Chapter 6
COORDINATION AND AGREEMENT

1. Mutual Exclusion in Distributed Systems

- Distributed processes often need to communicate their activities. If a collection of process share a resource or resources, then often mutual exclusion is required to prevent interference and ensure consistency when accessing the resources.
- The concurrent access of processes to a shared resource executed in a mutually exclusive manner in a distributed system is called distributed mutual exclusion.
- If two processes are allowed to concurrently be in competing critical sections, then incorrect results may be computed –called a Race Condition. The process of ensuring that this destructive interaction does not occur is called mutual exclusion (mutex). In other words, mutual exclusion is the process of making only one process to enter into the critical section at the same time.
- Mutual exclusion is a mechanism that prevent interference and ensure consistency when accessing the resources by a collection of processes.

In single computer system, memory and other resources are shared between different processes. The status of shared resources and the status of users is easily available in the shared memory so with the help of shared variable (For example: Semaphores) mutual exclusion problem can be easily solved.

In Distributed systems, we neither have shared memory nor a common physical clock and there for we can not solve mutual exclusion problem using shared variables or local kernel. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used.

The basic requirements for mutual exclusion mechanism are:

I. At most one process may execute in the critical section at a time.
II. A process is granted entry to critical section if no any other process is executing within critical section.

2. Algorithms for Mutual Exclusion

There are three basic approaches to distributed mutual exclusion:

Non-token-based (permission based):
- In this method each process freely and equally competes for the right to use the shared resource; requests are arbitrated by a central control site or by distributed agreement.
- It includes central coordinator algorithm, Lamport algorithm and Ricart-Agrawala algorithm.

Token-based:
- In this method a logical token representing the access right to the shared resource is passed in a regulated fashion among the processes; whoever holds the token is allowed to enter the critical section.
- It includes Ricart-Agrawala second algorithm and token ring algorithm.

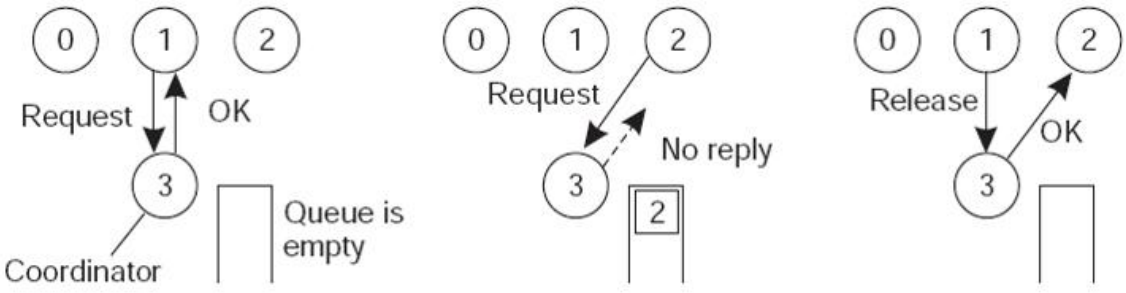
Quorom-based:
- In this method, each site requests a permission to execute critical section from a subset of sites, known as quorom.

Non-token-based Algorithms

A. Central Coordinator Algorithm

A central coordinator grants permission to enter a CS. In this algorithm, a process that wants to enter into a CS has to take a permission from a central coordinator.

Algorithm
- ✧ To enter a CS, a process sends a request message to the coordinator and then waits for a reply (during this waiting period the process can continue with other work).
- ✧ The reply from the coordinator gives the right to enter the CS.
- ✧ After finishing work in the CS the process notifies the coordinator with a release message.



Advantages
i. The scheme is simple and easy to implement.
ii. It requires only three messages per use of a CS (request, OK, release).

Disadvantages
i. System performance may degrade
ii. If coordinator crashes, we need an election algorithm.

