CS 60002: Distributed Systems

T9:
Leader Election

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

Leader Election in Distributed System

- Each process eventually decides whether it is the leader or not, subject to the constraint that there is **exactly one leader**
  - Processes will be in one of the three states – **undecided, leader, not leader**
  - Initial state: **undecided**
  - Final state: **leader** or **not leader**

Leader Election in Distributed System

- Each process eventually decides whether it is the leader or not, subject to the constraint that there is **exactly one leader**
  - Processes will be in one of the three states – **undecided, leader, not leader**
  - Initial state: **undecided**
  - Final state: **leader** or **not leader**
- We have already seen in RAFT and PBFT ! -- the role of a leader
  - Central server for mutual exclusion
  - Message ordering
  - Ensure serializability
  - Ensure consensus
  - Take snapshot
  - …

Leader Election in Distributed System

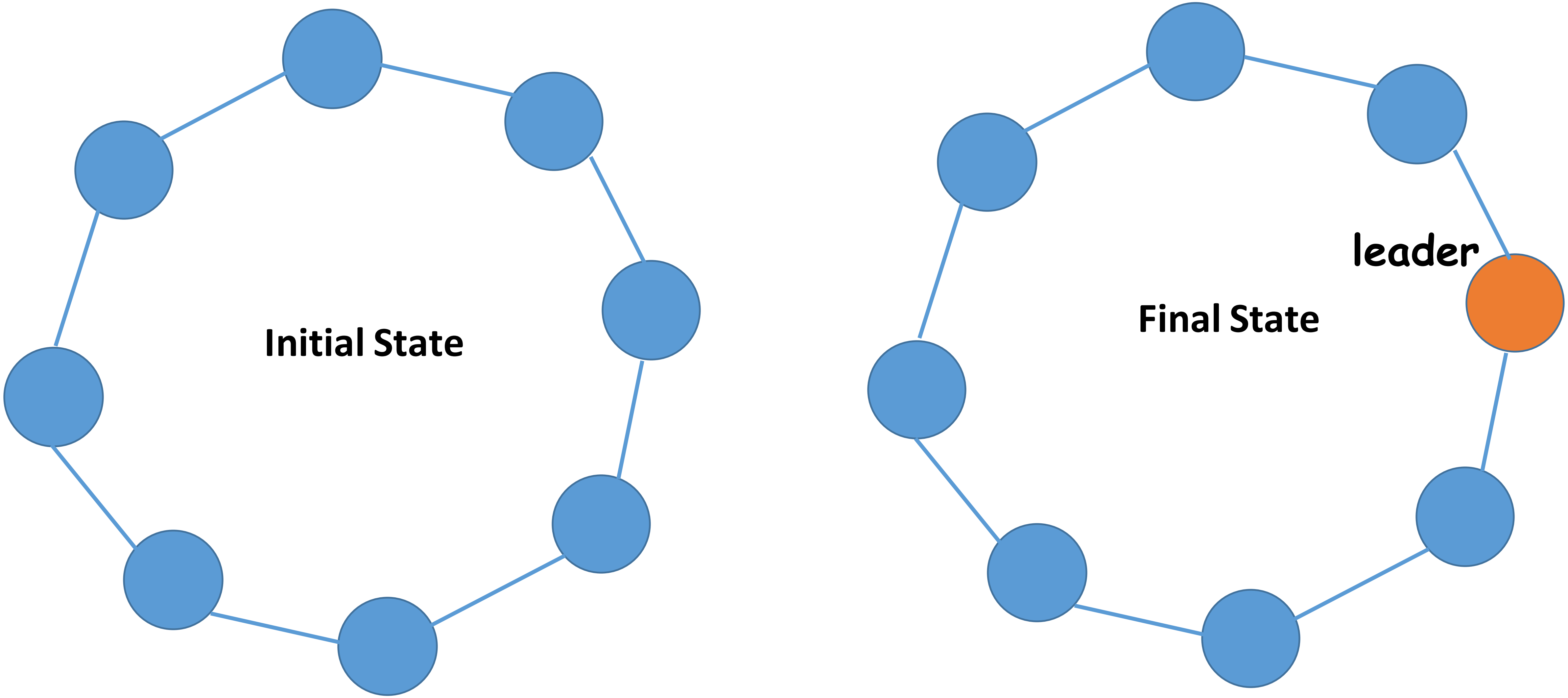
- Requirements:
  - The protocol should eventually terminate
  - In each round, exactly one process will be elected as the leader
  - On termination, the leader process should know that it is the leader
  - All other processes know that they are not the leader, and (optionally) knows who the leader is

- Distributed leader election has been studied in different topologies
  - Rings
  - Arbitrary topology

Leader Election in Rings

- System models
  - Synchronous or Asynchronous
  - Unidirectional or Bidirectional ring
  - Anonymous (no unique ID) or Non-anonymous (no unique IDs)
  - Uniform (no knowledge on the number of processes) or Non-uniform (the information about the number of processes is known)

Leader Election in Rings



Why Do We Study Rings

- Simple starting point to understand leader election
  - Easy to analyze the algorithms
- The lower bounds and impossibility results derived for ring topologies also apply to arbitrary topologies as well
  - If you cannot do it over a ring, you cannot do over an arbitrary topology

Leader Election in Rings

- System models
  - Synchronous or Asynchronous
  - Unidirectional or Bidirectional ring
  - Anonymous (no unique ID) or Non-anonymous (no unique IDs)
  - Uniform (no knowledge on the number of processes) or Non-uniform (the information about the number of processes is known)

- **Impossibility result:**
  - There is no deterministic leader election protocol for anonymous rings even if
    - The protocol knows the ring size (non-uniform)
    - The channel is synchronous

Impossibility Proof

- **Deterministic leader election in an anonymous ring is impossible**
  - Processes do not have unique identifiers – there is no way to distinguish the processes from each other
  - Every processor starts in the same state (**undecided**) with the same outgoing messages (as they are anonymous)
  - **Every processor runs the same algorithm** -- Everyone does the same computation, sends and receives same messages, so end up in same states
  - If one node decides to become the leader, then every other nodes does so

Impossibility Proof

- **Deterministic leader election in an anonymous ring is impossible**
  - Processes do not have unique identifiers – there is no way to distinguish the processes from each other
  - Every processor starts in the same state (**undecided**) with the same outgoing messages (as they are anonymous)
  - **Every processor runs the same algorithm** -- Everyone does the same computation, sends and receives same messages, so end up in same states
  - If one node decides to become the leader, then every other nodes does so

- The same result holds for weaker models
  - Asynchronous
  - Uniform

Impossibility Proof

- However, **Randomized algorithms are possible (but not always easy!)**
  - Galindo, David, et al. "Fully distributed verifiable random functions and their application to decentralised random beacons." 2021 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2021.

Rings with Identifiers

- **Identifiers** (IDs)
  - Arbitrary non-negative integers
  - Available to the processes

- Every process has a unique ID
  - We typically use 0 to n-1 as the indices of the processes in the rings – these are not identifiers, we typically use them for analysis
  - Identifiers can be anything, like 12643 – some arbitrary positive integers but unique to individual processes

Leader Election in Rings with IDs -- Overview

- **Best Results:**
  - Asynchronous Rings: $\Theta(n \log n)$ messages
  - Synchronous Rings: $\Theta(n)$ messages

Leader Election in Rings with IDs -- Overview

- **Best Results:**
  - Asynchronous Rings: $\Theta(n \log n)$ messages
  - Synchronous Rings: $\Theta(n)$ messages

- If all processes know the highest ID (say, k), then we do not need a leader election
  - Everyone considers the process with ID k as the leader
  - The process with ID k can start operating as the leader

Leader Election in Rings with IDs -- Overview

- **Best Results:**
  - Asynchronous Rings: $\Theta(n \log n)$ messages
  - Synchronous Rings: $\Theta(n)$ messages
- If all processes know the highest ID (say, k), then we do not need a leader election
  - Everyone considers the process with ID k as the leader
  - The process with ID k can start operating as the leader
- However, the process with ID k can fail
  - If we assume the next higher ID as the leader, that process can also fail

Leader Election in Rings with IDs -- Overview

- **Best Results:**
  - Asynchronous Rings: Θ(n log n) messages
  - Synchronous Rings: Θ(n) messages

- If all processes know the highest ID (say, *k*), then we do not need a leader election
  - Everyone considers the process with ID *k* as the leader
  - The process with ID *k* can start operating as the leader

- How
  - I

Broad Idea: The process with the highest ID and still surviving becomes the leader

Chang and Roberts Algorithm

- Assume that each process has to pointers to other processes that it knows about
  - The previous process (anti-clockwise)
  - The next process (clockwise)

- The actual network might not be a ring, but we assume that every process maintains the above information to form a logical ring overlay

- Chang and Roberts algorithm needs the next pointer only (Unidirectional Ring)

Chang, Ernest, and Rosemary Roberts. "An improved algorithm for decentralized extrema-finding in circular configurations of processes." *Communications of the ACM* 22.5 (1979): 281-283.

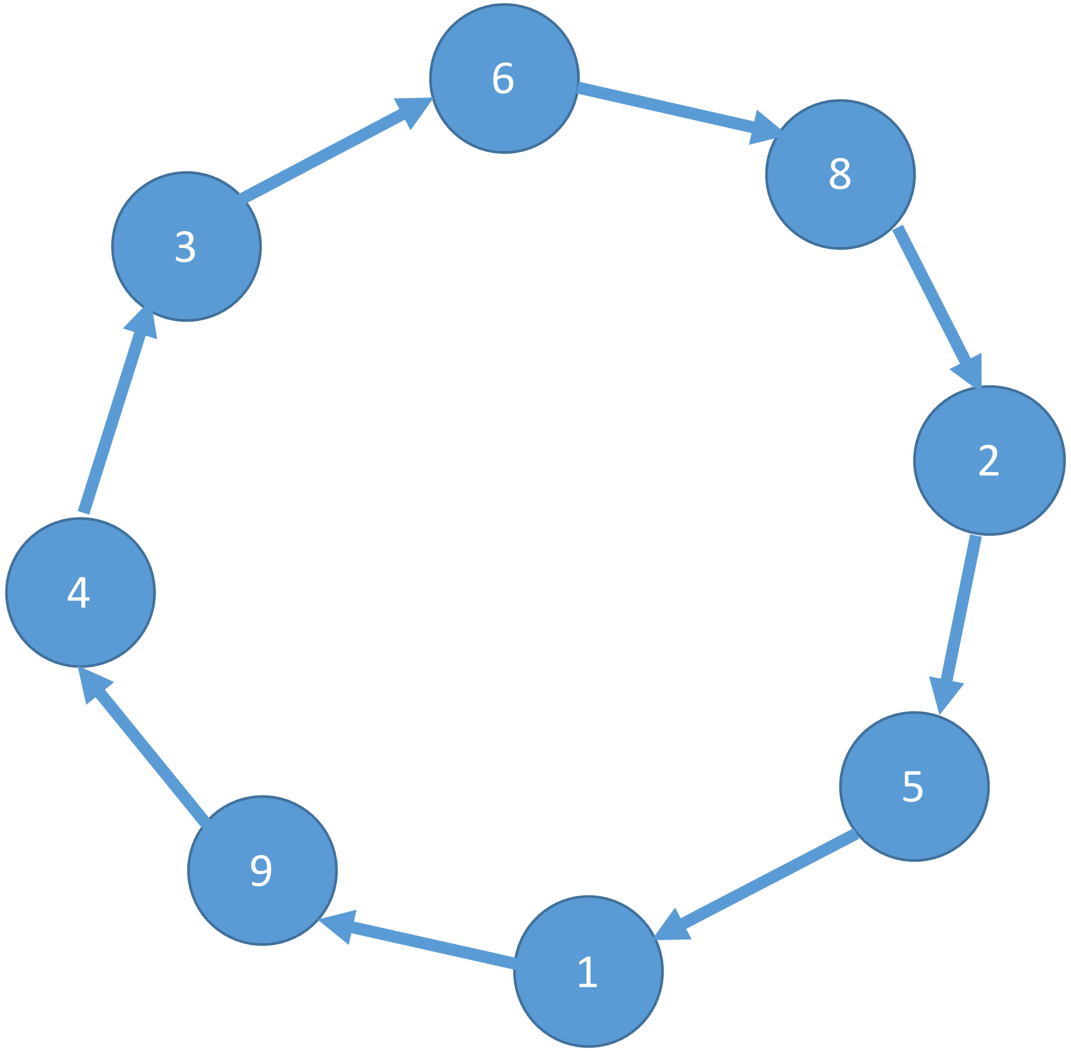
Chang and Roberts Algorithm

- **Uniform**: The algorithm does not need the information about the number of processes participating in the algorithm

