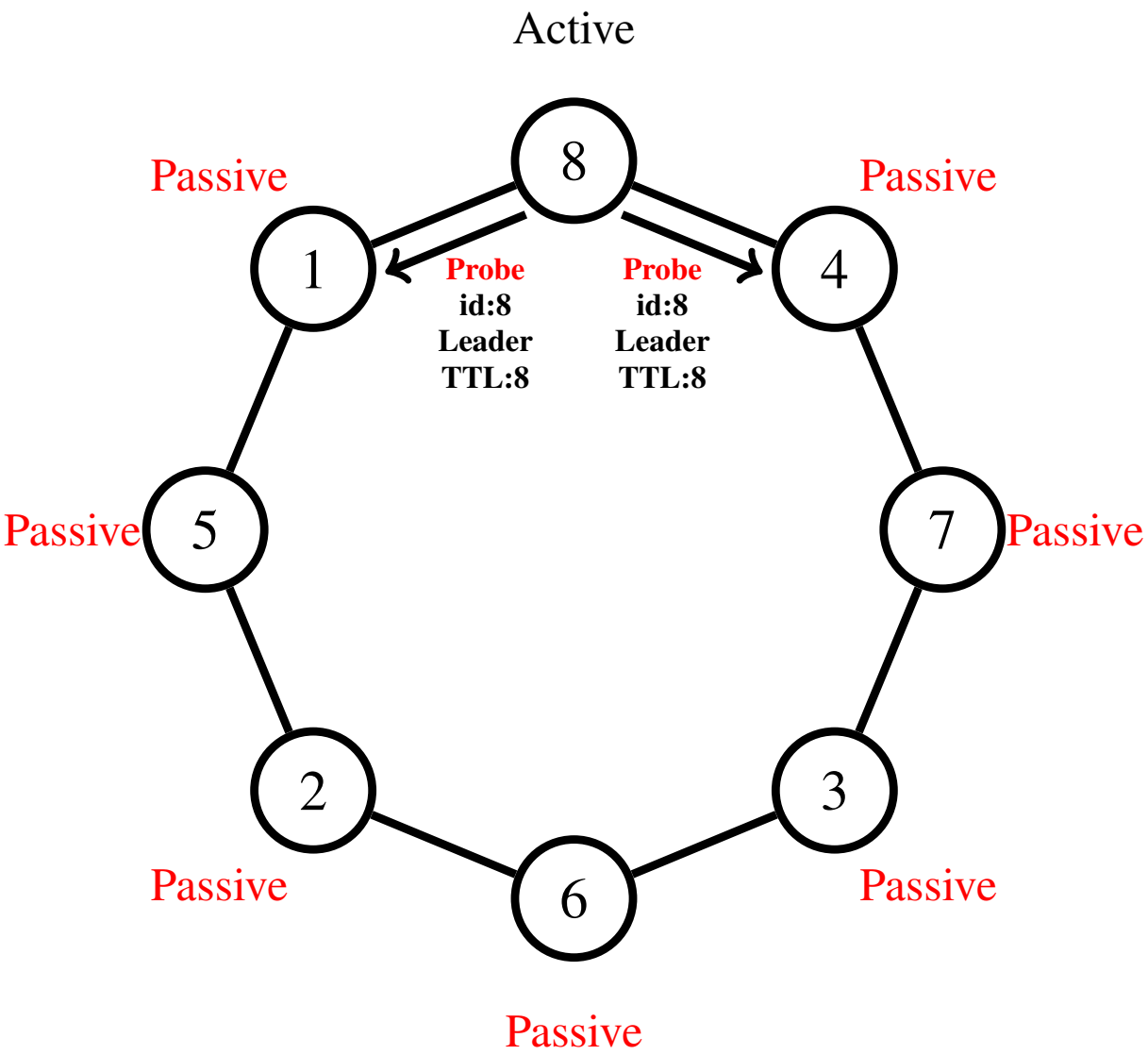
Example


Active
8
Active
Active
1
4
Passive
5
7
Active
2
3
Passive
Passive
6
Passive