B. Ricart-Agrawala Algorithm

- Ricart–Agrawala algorithm is an algorithm for mutual exclusion in a distributed system proposed by Glenn Ricart and Ashok Agrawala. This algorithm is an extension and optimization of Lamport's Distributed Mutual Exclusion Algorithm. Like Lamport's Algorithm, it also follows permission based approach to ensure mutual exclusion.
- In this algorithm, two type of messages (REQUEST and REPLY) are used and communication channels are assumed to follow FIFO order.
- A site send a REQUEST message to all other site to get their permission to enter critical section. A site send a REPLY message to other site to give its permission to enter the critical section.
- A timestamp is given to each critical section request using Lamport's logical clock.Timestamp is used to determine priority of critical section requests. Smaller timestamp gets high priority over larger timestamp. The execution of critical section request is always in the order of their timestamp.

Algorithm:

i. To enter Critical section:

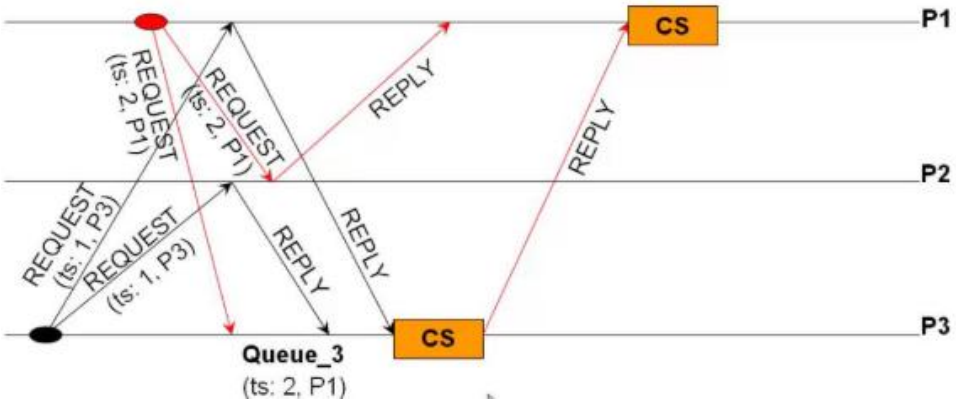
- ✓ When a site Si wants to enter the critical section, it sends a timestamped REQUEST message to all other sites.
- ✓ When a site Sj receives a REQUEST message from site Si, it sends a REPLY message to site Si if and only if Site Sj is neither requesting nor currently executing the critical section.
- ✓ In case Site Sj is requesting, the timestamp of Site Si's request is smaller than its own request. Otherwise the request is deferred by site Sj.

ii. To execute the critical section:

- ✓ Site Si enters the critical section if it has received the REPLY message from all other sites.

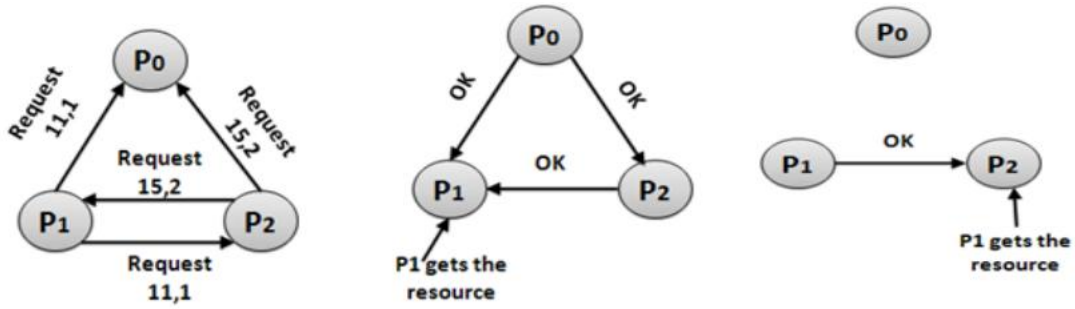
iii. To release the critical section:

- ✓ Upon exiting site Si sends REPLY message to all the deferred requests.