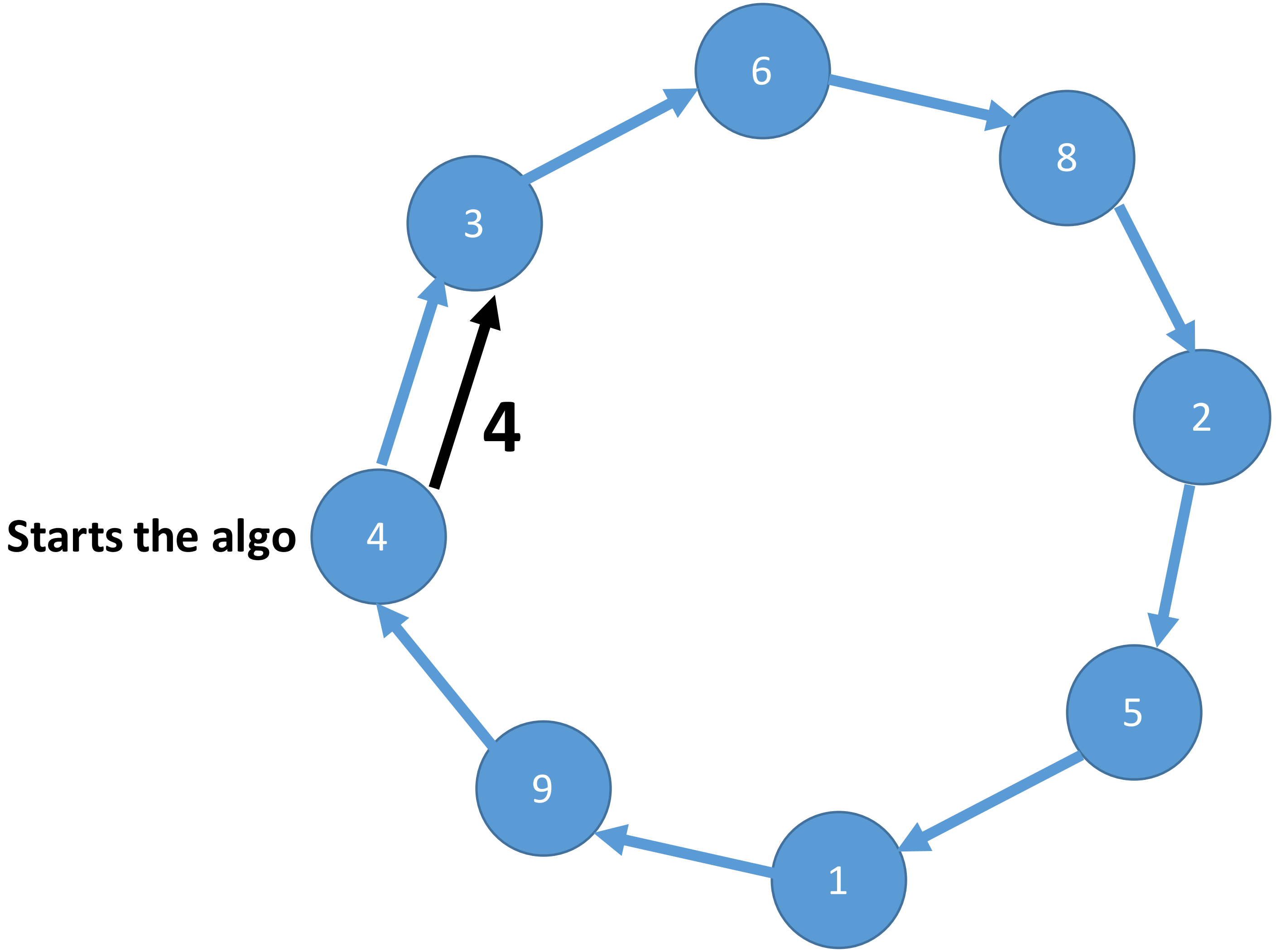
- Asynchronous but reliable channel

- **The algorithm:**
  - A process that observes lack of leader (random timeout), starts the election procedure
  - Every process send **max(own ID, received ID)** to the next process
  - If a process receives its own ID, then it becomes the leader
  - Leader passes a message across the ring announcing that it is the leader, all other processes mark them as non-leader