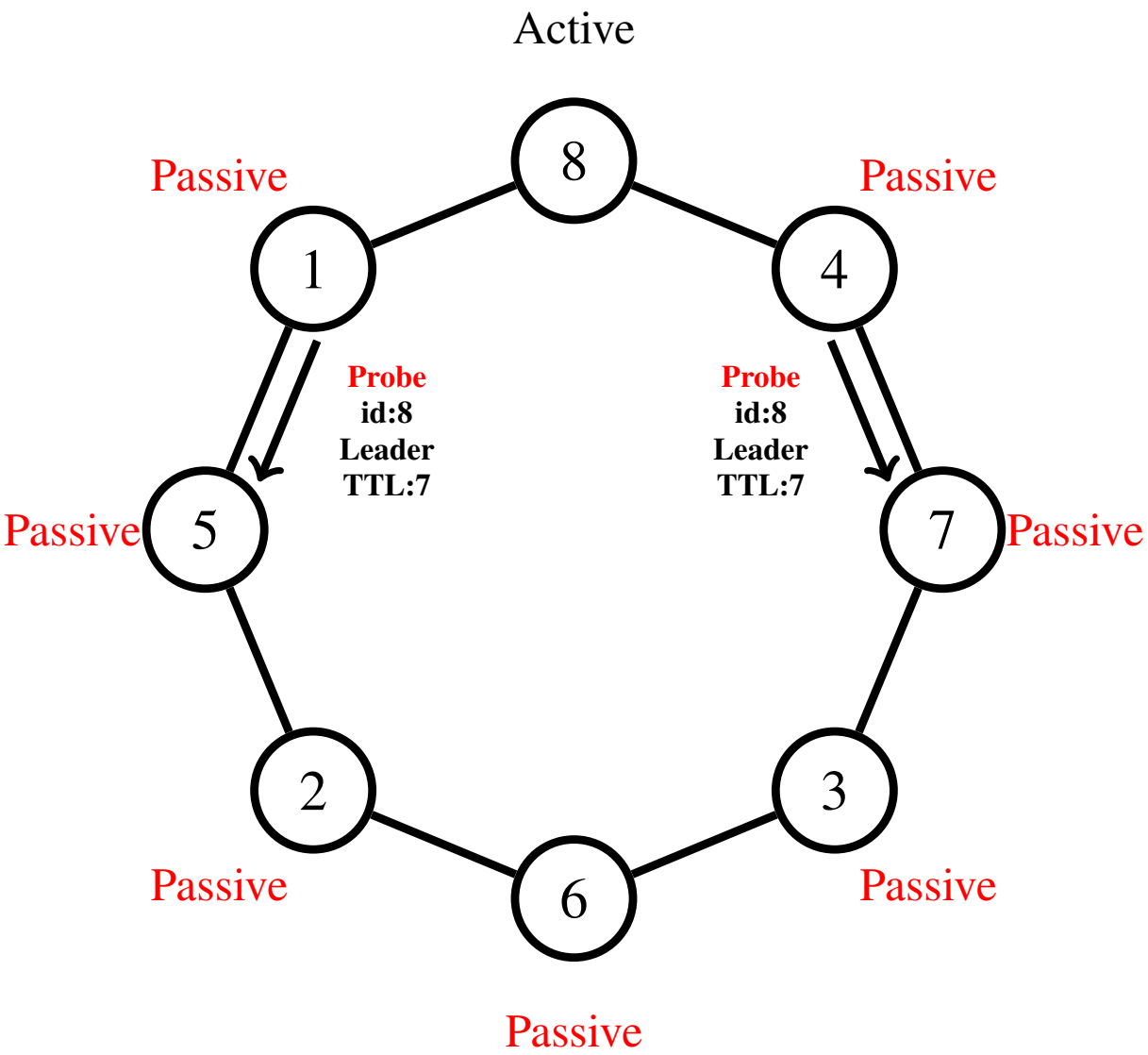
Example


Active
8
Active
Active
1
4
Passive
5
• • •
7
Active
2
3
Passive
Passive
6
Passive