Each process keeps its state w.r.t critical section which can be:

- ✧ Requested
- ✧ Held
- ✧ Released



Problems

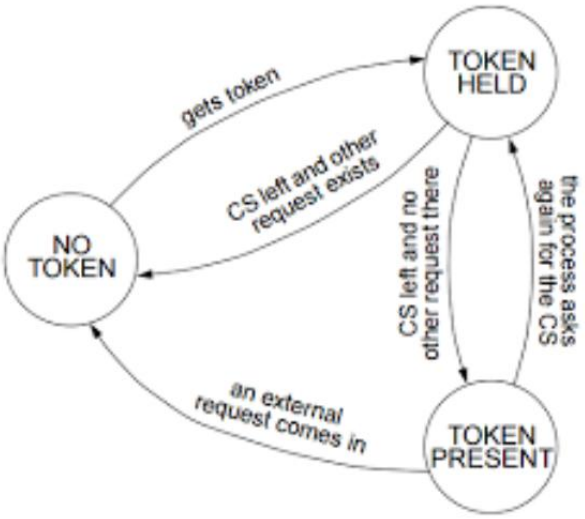
- ✧ It is expensive to handle message traffic [2(n-1) messages] ((N – 1) request messages and (N – 1) reply messages)
- ✧ Failure of one involved process blocks the progress.

Token Based Algorithm

A token is circulated in a logical ring. A process enters its CS if it has the token.

A. Ricart-Agrawala Second Algorithm

- ➢ A process is allowed to enter the critical section when it got the token. In order to get the token it sends a request to all other processes competing for the same resource. The request message consists of the requesting process' timestamp (logical clock) and its identifier.
- ➢ Initially the token is assigned arbitrarily to one of the processes. When a process Pi leaves a critical section it passes the token to one of the processes which are waiting for it; this will be the first process Pj, where j is searched in order [i+1, i+2, ..., n, 1, 2, ..., i-2, i-1] for which there is a pending request.
- ➢ If no process is waiting, Pi retains the token (and is allowed to enter the CS if it needs); it will pass over the token as result of an incoming request.

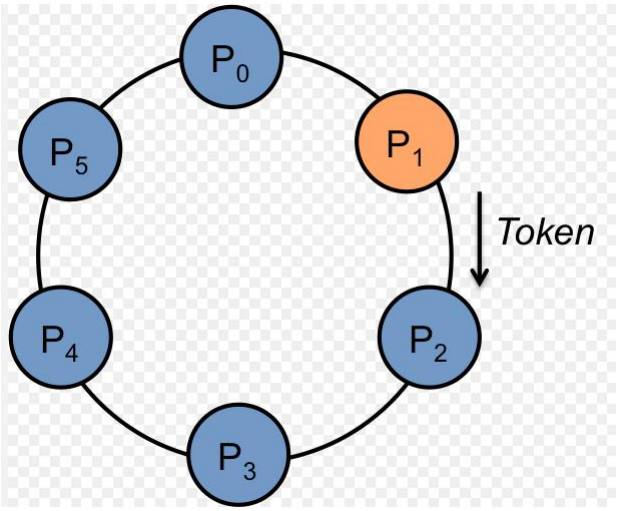


Advantages:

- ✧ It requires only n-1 requests and one reply.
- ✧ Failure of process which is not holding token does not prevent progress.