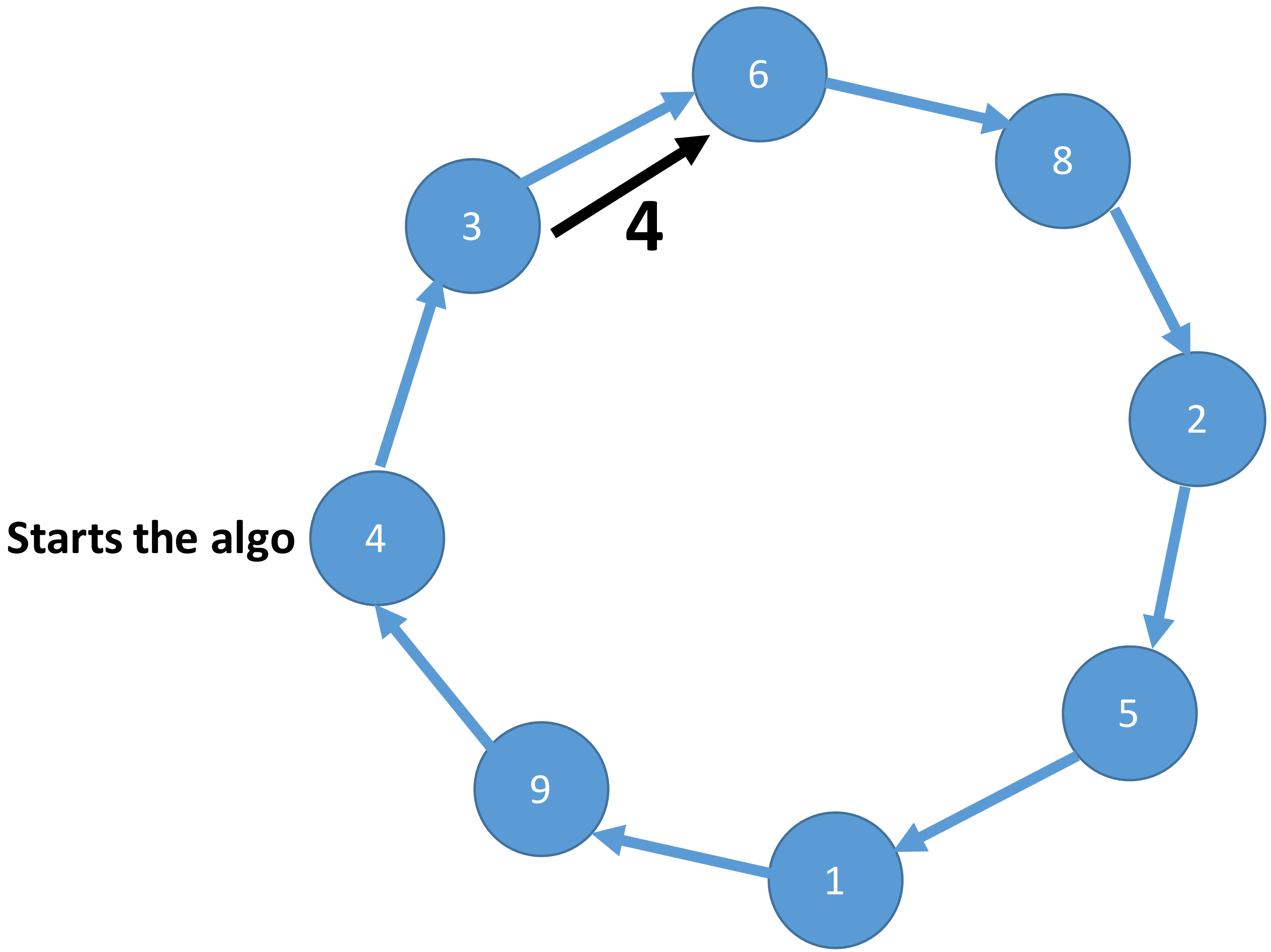
Chang and Roberts Algorithm



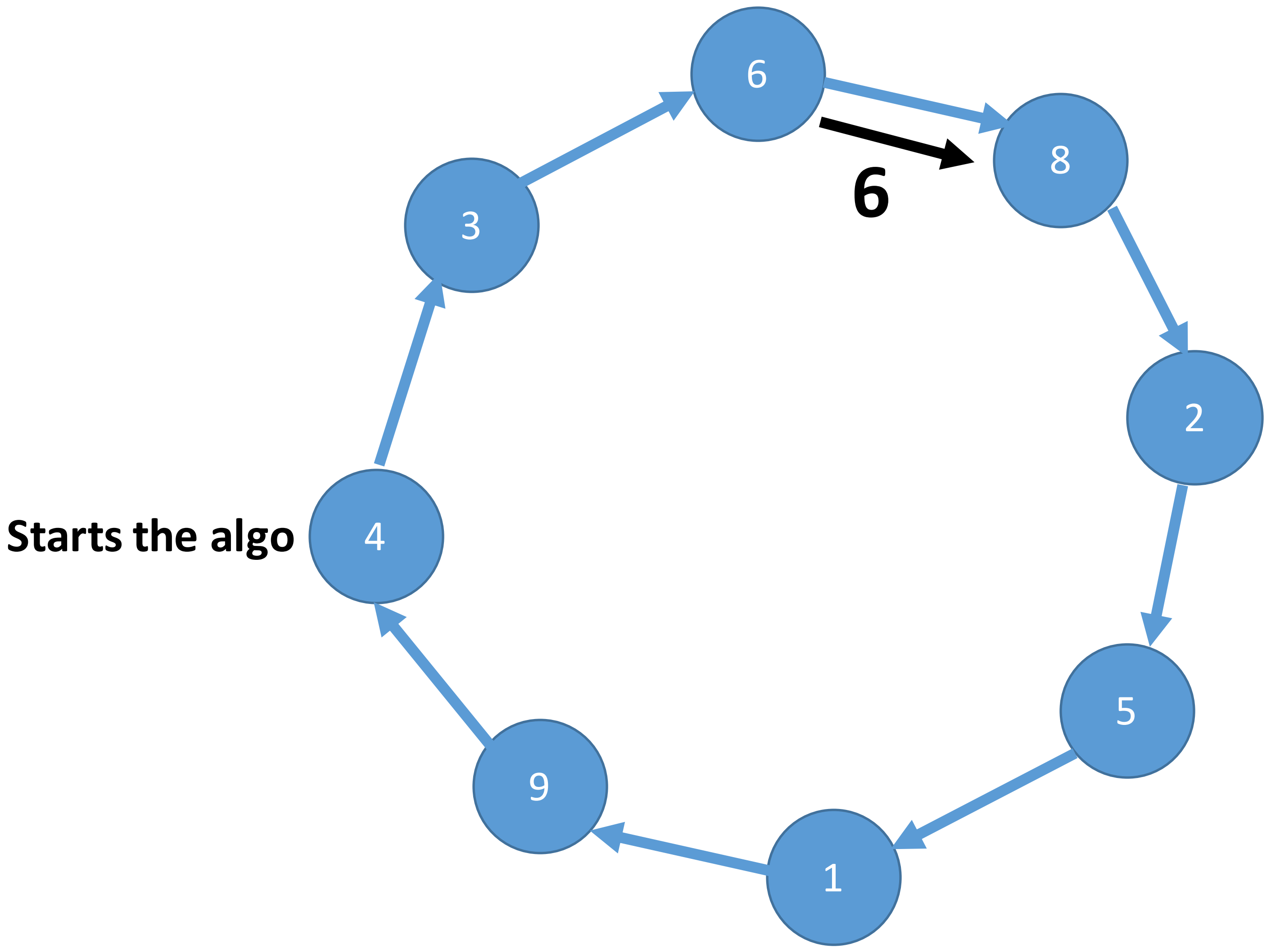
Chang and Roberts Algorithm



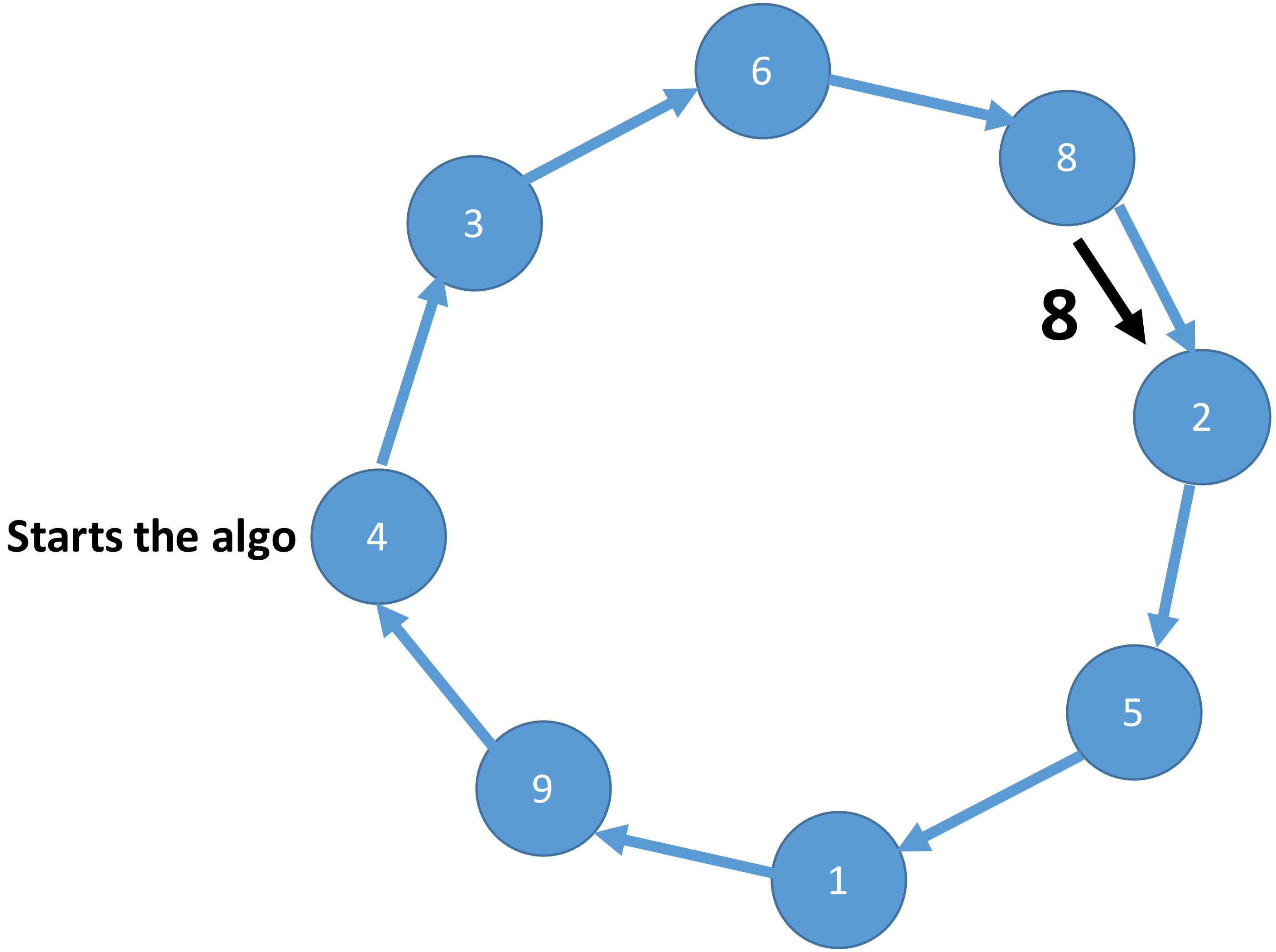
Chang and Roberts Algorithm



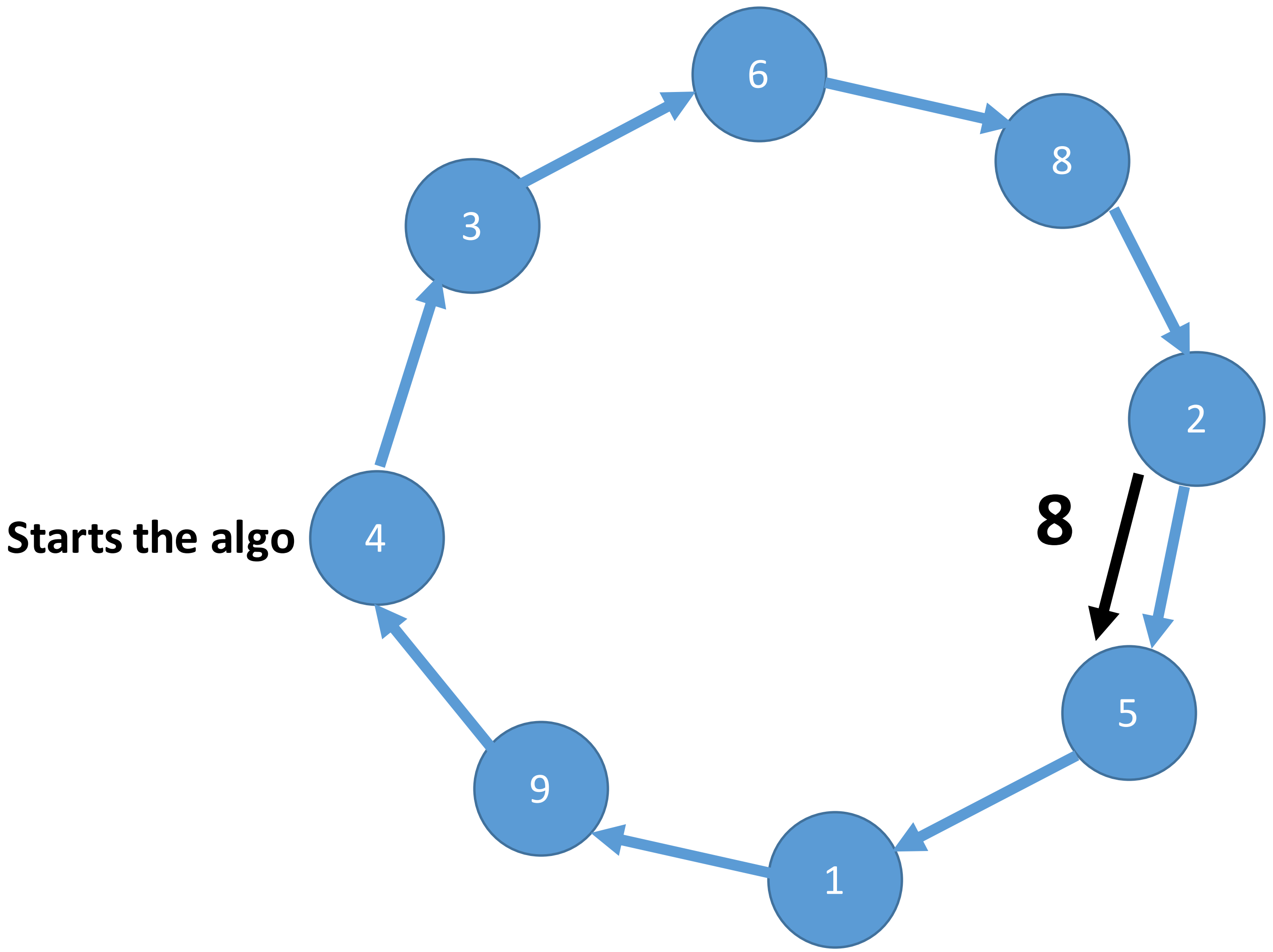
Chang and Roberts Algorithm



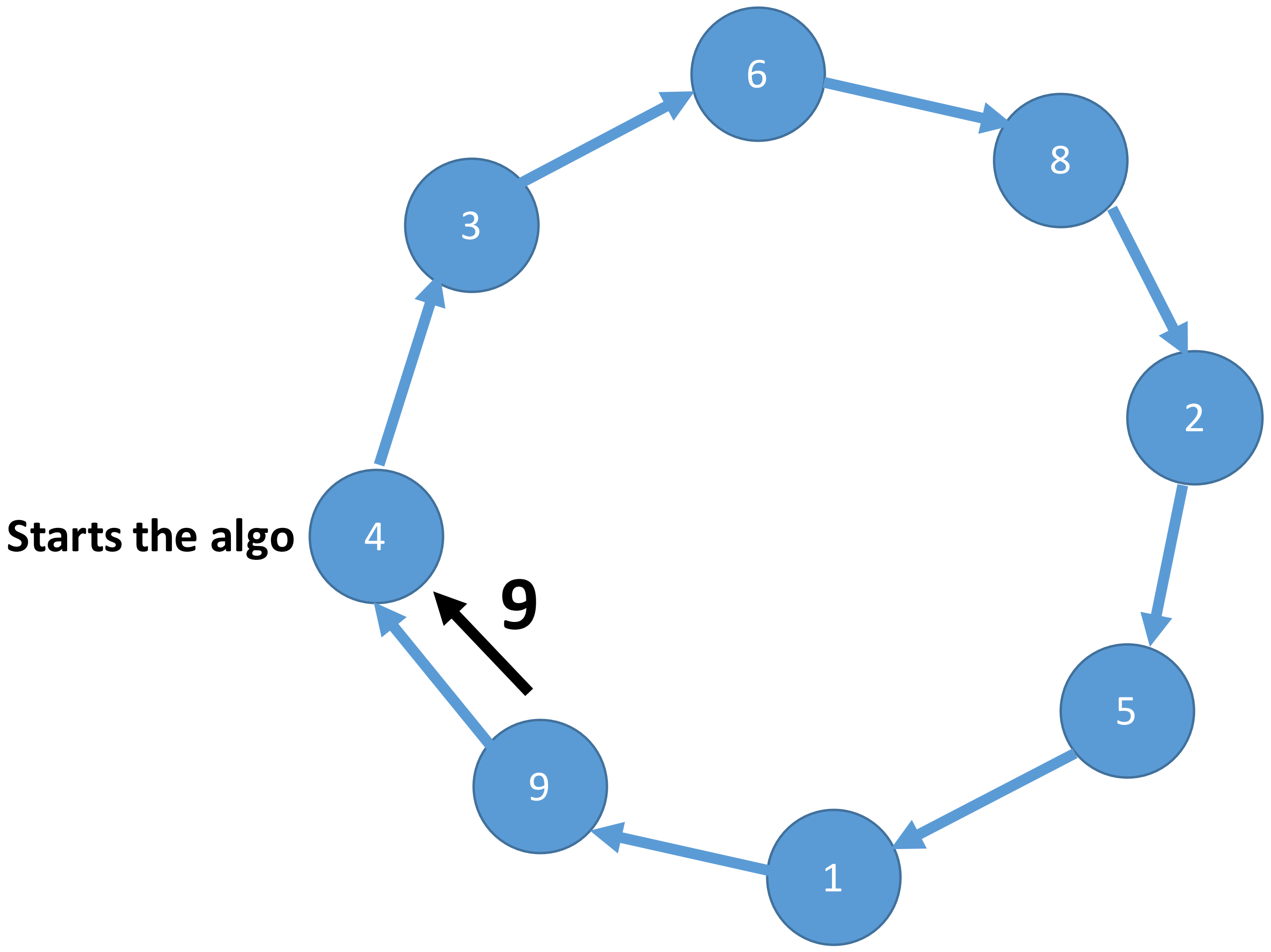
Chang and Roberts Algorithm



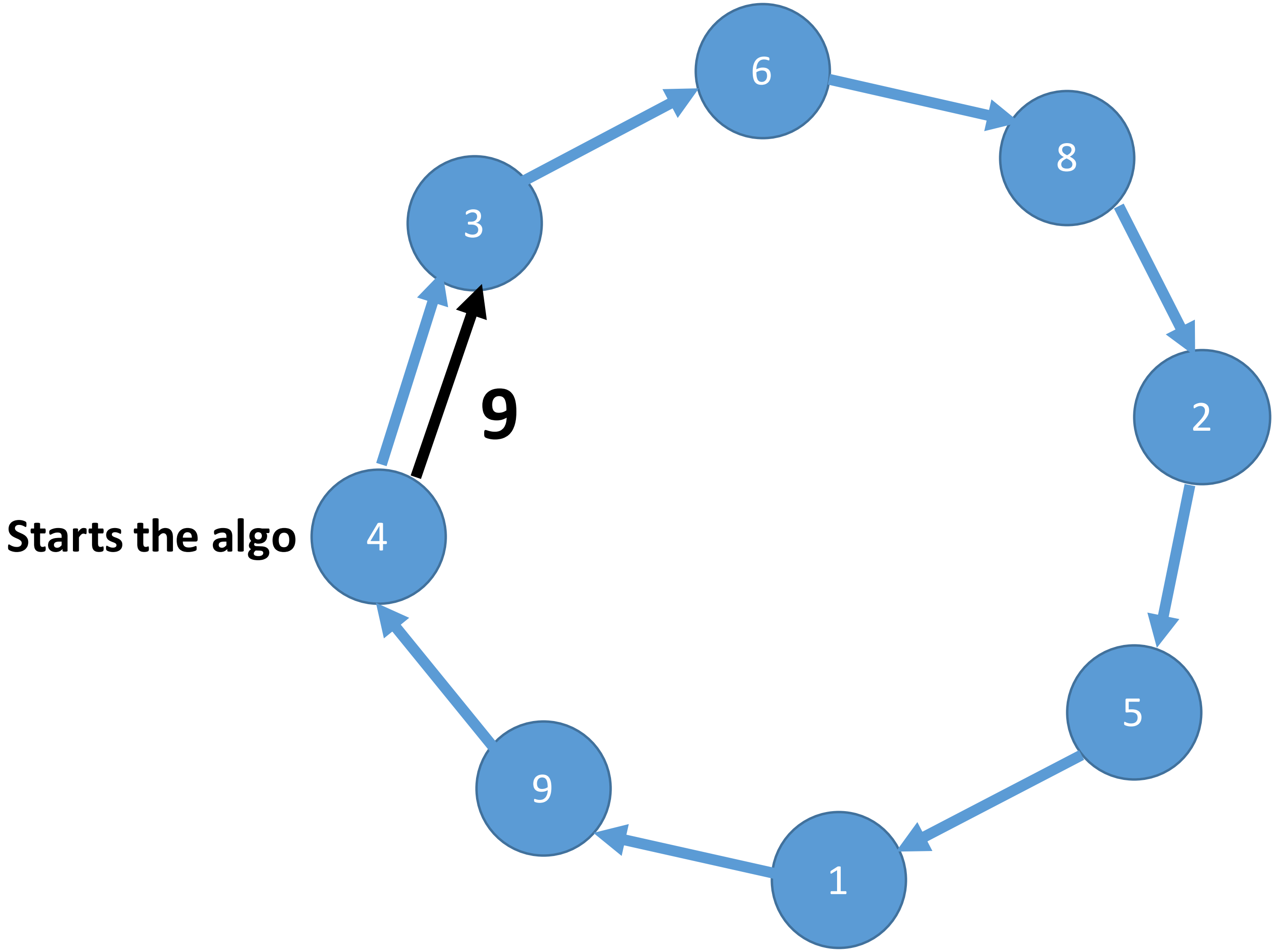
Chang and Roberts Algorithm



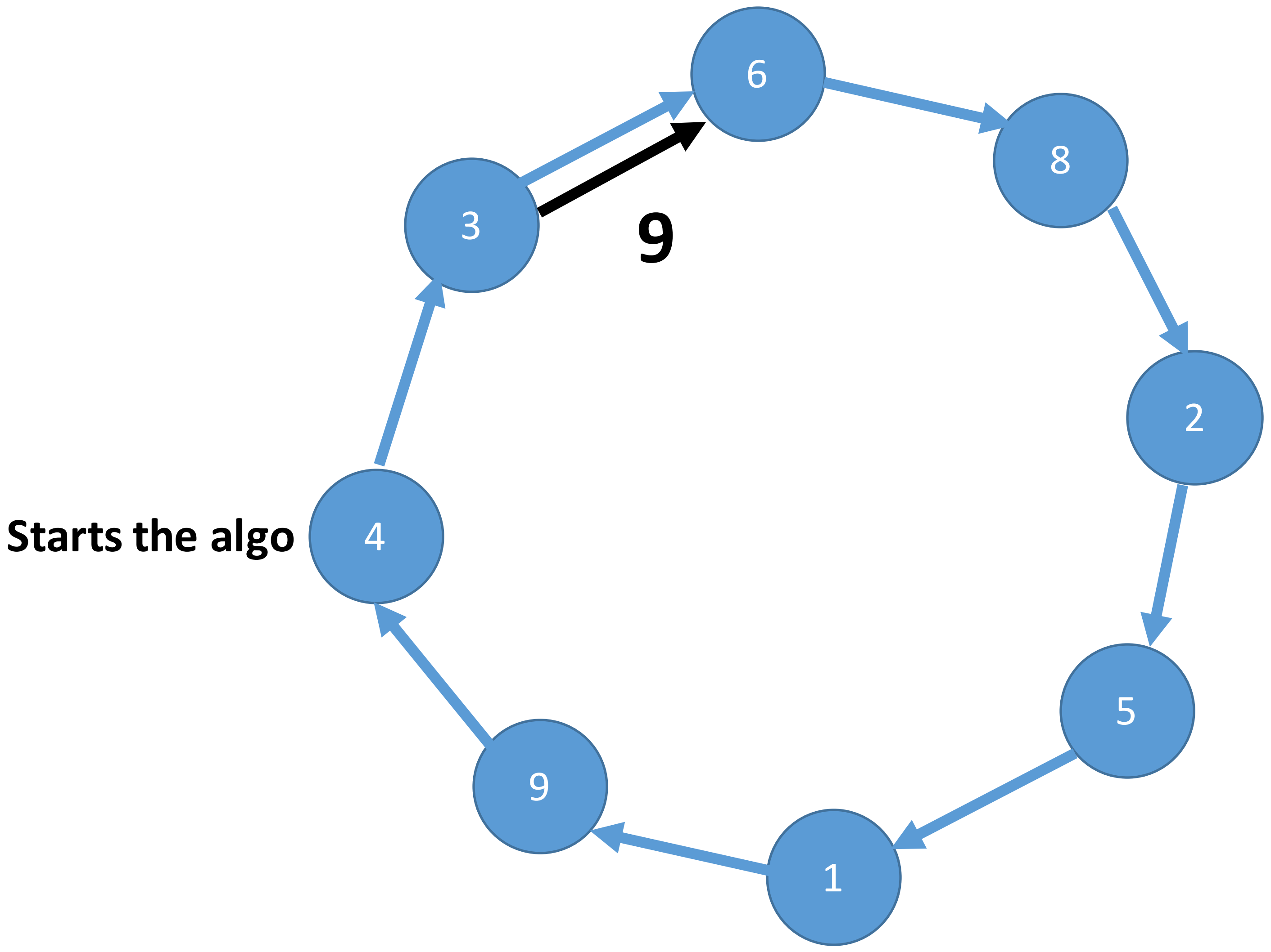