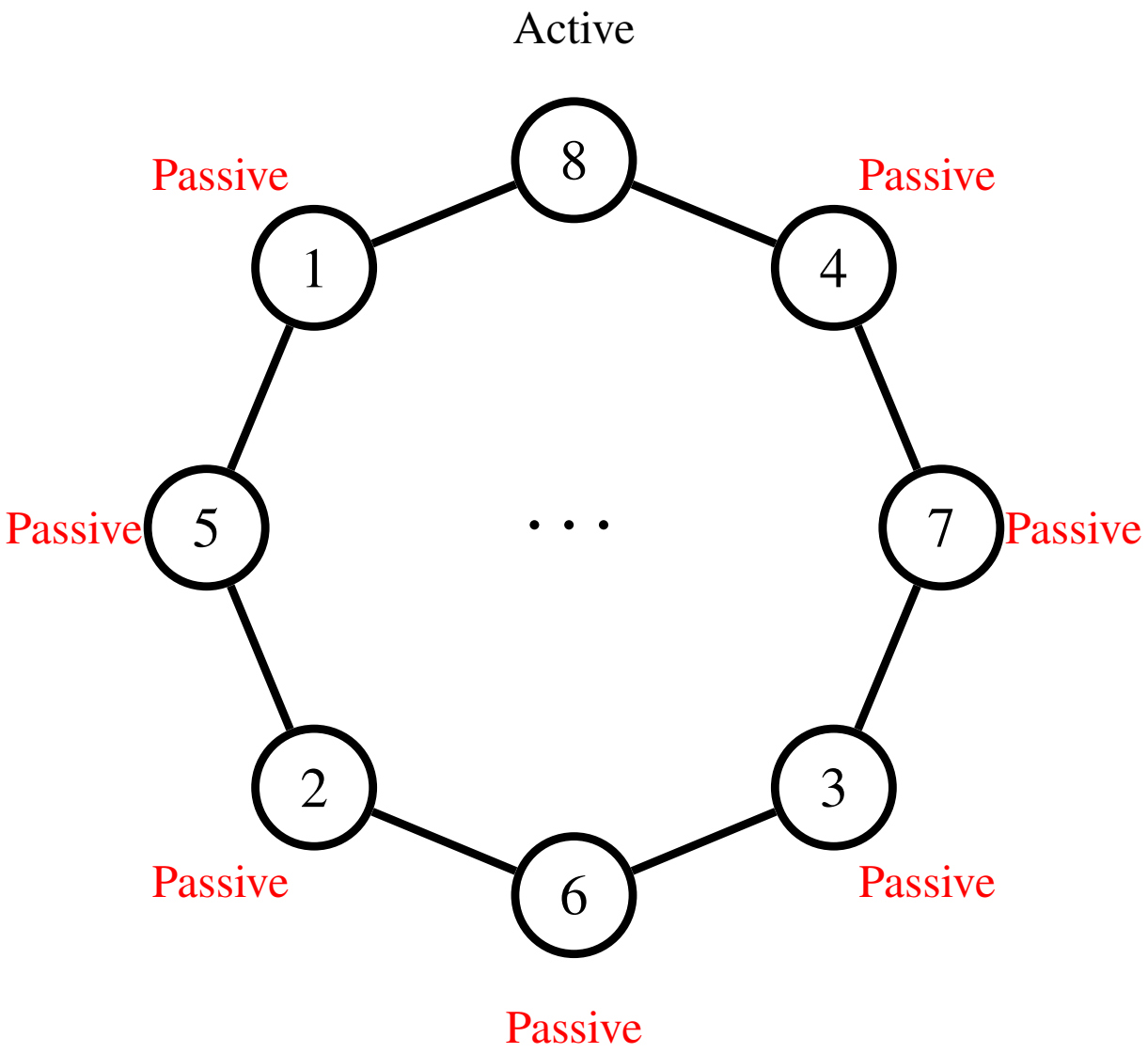
Example


Active
8
Passive
Passive
1
4
Probe
id:8
Leader
TTL:8
Probe
id:8
Leader
TTL:8
Passive
5
7
Passive
2
3
Passive
Passive
6
Passive