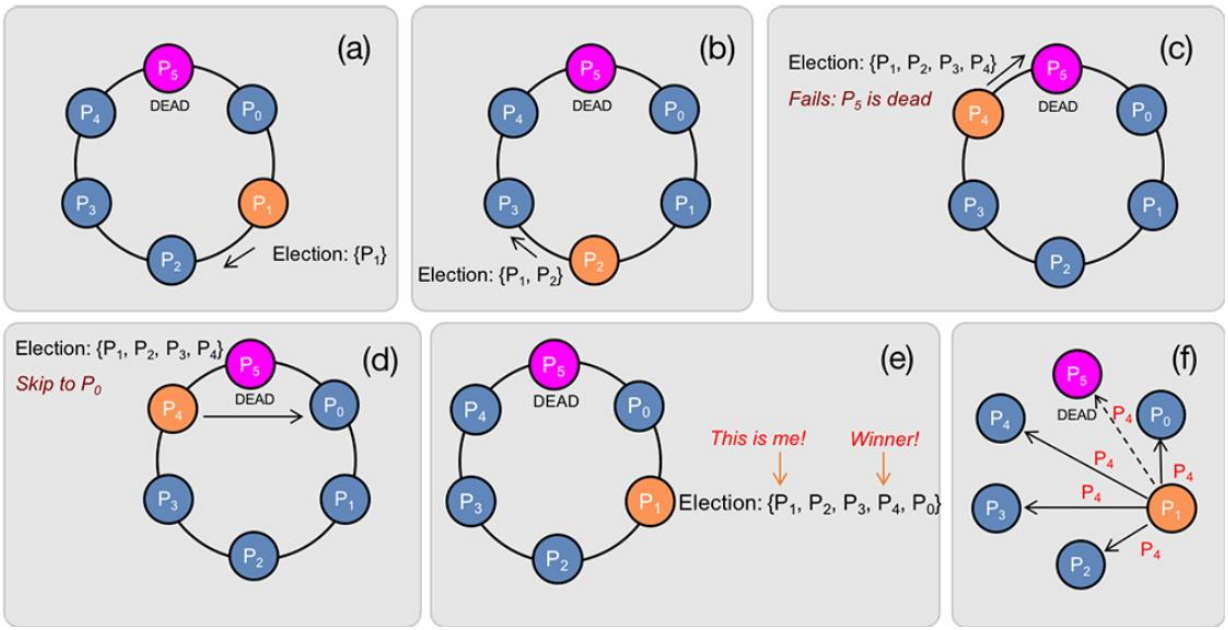
B. Token Ring Algorithm

- The n processes are arranged in a logical ring as shown in given figure:



Algorithm:

- Token is initially given to one process.
- When a process requires to enter CS, it waits until it gets token from its left neighbor and retains it. After it left CS, it passes token to its neighbor in clockwise direction.
- If a process gets token but does not require to enter CS, it immediately passes token along the ring.



Problem:

- It adds load to the network as token should be passed even the process do not need it.
- If one process fails, no progress is possible until the faulty process is extracted from the ring.
- Election process should be done if the process holding the token fails.

3. Distributed Elections

- Many distributed algorithms require one process to act as a coordinator or, in general, perform some special role, and that coordinator is selected using an Election Algorithm.
- A single leader makes system easy to understand, puts all the concurrency in the system into a single place, reduces partial failure and adds a single place to look for logs or metrics.

- Any process can be elected but only one process must be elected and all other processes must agree on it. Election is started after a failure of current coordinator.

- Phases of Election Algorithm
  - ✧ Select a leader with the highest priority.
  - ✧ Inform all processes about the winner

- Election algorithm assumes that every active process in the system has a unique priority number. The process with highest priority will be chosen as a new coordinator. Hence, when a coordinator fails, this algorithm elects that active process which has highest priority number.Then this number is send to every active process in the distributed system.

- Need of Election algorithm
  - ✧ Clock Synchronization
  - ✧ Mutual Exclusion
  - ✧ Any Distributed Computing

A. Bully Algorithm

- It assumes that each process knows identifiers of other processes and it can communicate with process with higher identifier. It assumes that the system is synchronous.
- It has three types of messages : election message, answer message and coordinator message.

- Any process can crash during the election. This algorithm is applicable to elect a leader in a distributed system connected with each other (say in a mesh topology). The bully algorithm is a method in distributed computing for dynamically selecting a coordinator by process ID number.
- When a process Pi detects a failure and election is to be held, it sends election message to all processes with higher identifier and waits for the answer message.

Algorithm:

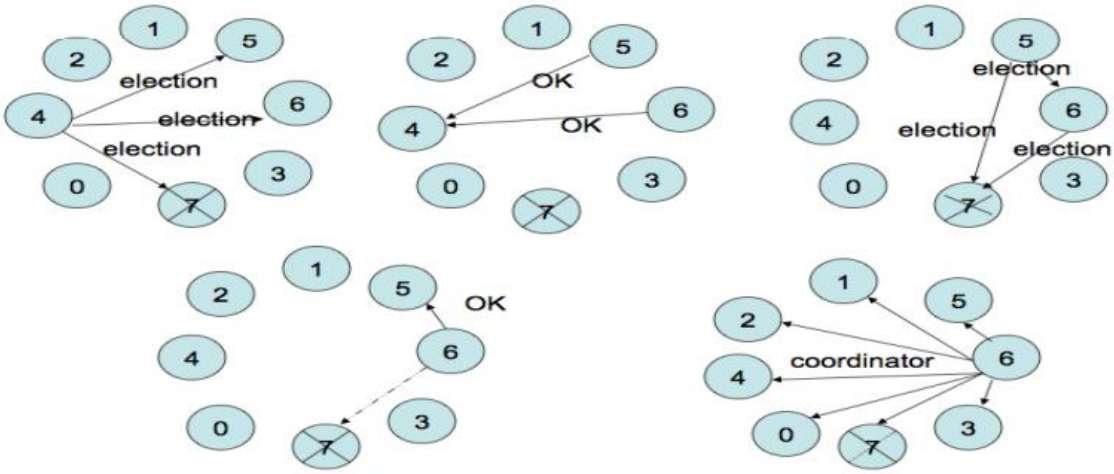
i. Rule for election process initiator

```
State Pi := ELECTION-ON
Pi sends election message to processes with higher identifier
        Pi waits for answer message
        if no answer message before timeout then
        Pi is coordinator and sends coordinator message to all processes
else
      Pi waits for coordinator message
```

if no coordinator message arrives before timeout then restart election procedure
end if
end if

ii. Rule for handling incoming election message

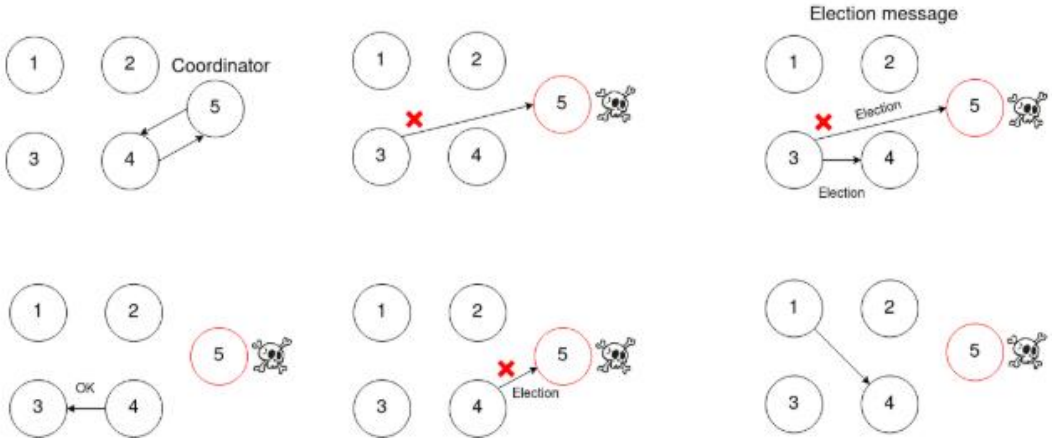
Pi replies with an answer message to Pj
if state Pi := ELECTION-OFF then
start election procedure
end if



Best Case Scenario:
The process with second highest identifier notices coordinator's failure, it then selects itself as coordinator and sends (n-2) coordinator messages.

Worst Case Scenario:
The process with lowest identifier indicates election. It sends (n-1) election messages to processes which themselves initiate an election. So, O(n2) messages are required.

Another Example:



B. Ring Based Algorithm (Chang and Roberts algorithm)

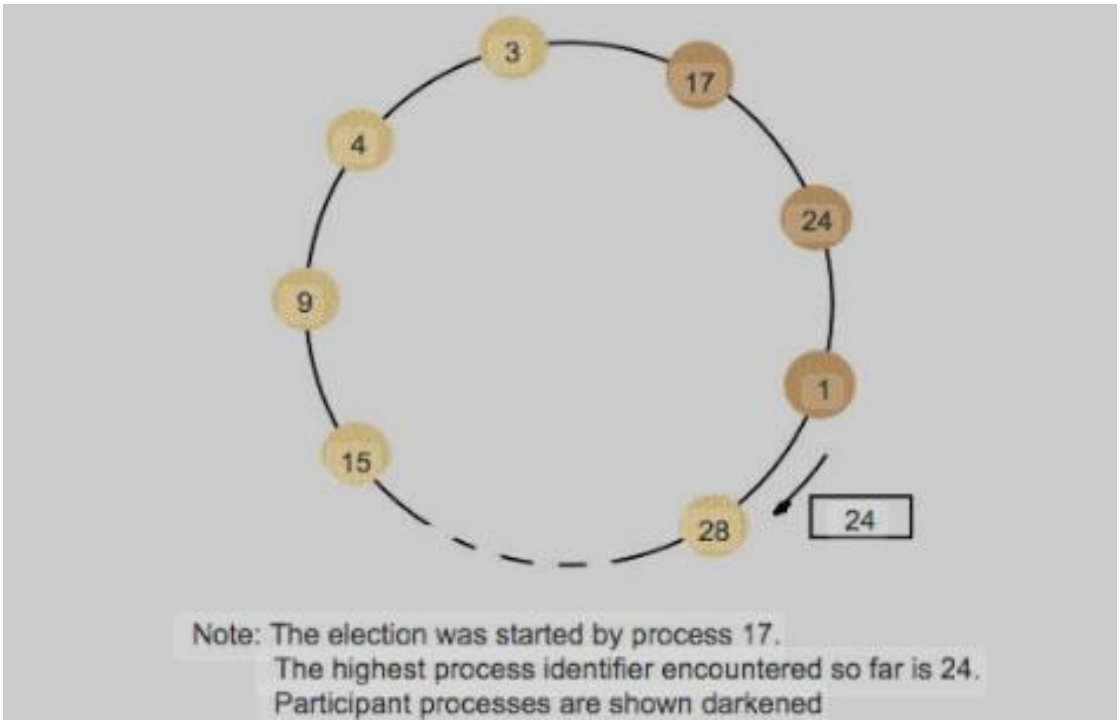
- Chang and Roberts is a ring-based election algorithm used to find a process with the largest identification. It is a useful method of election in decentralized distributed computing where the systems are connected in a logical or physical ring.
- The processes are arranged in a logical ring. Each process knows address of one other neighbor process in the clockwise direction. It assumes that the system is asynchronous.
- The algorithm works for any number of processes N, and does not require any process to know how many processes are in the ring. Is often referred as a ring algorithm.

Algorithm

i. Initially each process in the ring is marked as non-participant.
ii. A process that notices a lack of leader starts an election. It creates an election message containing its UID. It then sends this message clockwise to its neighbor.
iii. Every time a process sends or forwards an election message, the process also marks itself as a participant.

iv. When a process receives an election message it compares the UID in the message with its own UID.
    - ✧ If the UID in the election message is larger, the process unconditionally forwards the election message in a clockwise direction.
    - ✧ If the UID in the election message is smaller, and the process is not yet a participant, the process replaces the UID in the message with its own UID, sends the updated election message in a clockwise direction.
    - ✧ If the UID in the election message is smaller, and the process is already a participant (i.e., the process has already sent out an election message with a UID at least as large as its own UID), the process discards the election message.
    - ✧ If the UID in the incoming election message is the same as the UID of the process, that process starts acting as the leader.