Chang and Roberts Algorithm



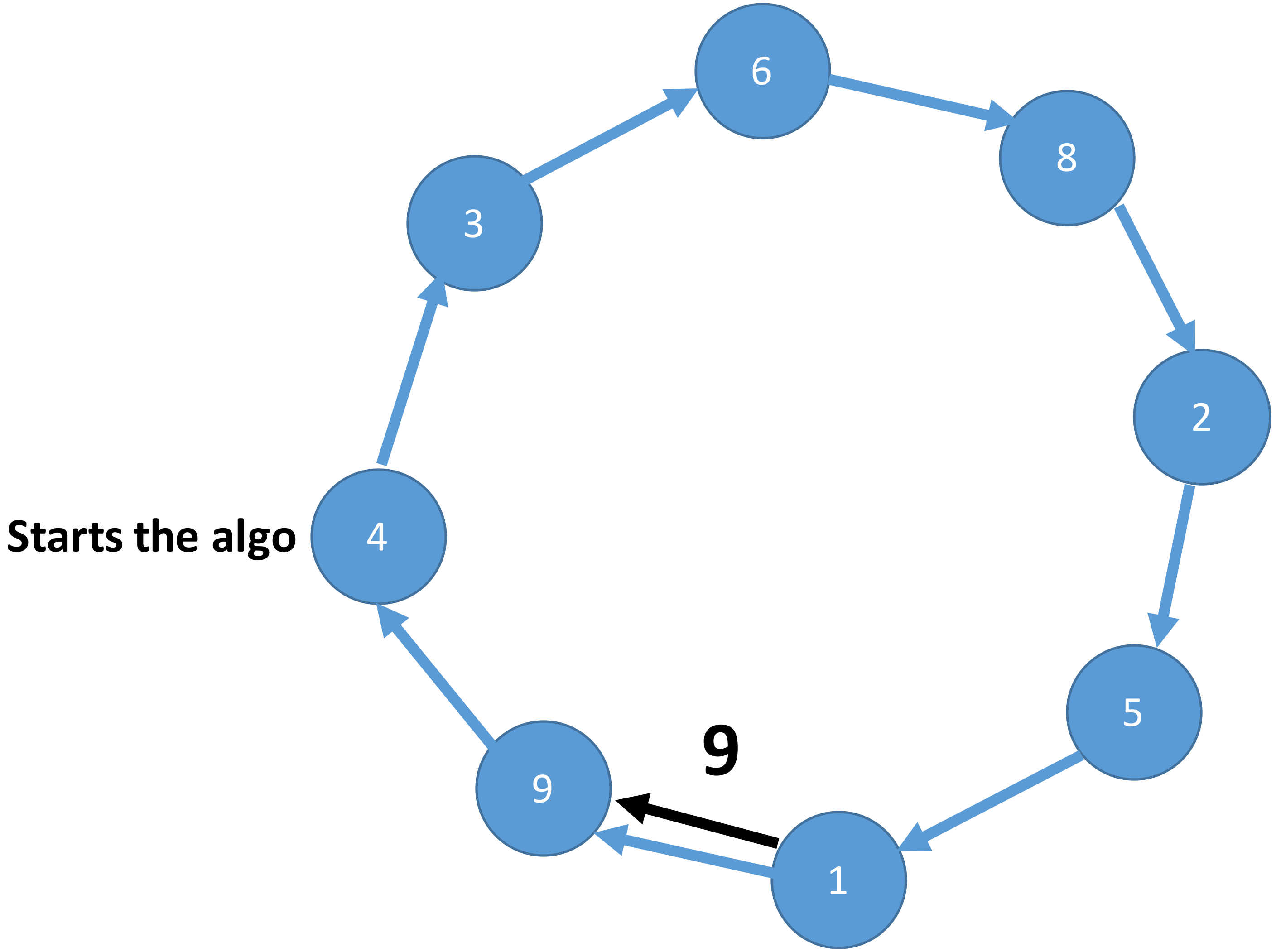
Chang and Roberts Algorithm



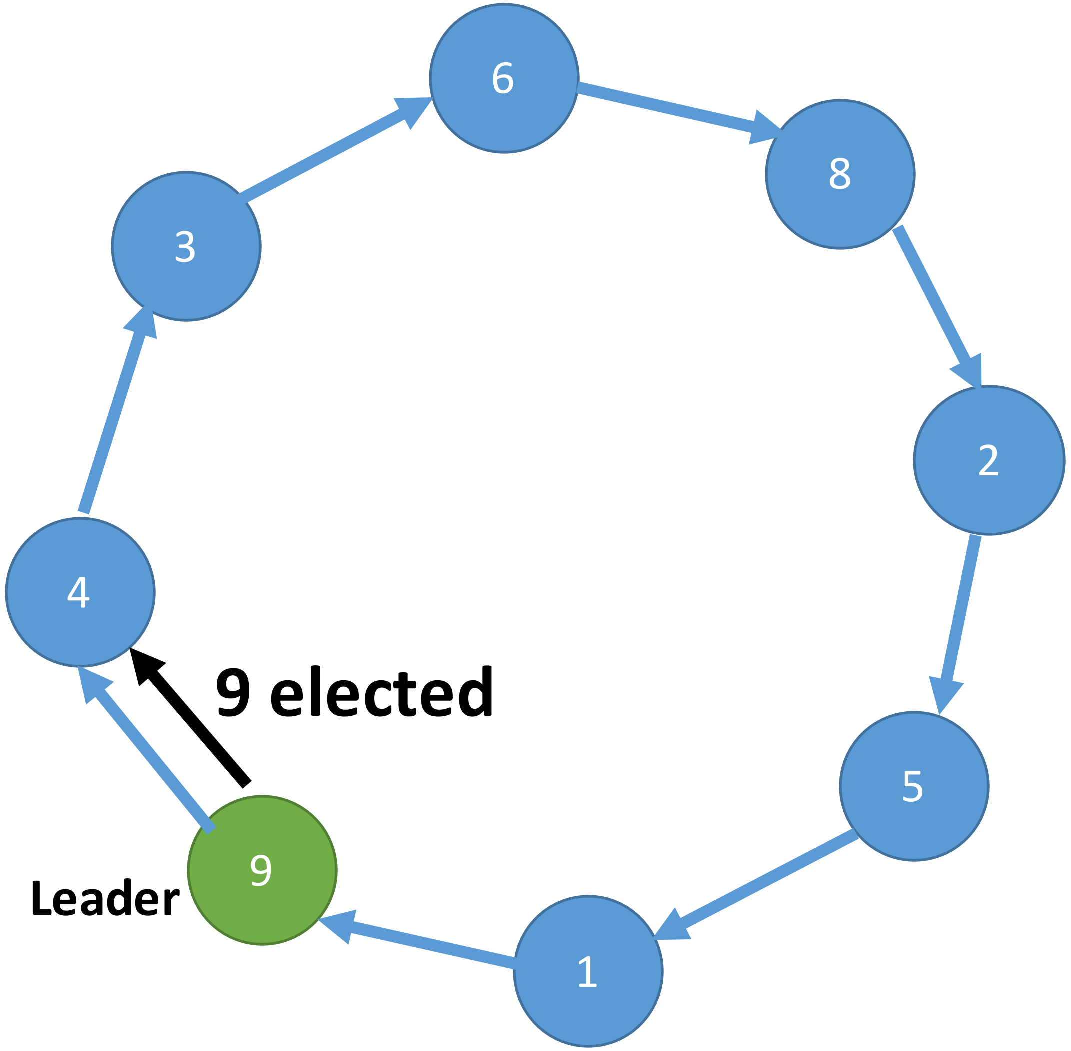
Chang and Roberts Algorithm



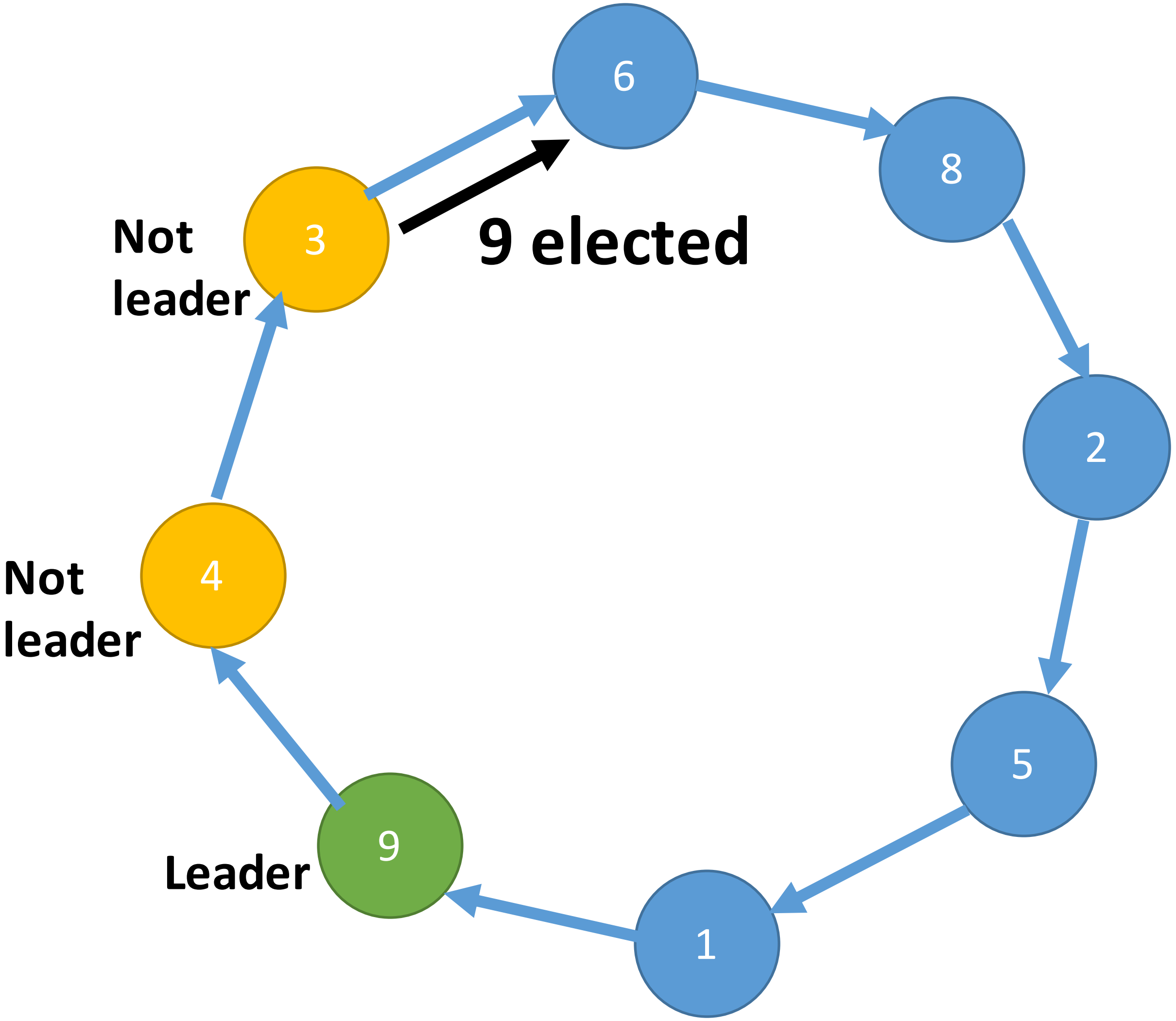
Chang and Roberts Algorithm



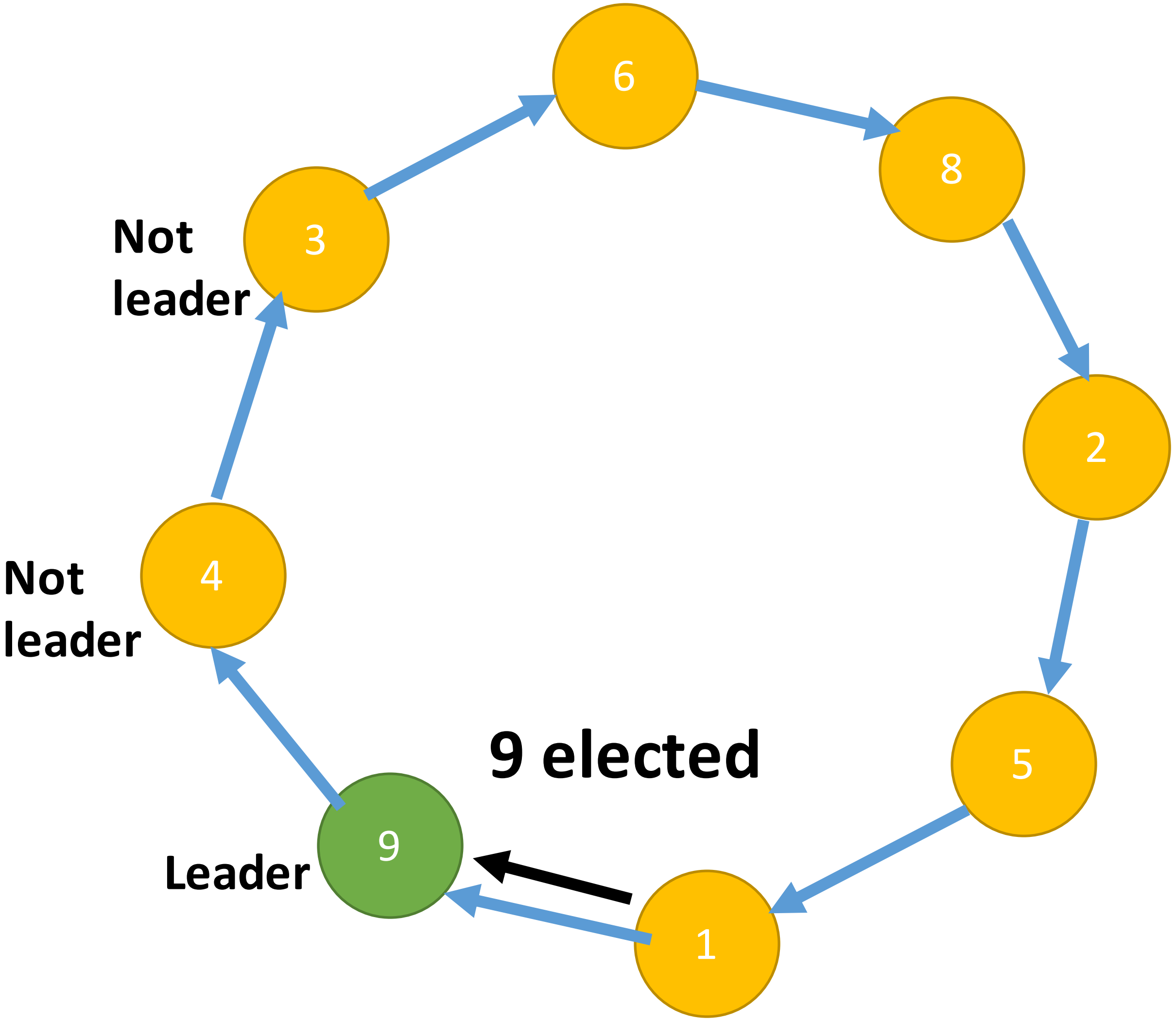
Chang and Roberts Algorithm



Chang and Roberts Algorithm



Chang and Roberts Algorithm



Chang and Roberts Algorithm