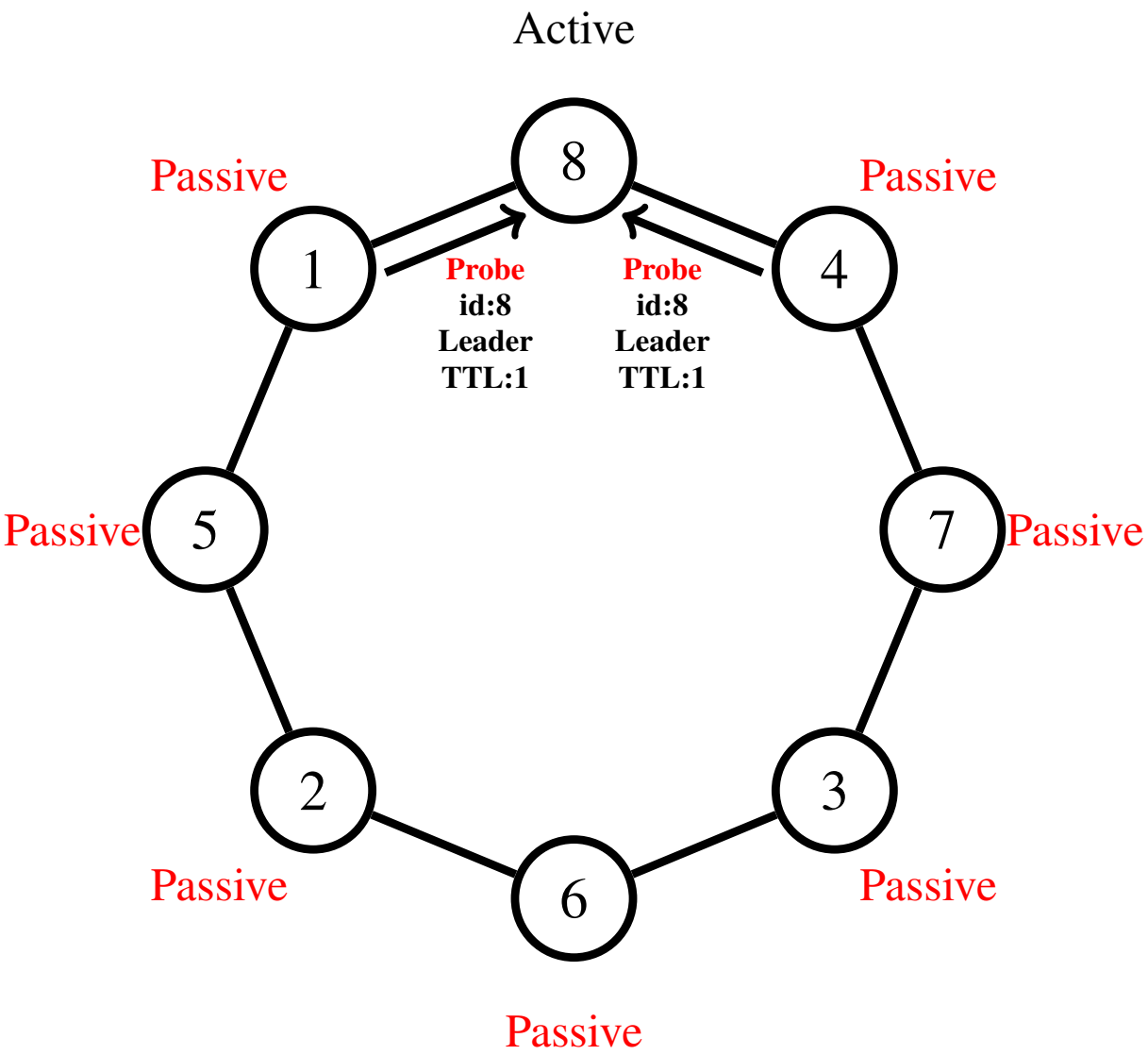
Example


Active
8
Passive
Passive
1
4
Probe
id:8
Leader
TTL:7
Probe
id:8
Leader
TTL:7
Passive
5
7
Passive
2
3
Passive
Passive
6
Passive