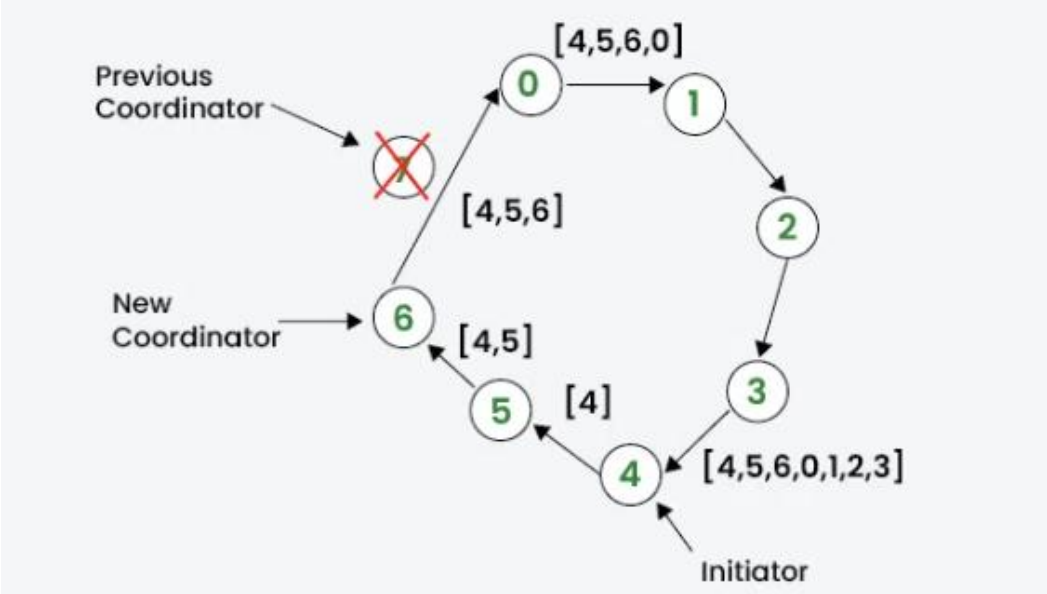
v. When a process starts acting as the leader, it begins the second stage of the algorithm.
vi. The leader process marks itself as non-participant and sends an elected message to its neighbor announcing its election and UID.
vii. When a process receives an elected message, it marks itself as non-participant, records the elected UID, and forwards the elected message unchanged.
viii. When the elected message reaches the newly elected leader, the leader discards that message, and the election is over.



- Average Case:
  - ✧ n/2 message need to reach maximal node.
  - ✧ n message to return maximal node.
  - ✧ n message to rotate elected message.
  - ✧ Total message : 2n + n/2

- Worst Case:
  - ✧ n-1 message to reach maximal node
  - ✧ Total message : 3n -1

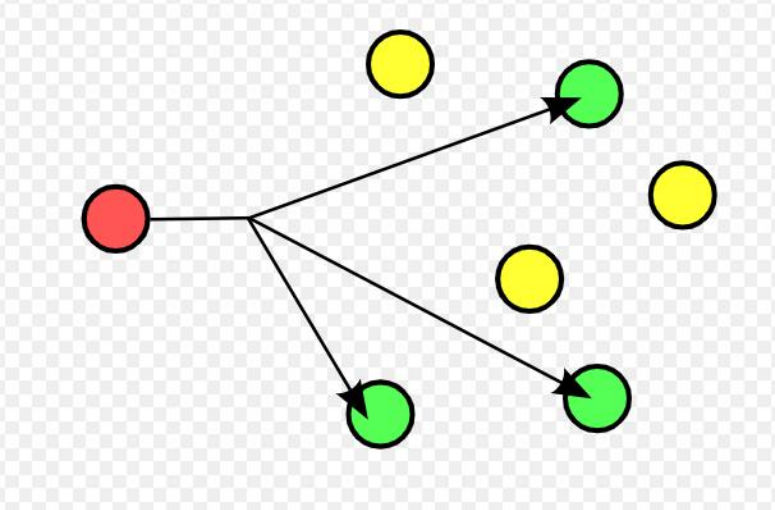
Another Example:



4. Multicast Communication

- The process of sending a message to multiple nodes is called multicast.
- It requires coordination and agreement. The essential feature of multicast is that a process issues only one multicast operation to send a message to each of a group of processes instead of issuing multiple send operations to individual processes. It is efficient in utilization of bandwidth.

- Multicasting in computer network is a group communication, where a sender(s) send data to multiple receivers simultaneously. It supports one – to – many and many – to – many data transmission across LANs or WANs. Through the process of multicasting, the communication and processing overhead of sending the same data packet or data frame in minimized.



- A node can join a multicast group, and receives all messages sent to that group. The sender sends only once: to the group address. The network takes care of delivering to all nodes in the group.

5. Consensus

- The task of getting all processes in a group to agree on some specific value based on the votes of each processes.
- All processes must agree upon the same value and it must be a value that was submitted by at least one of the processes (i.e., the consensus algorithm cannot just invent a value).

- Examples:
  - ✧ Clock Synchronization
  - ✧ Distributed Mutual Exclusion
  - ✧ Leader Election
  - ✧ Synchronizing Replications
  - ✧ Distributed, fault-tolerant logging with globally consistent sequencing
  - ✧ Managing group membership, etc.