- **Correctness:** Process with largest ID is elected as the leader, that message passes through all other processes

Chang and Roberts Algorithm

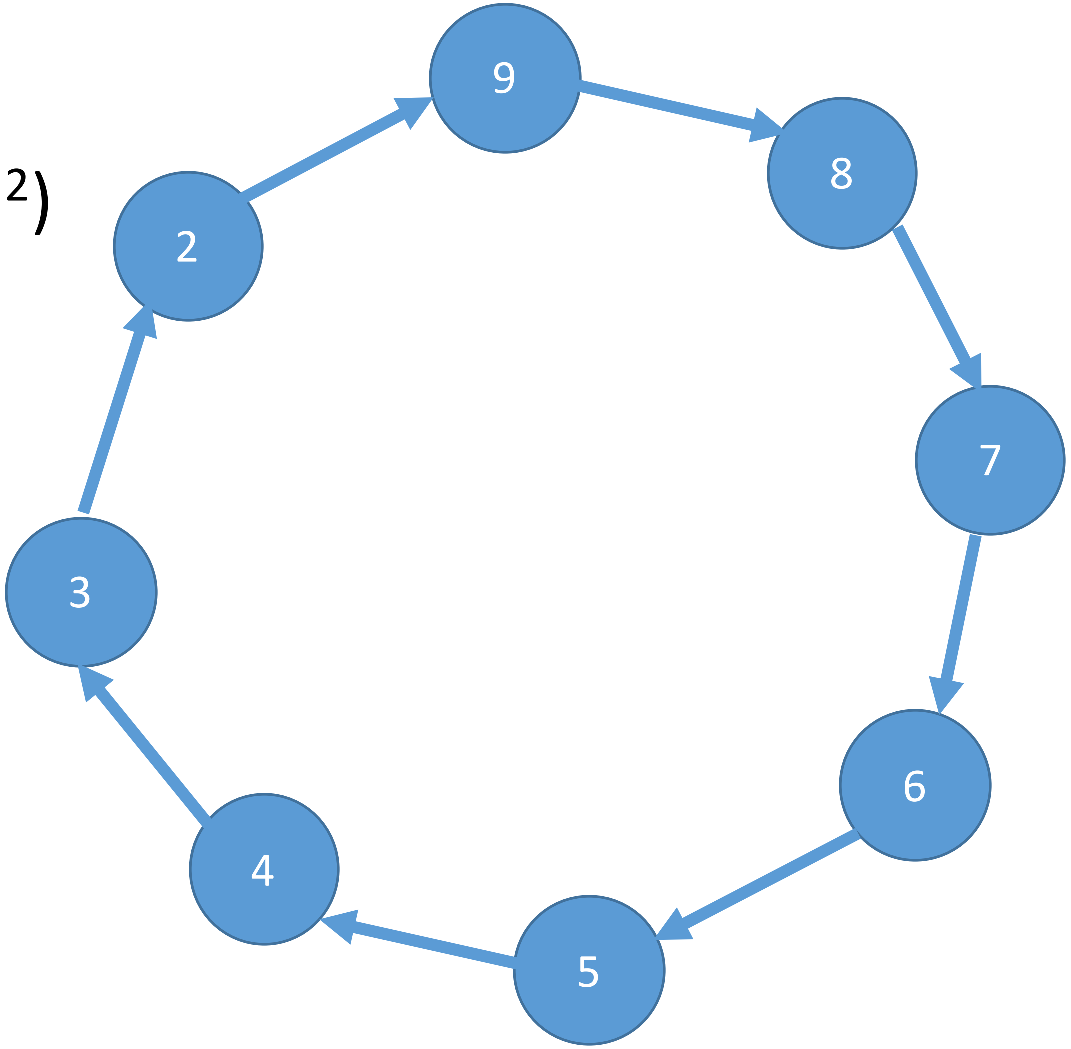
- **Correctness:** Process with largest ID is elected as the leader, that message passes through all other processes

- **Worst-case Message Complexity**: $O(n^2)$
  - **When does this occur?**

Chang and Roberts Algorithm

- **Correctness:** Process with largest ID is elected as the leader, that message passes through all other processes