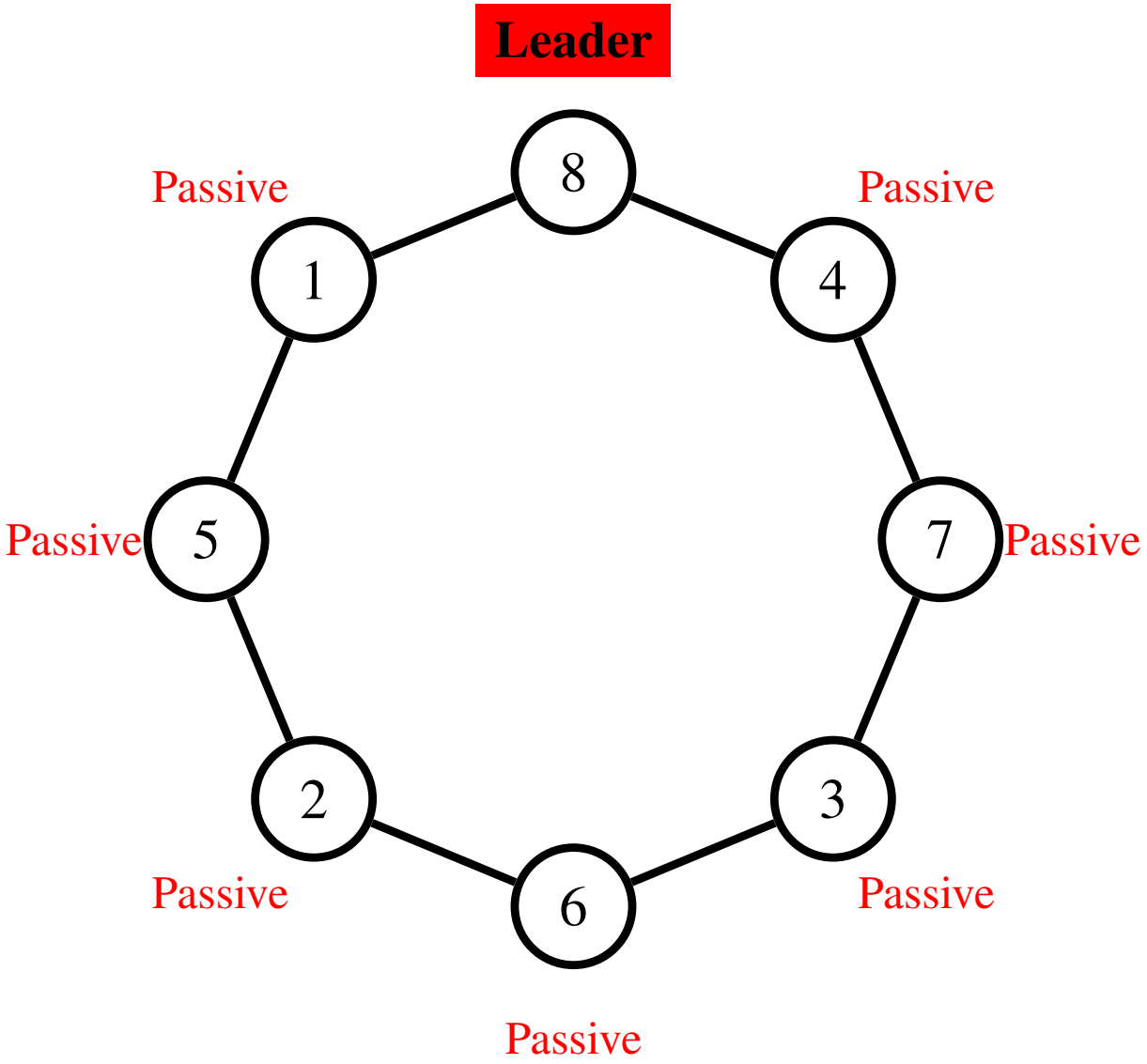
Example


Active
8
Passive
Passive
1
4
Passive
5
7
Passive
. . .
2
3
Passive
Passive
6
Passive

Example


Active
8
Passive
Passive
1
4
Probe
id:8
Leader
TTL:1
Probe
id:8
Leader
TTL:1
Passive
5
7
Passive
2
3
Passive
Passive
6
Passive

Example


Leader
8
Passive
Passive
1
4
Passive
5
7
Passive
2
3
Passive
Passive
6
Passive

Clockwise Leader Election

ALGORITHM 3.6: CLOCKWISE LEADER ELECTION()

1: **Each node** v executes the following code:
2: v sends a message with its identifier (for simplicity also v) to its clockwise neighbor.
3: v sets $m := v$ {the largest identifier seen so far }
4: **if** v receives a message w with $w > m$ **then**
5: v forwards message w to its clockwise neighbor and sets $m := w$.
6: v decides not to be the leader, if it has not done so already.
7: **if** v receives its own identifier v **then**
8: v decides to be the leader.