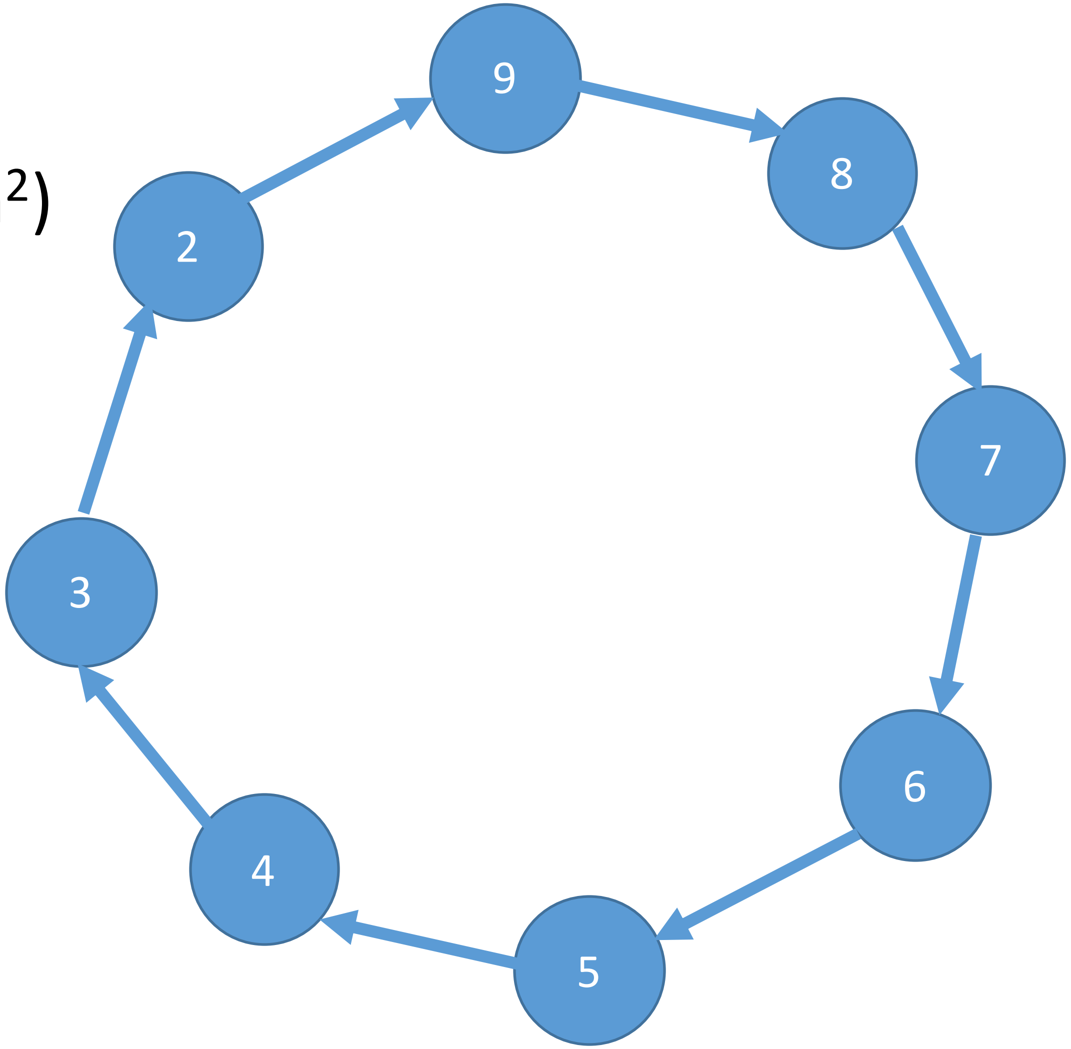
- **Worst-case Message Complexity**: $O(n^2)$
  - When does this occur?
  - Arrange the IDs in decreasing order



Chang and Roberts Algorithm

- **Correctness:** Process with largest ID is elected as the leader, that message passes through all other processes

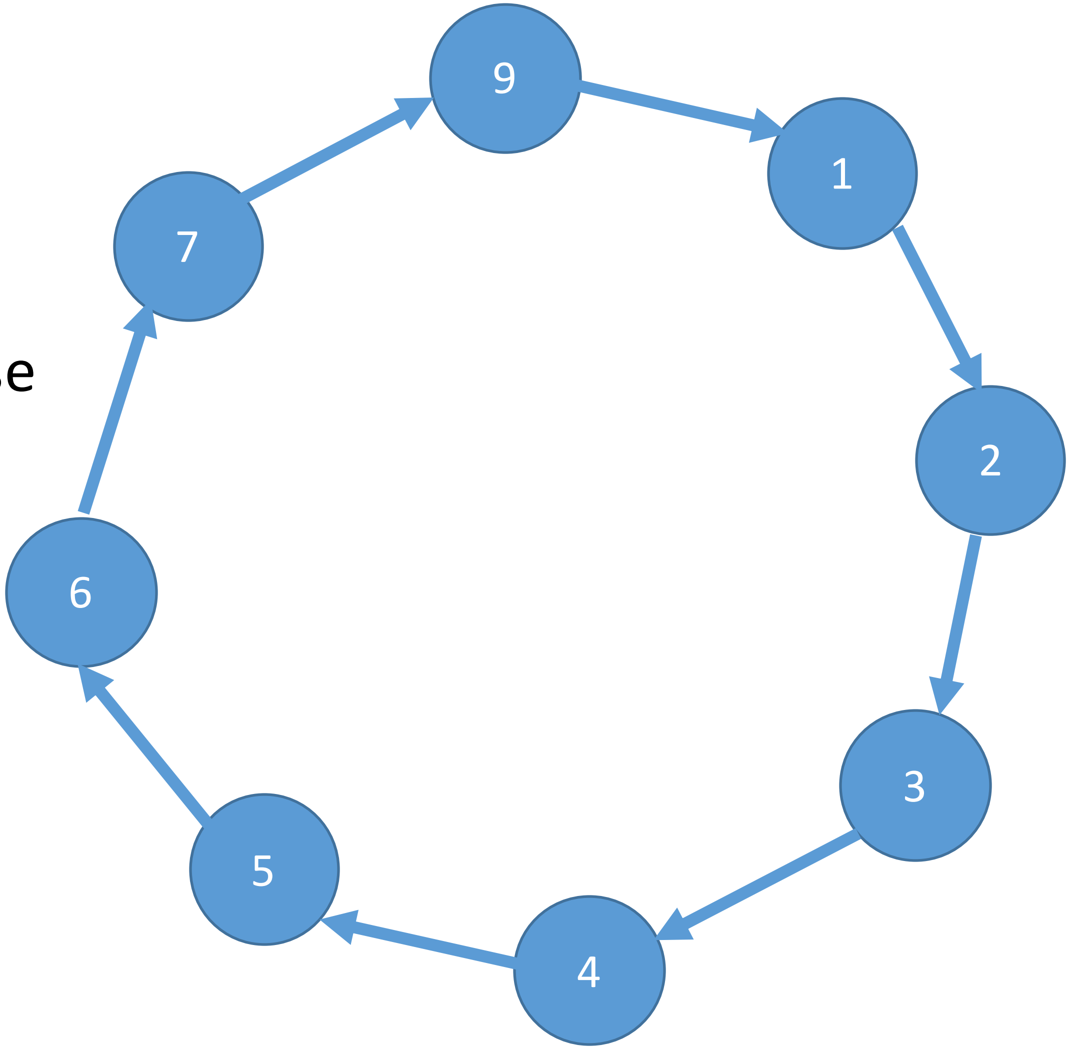
- **Worst-case Message Complexity**: $O(n^2)$
  - **When does this occur?**
  - **Arrange the IDs in decreasing order**
  - 2nd largest ID causes n-1 messages
  - 3rd largest ID causes n-2 messages
  - 4th largest ID causes n-3 messages
  - …
  - Total: $n + (n-1) + (n-2) + \ldots + 2 + 1 = O(n^2)$



Chang and Roberts Algorithm

- **Correctness:** Process with largest ID is elected as the leader, that message passes through all other processes

- **Best-case Message Complexity**: O(n)
  - **Arrange the IDs in increasing order**
  - Largest ID cases n messages, others cause exactly one message
  - Total messages = n + (1 + 1 … n-1 times)
    = 2n – 1 = O(n)



Chang and Roberts Algorithm – Average Case Analysis

- Let the IDs be 0, 1, 2, …, i, i+1, …, n-1
- Let P(i, k) be the probability that ID i makes exactly k steps
  - k-1 clockwise neighbors of i have IDs less than i and the k^{th} clockwise neighbor of i is greater than i
- There are (i - 1) processes having IDs less than i and (n - i) processes having IDs larger than i
- $P(i, k) = {}^{(i-1)}C_{(k-1)} \,/\, {}^{(n-1)}C_{(k-1)} \times (n-i)/(n-k)$
- Expected total number of messages: $E(k) = n + \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} kP(i, k).$
- The above expected value can be simplified to $n(1+½+\ldots+1/n) = O(n \log n)$

Chang and Roberts Algorithm

- The algorithm is simple and works both in synchronous and asynchronous models

- But, can we reduce the message complexity?

Chang and Roberts Algorithm

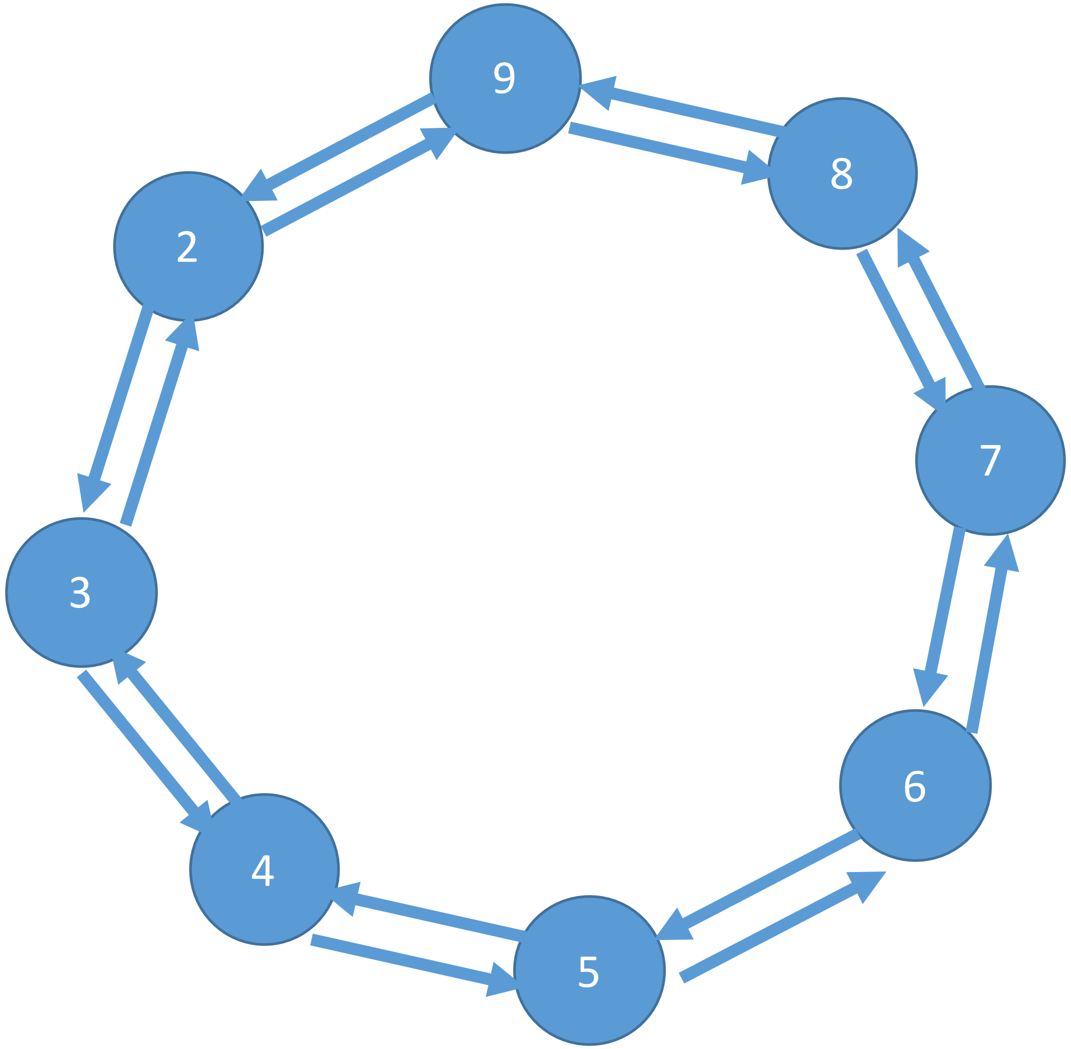
- The algorithm is simple and works both in synchronous and asynchronous models

- But, can we reduce the message complexity?

- Core Idea: **Let the messages having the larger IDs travel smaller distance in the ring**

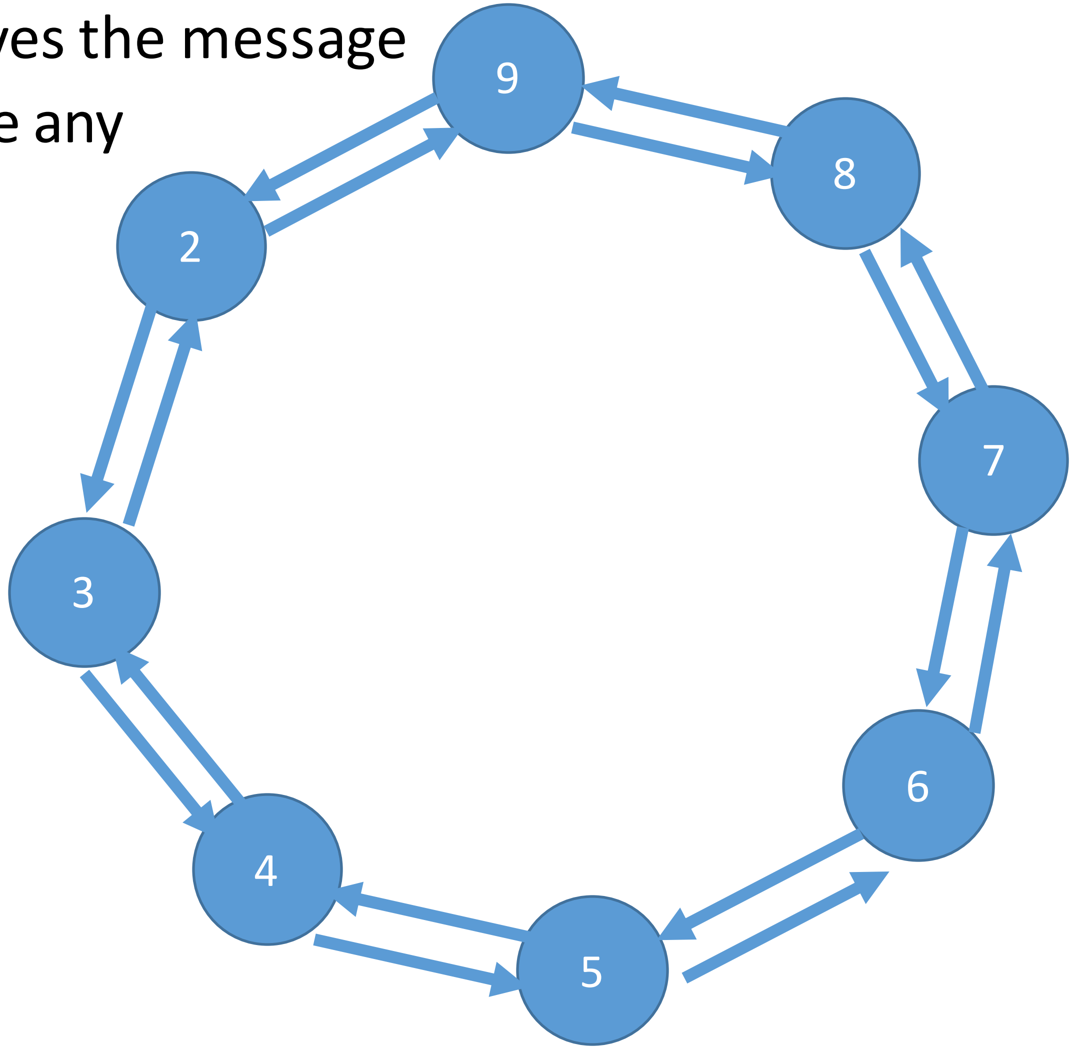
Hirschberg-Sinclair Algorithm

- **System model**: Same as earlier, but we need a bi-directional ring for this case
- We consider a scenario when each node wants to know the leader
- Define, *k*-neighborhood of a node *p*
  - *k* nodes at both sides of *p*



Hirschberg-Sinclair Algorithm

- **How does a node send message to distance *k*?**
  - Every message has a "**Time to Live**" variable
  - Each node decrements m.TTL as it receives the message
  - If m.TTL = 0, do not forward the message any further



Hirschberg-Sinclair Algorithm

- The algorithm operates in phases
- Phase 0: Node p sends an election message m to both p.NEXT and p.PREVIOUS with m.ID = p.ID and TTL=1

Hirschberg-Sinclair Algorithm

- The algorithm operates in phases
- Phase 0: Node p sends an election message m to both p.NEXT and p.PREVIOUS with m.ID = p.ID and TTL=1

- Suppose, q receives this message
  - Set m.TTL = 0
  - If q.ID > m.ID, do nothing
  - If q.ID < m.ID, return the message m to p

Hirschberg-Sinclair Algorithm

- The algorithm operates in phases
- Phase 0: Node p sends an election message m to both p.NEXT and p.PREVIOUS with m.ID = p.ID and TTL=1

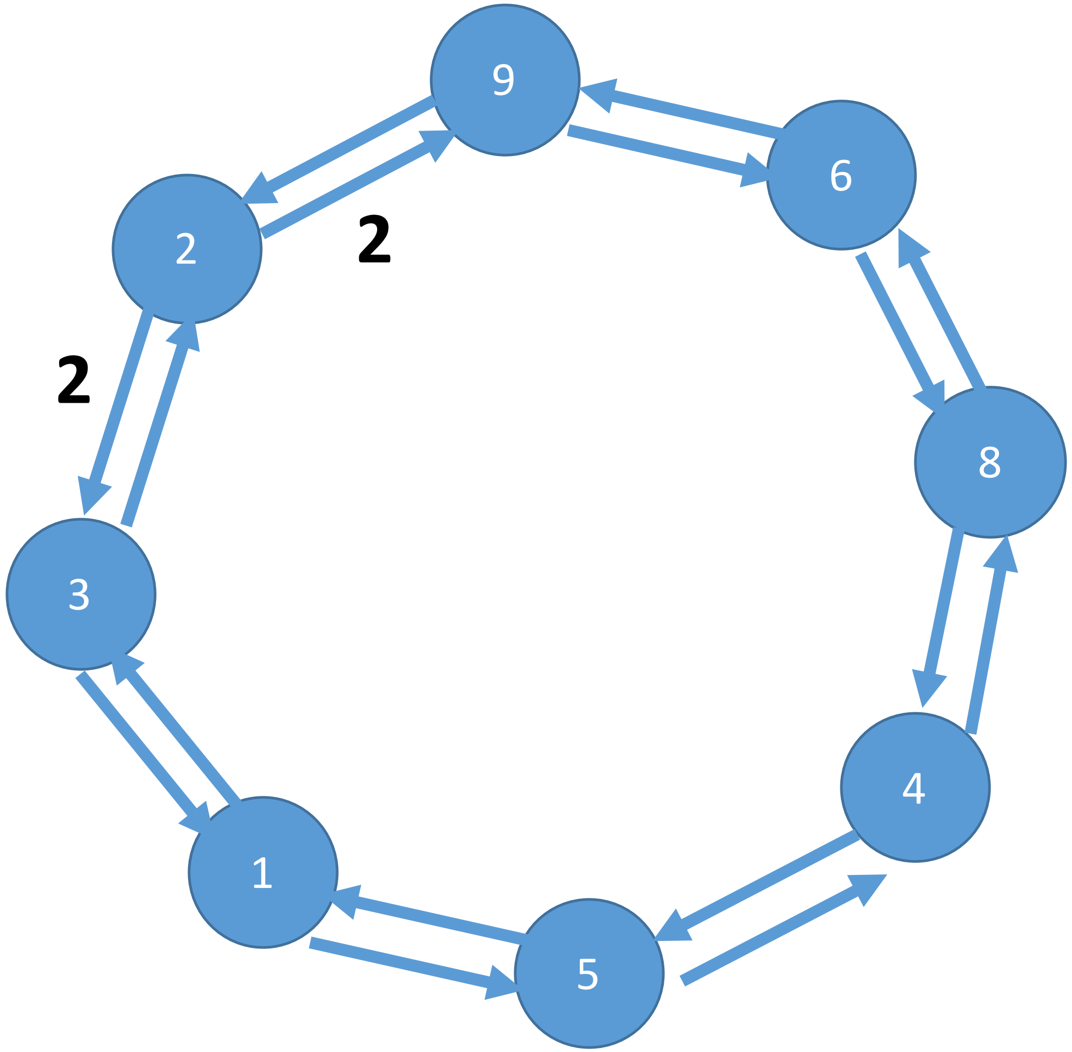
- Suppose, q receives this message
  - Set m.TTL = 0
  - If q.ID > m.ID, do nothing
  - If q.ID < m.ID, return the message m to p

- If p gets back both the messages, it declares leader of its 1 neighborhood, and proceeds to the next phase

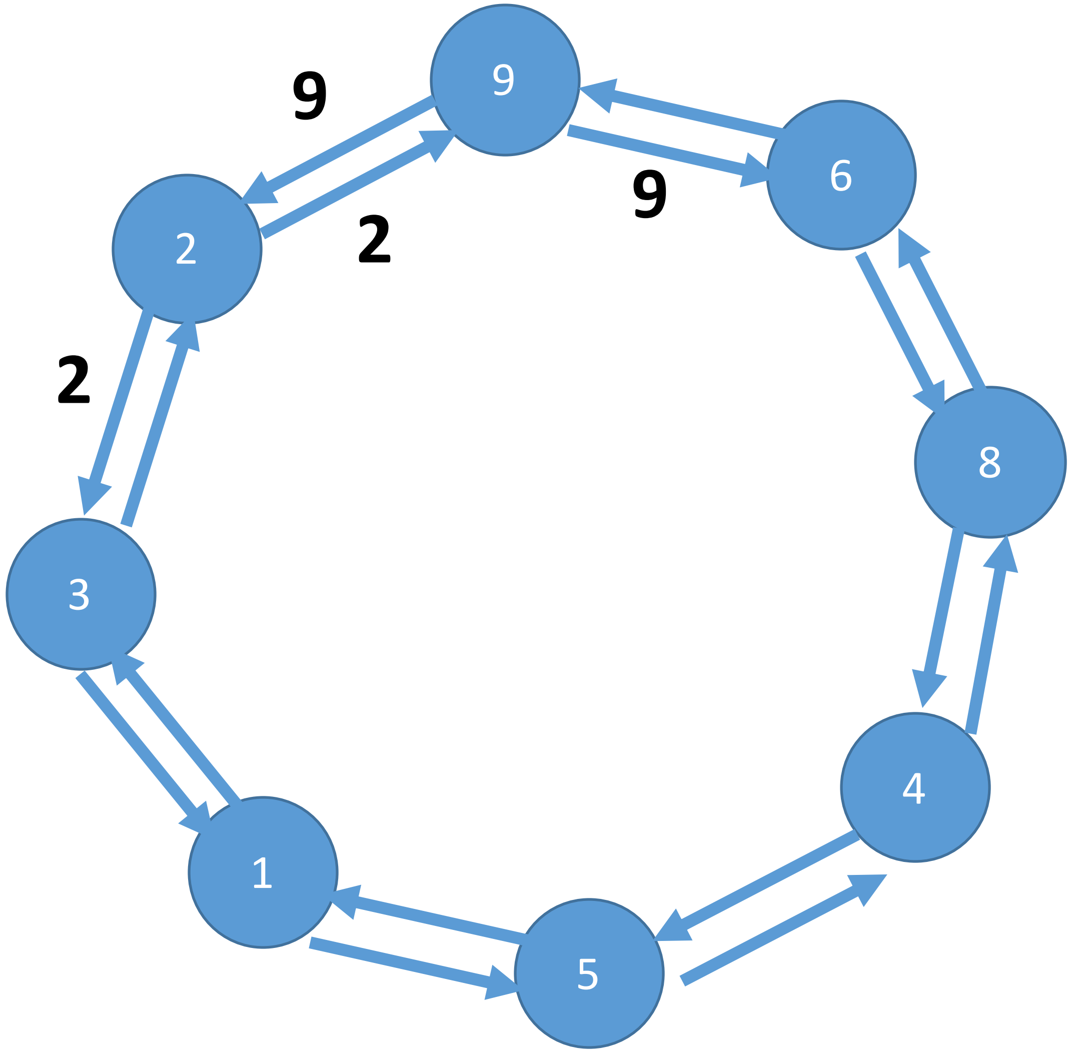
Hirschberg-Sinclair Algorithm

- The algorithm operates in phases
- Phase 0: Node p sends an election message m to both p.NEXT and p.PREVIOUS with m.ID = p.ID and TTL=1
- Suppose, q receives this message
  - Set m.TTL = 0
  - If q.ID > m.ID, do nothing
  - If q.ID < m.ID, return the message m to p
- If p gets back both the messages, it declares leader of its 1 neighborhood, and proceeds to the next phase