Theorem

Algorithm 3.6 *is correct. The time complexity is* (n). *The message complexity is* (n^2).

Proof of Theorem 3.7

Proof.

Let node z be the node with the maximum identifier. Node z sends its identifier in clockwise direction, and since no other node can swallow it, eventually a message will arrive at z containing it. Then z declares itself to be the leader. Every other node will declare non-leader at the latest when forwarding message z. Since there are n identifiers in the system, each node will at most forward n messages, giving a message complexity of at most n^2. We start measuring the time when the first node that "wakes up" sends its identifier. For asynchronous time complexity (Definition 2.8) we assume that each message takes at most one time unit to arrive at its destination. After at most $n-1$ time units the message therefore arrives at node z, waking z up. Routing the message z around the ring takes at most n time units. Therefore node z decides no later than at time $2n-1$. Every other node decides before node z. □

Remarks

- Note that in Algorithm 3.6 nodes distinguish between clockwise and counterclockwise neighbors. This is not necessary: It is okay to simply send your own identifier to any neighbor, and forward a message to the neighbor you did not receive the message from. So nodes only need to be able to distinguish their two neighbors.
- Careful analysis shows, that while having worst-case message complexity of $O(n^2)$, Algorithm 3.6 has an average message complexity of $O(n \log n)$. Can we improve this algorithm?

Radius Growth

ALGORITHM 3.8: RADIUS GROWTH()

1: **Each node** does the following:
2: Initially all nodes are active. all nodes may still become leaders
3: Whenever a node v sees a message w with $w > v$, then v decides to not be a leader and becomes passive.
4: Active nodes search in an exponentially growing neighborhood (clockwise and counterclockwise) for nodes with higher identifiers, by sending out probe messages. A probe message includes the ID of the original sender, a bit whether the sender can still become a leader, and a time-to-live number (TTL). The first probe message sent by node v includes a TTL of 1.
5: Nodes (active or passive) receiving a probe message decrement the TTL and forward the message to the next neighbor; if their ID is larger than the one in the message, they set the leader bit to zero, as the probing node does not have the maximum ID. If the TTL is zero, probe messages are returned to the sender using a

Radius Growth

Theorem

Algorithm 3.8 is correct. The time complexity is $O(n)$. The message complexity is $O(n \log n)$.

Proof.

Correctness is as in Theorem 3.7. The time complexity is $\mathcal{O}(n)$ since the node with maximum identifier z sends messages with round-trip times $2, 4, 8, 16, \cdots, 2 \cdot 2k$ with $k \cdot \log(n+1)$. (Even if we include the additional wake-up overhead, the time complexity stays linear.) Proving the message complexity is slightly harder: if a node v manages to survive round r, no other node in distance $2r$ (or less) survives round r. That is, node v is the only node in its $2r$-neighborhood that remains active in round $r+1$. Since this is the same for every node, less than $n = 2r$ nodes are active in round $r+1$. Being active in round r costs $2 \cdot 2 \cdot 2r$ messages. Therefore, round r costs at most $2 \cdot 2 \cdot 2^r \cdot \frac{n}{2^{r-1}} = 8$ nmessages. Since there are only logarithmic many possible rounds, the message complexity follows immediately. □

Radius Growth (cont'd)

- This algorithm is asynchronous and uniform as well.
- The question may arise whether one can design an algorithm with an even lower message complexity.

Synchronous Ring

- Basic idea:
  - In the synchronous model, not receiving a message is information as well.
- Assumptions:
  - Non-uniform algorithm.
  - Every node starts at the same time.
  - The node with the minimum identifier becomes the leader.
  - Identifiers are integers.