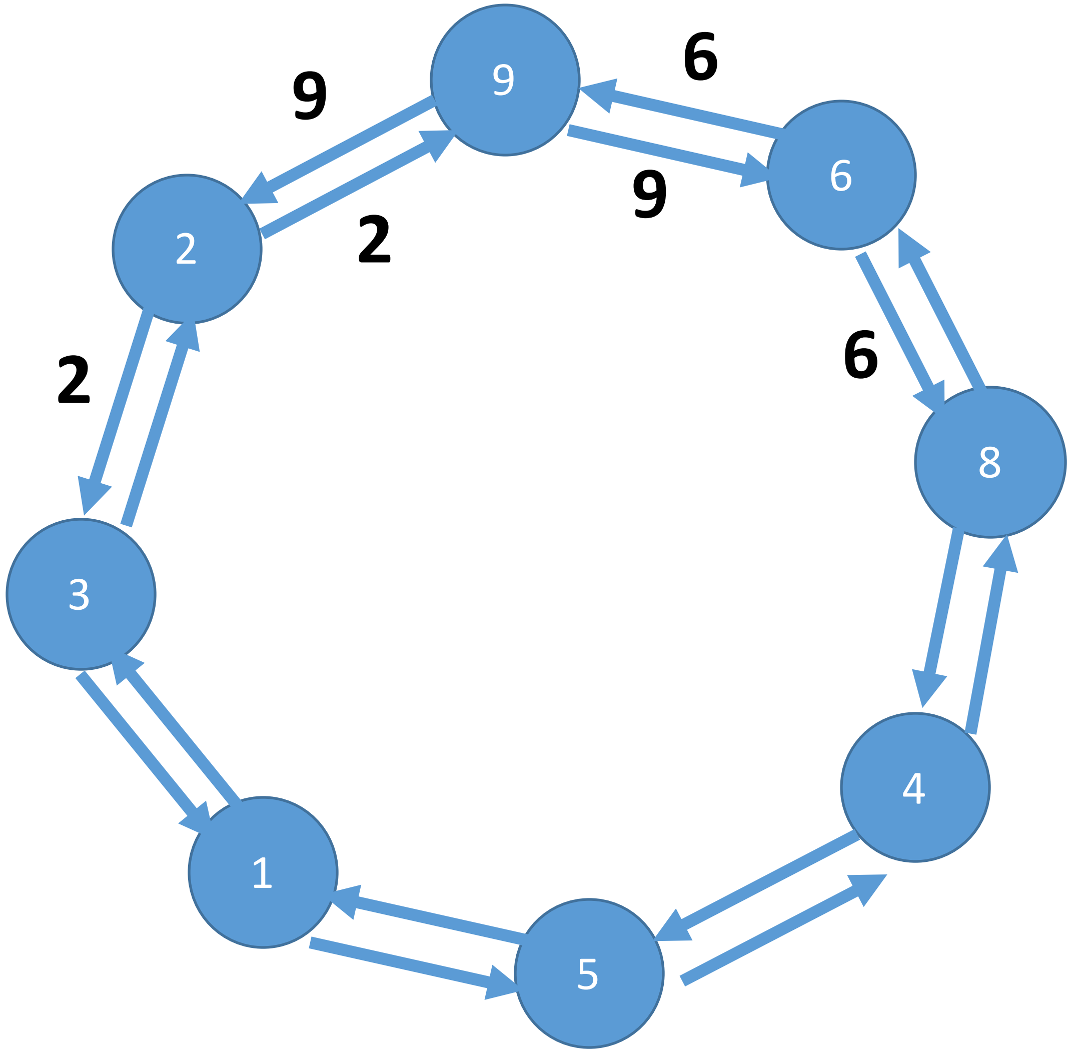
Hirschberg-Sinclair Algorithm



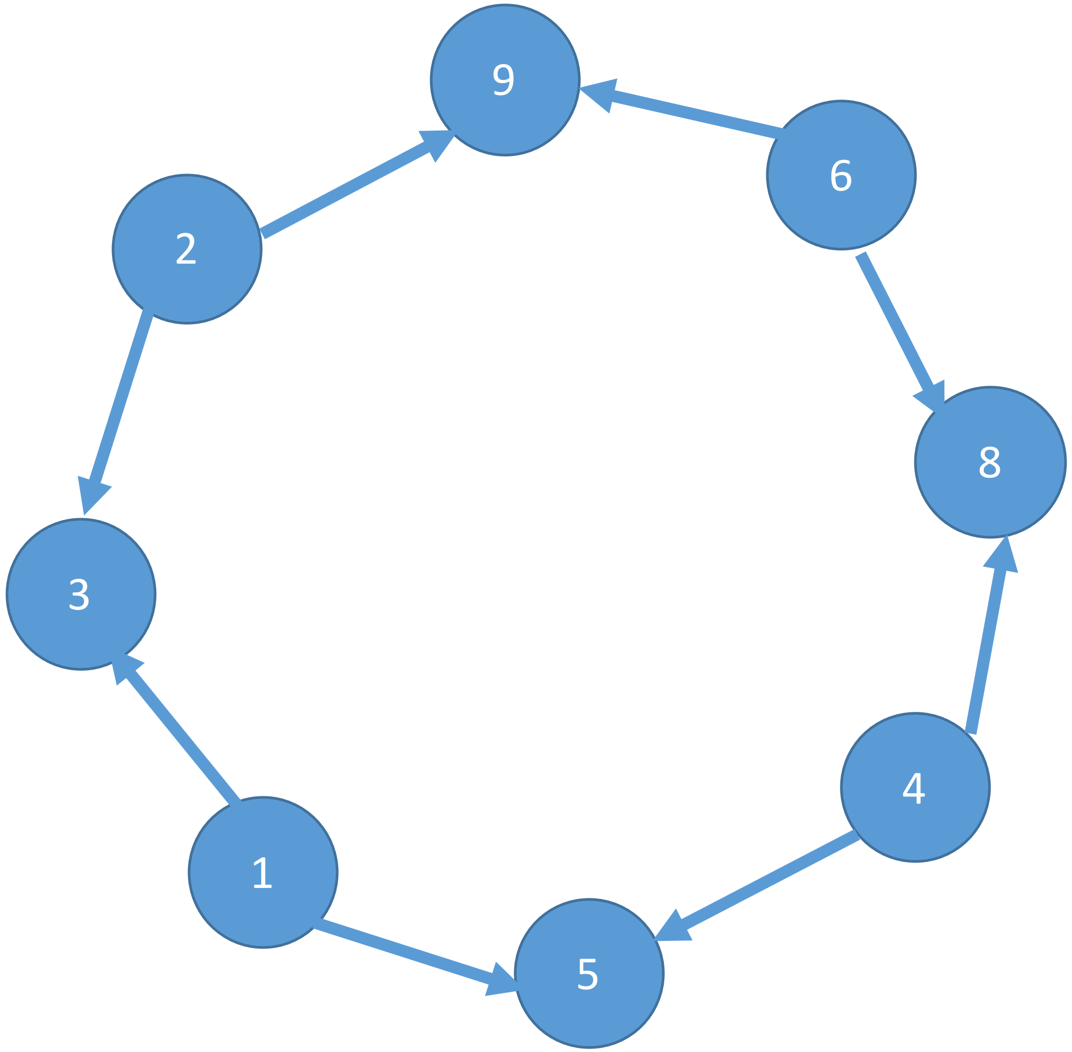
Hirschberg-Sinclair Algorithm



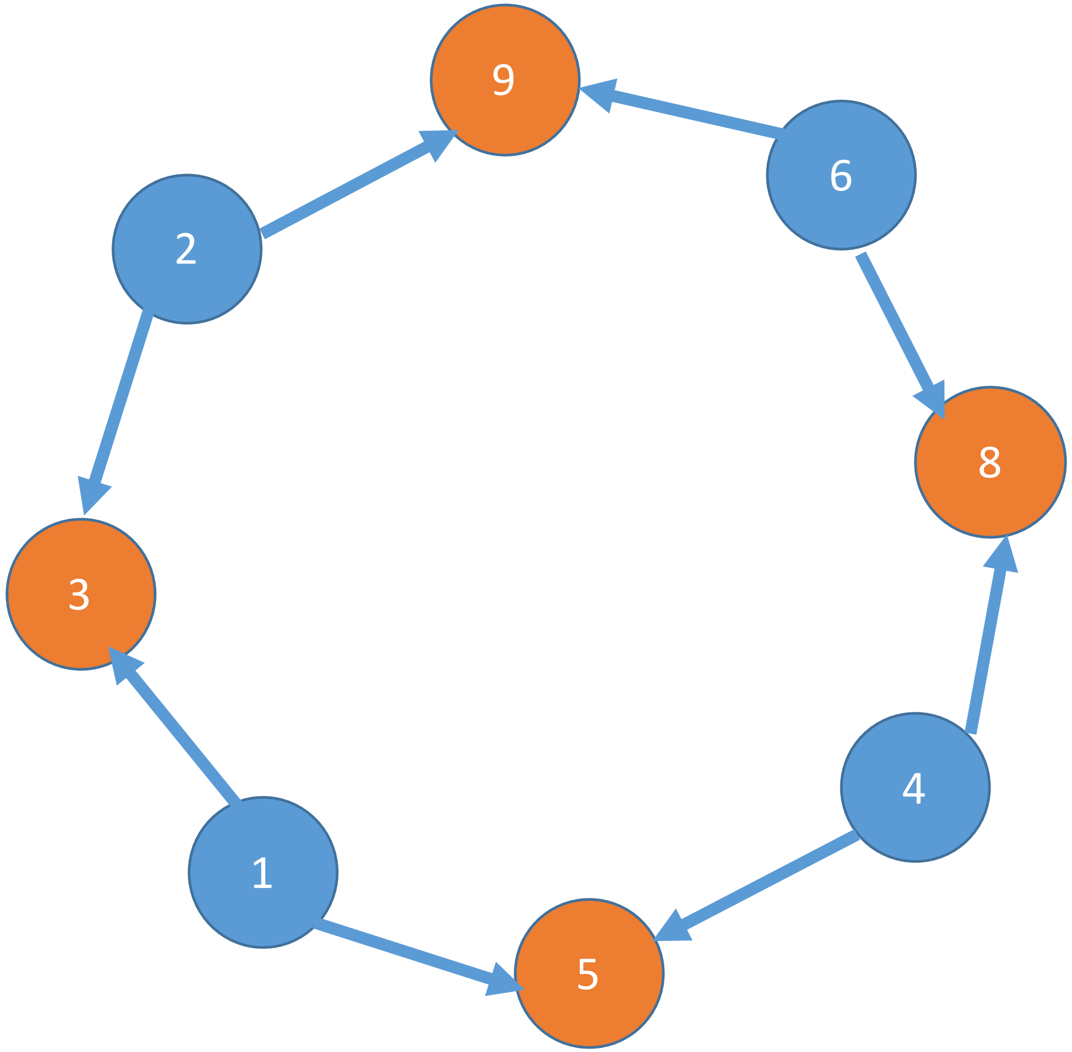
Hirschberg-Sinclair Algorithm



Hirschberg-Sinclair Algorithm



Hirschberg-Sinclair Algorithm



Hirschberg-Sinclair Algorithm

- The algorithm operates in phases
- Phase i: Node p sends an election message m to both p.NEXT and p.PREVIOUS with m.ID = p.ID and TTL=2^i

- Suppose, q receives this message
  - Set m.TTL = m.TTL-1
  - If q.ID > m.ID, do nothing
  - Else
    - If m.TTL=0, the return to the sending process
    - Else forward suitably to the previous or next process
- If p gets back both the messages, it declares leader of its 2^i neighborhood, and proceeds to the next phase

Hirschberg-Sinclair Algorithm

- When $2^i \geq n/2$, only one process gets back the message and becomes the leader
- So, number of phases = $O(\log n)$

Hirschberg-Sinclair Algorithm

- When $2^i \geq n/2$, only one process gets back the message and becomes the leader
- So, number of phases = $O(\log n)$

- **What is the message complexity?**

Hirschberg-Sinclair Algorithm – Message Complexity

- In phase i,
  - At most one process initiates message in any sequence of 2^{i-1} processes
  - So, we have a total of $n/2^{i-1}$ candidate processes for sending messages
  - Each of the candidate processes sends 2 messages going at most 2^i distance; so total number of messages = 2×2^i
  - So, total messages = $O(n)$ in the ith phase
- There are $O(\log n)$ phases
- Therefore, message complexity = $O(n \log n)$

Hirschberg-Sinclair Algorithm – Message Complexity

- In phase i,
  - At most one process initiates message in any sequence of 2^{i-1} processes
  - So, we have a total of $n/2^{i-1}$ candidate processes for sending messages
  - Each of the candidate processes sends 2 messages going at most 2^i distance; so total number of messages = $2 \times 2 \times 2^i$
  - So, total messages = O(n) in the ith phase
- There are O(log n) phases
- Therefore, message complexity = O(n log n)

- **Can we do it in O(n log n) for unidirectional ring?**

Hirschberg-Sinclair Algorithm – Message Complexity

- In phase i,
  - At most one process initiates message in any sequence of 2^{i-1} processes
  - So, we have a total of $n/2^{i-1}$ candidate processes for sending messages
  - Each of the candidate processes sends 2 messages going at most 2^i distance; so total number of messages = $2 \times 2 \times 2^i$
  - So, total messages = O(n) in the ith phase
- There are O(log n) phases
- Therefore, message complexity = O(n log n)

- **Can we do it in O(n log n) for unidirectional ring?**
  - Yes, Peterson's Algorithm

Hirschberg-Sinclair Algorithm – Message Complexity

- In phase i,
  - At most one process initiates message in any sequence of 2^{i-1} processes
  - So, we have a total of $n/2^{i-1}$ candidate processes for sending messages
  - Each of the candidate processes sends 2 messages going at most 2^i distance; so total number of messages = $2 \times 2 \times 2^i$
  - So, total messages = O(n) in the ith phase
- There are O(log n) phases
- Therefore, message complexity = O(n log n)

- **Can we do it better than O(n log n)?**

Hirschberg-Sinclair Algorithm – Message Complexity

- In phase i,
  - At most one process initiates message in any sequence of 2^{i-1} processes
  - So, we have a total of $n/2^{i-1}$ candidate processes for sending messages
  - Each of the candidate processes sends 2 messages going at most 2^i distance; so total number of messages = $2 \times 2 \times 2^i$
  - So, total messages = O(n) in the ith phase
- There are O(log n) phases
- Therefore, message complexity = O(n log n)

- **Can we do it better than O(n log n)? Not in asynchronous ring**
  - Any asynchronous Leader Election algorithm requires $\Omega(n \log n)$ messages.

Variable Time Algorithm in a Synchronous Ring

- Synchronous, round-based algorithm
  - Round = Maximum message transmission delay
  - A phase is equal to n rounds

- Node k does the following
  - If no message received when k-th phase starts, declare itself the leader and send a leader message with its id around the ring
  - If message received before k-th phase starts, record id in message as leader and forward the message around the ring

- Message complexity O(n)

Happy
Learning