Synchronous Ring Leader Election

ALGORITHM 3.17: SYNCHRONOUS LEADER ELECTION()

1: **Each node** *v* concurrently executes the following code:
2: The algorithm operates in synchronous phases. Each phase consists of n time steps. Node v counts phases, starting with 0.
3: **if** phase=*v* **and** *v* did not yet receive a message **then**
4: v decides to be the leader
5: v sends the message ǐs leader" around the ring

Analysis

- Message complexity is indeed n, the number of nodes.
- The time complexity is huge.
  - If m is the minimum identifier, the time complexity is $m \cdot n$.
  - Why?
- The synchronous start and the non-uniformity assumptions can be dropped.

Remarks

- There are several lower bounds for the synchronous model: comparison based algorithms or algorithms where the time complexity cannot be a function of the identifiers have message complexity $(n \log n)$ as well.
- In general graphs, efficient leader election may be tricky. While time-optimal leader election can be done by parallel flooding-echo (see Chapter 2), bounding the message complexity is more difficult.

What is the message lower bound for Leader-election in Asynchronous Ring?

Lower Bounds for Leader Election

Definition (Execution)

An execution of a distributed algorithm is a list of events, sorted by time. An event is a record (time, node, type, message), where type is ”send” or ”receive”.

- Some assumptions:
  - No two events happen at exactly the same time.
  - If more than one message is in transit, we can choose which one arrives first.
  - If two messages are transmitted over the same directed edge, the message first transmitted will also be received first (”FIFO”).
  - Nodes may wake up at arbitrary times (but at the latest when receiving the first message)
  - Uniform algorithms (can be dropped).
  - Every node that is not the leader must know the identity of the leader (can be dropped).

Open Schedule

Definition (Open Schedule)

A schedule is an execution chosen by the scheduler. An open (un-directed) edge is an edge where no message traversing the edge has been received so far. A schedule for a ring is open if there is an open edge in the ring.

Induction

Lemma

Given a ring R with two nodes, we can construct an open schedule in which at least one message is received. The nodes cannot distinguish this schedule from one on a larger ring with all other nodes being where the open edge is.

Proof.

Let the two nodes be u and v with $u < v$. Node u must learn the identity of node v, thus receive at least one message. We stop the execution of the algorithm as soon as the first message is received. (If the first message is received by v, bad luck for the algorithm!) Then the other edge in the ring (on which the received message was not transmitted) is open. Since the algorithm needs to be uniform, maybe the open edge is not really an edge at all, nobody can tell. We could use this to glue two rings together, by breaking up this imaginary open edge and connect two rings by two edges. An example can be

Induction

Lemma

By gluing together two rings of size $n/2$ for which we have open schedules, we can construct an open schedule on a ring of size n. If $M(n/2)$ denotes the number of messages already received in each of these schedules, at least $2M(n/2)+n/4$ messages have to be exchanged in order to solve leader election.

Lower Bound

Lemma

Any uniform leader election algorithm for asynchronous rings has at least message complexity $M(n) \geq n4(\log n + 1)$.

Proof.

By induction: For the sake of simplicity, we assume n being a power of 2. The base case $n = 2$ works because of Lemma 3.12 which implies that $M(2) \geq 1 = \frac{2}{4}(\log 2 + 1)$. For the induction step, using Lemma 3.14 and the induction hypothesis, we have

$$M(n) = 2 \cdot M(\frac{n}{2}) + \frac{n}{4}$$

$$\geq 2 \cdot \left(\frac{n}{8} \left(\log \frac{n}{2} + 1 \right) \right) + \frac{n}{4}$$

$$= n \frac{}{4 \log n + \frac{n}{4} = \frac{n}{4}(\log n + 1)}$$

Leader Election for General Networks

- This technique in essence is similar to finding a Minimum Spanning Tree (MST)
  - the root of the tree becomes the leader.
- The basic idea in this method is
  - individual nodes merge with each other to form bigger structures.
  - The result of this algorithm is a tree (a graph with no cycle) whose root is the leader of entire system.
  - The cost of mega-merger method is $O(m + n \log n)$ where m is the number of edges and n is the number of nodes.