UNIVERSITY OF TARTU
Institute of Computer Science

LTAT.06.007 Distributed Systems

Lecture 7 – Coordination II (Mutual exclusion)

Huber Flores, PhD

ASSOCIATE PROFESSOR

Tartu, Estonia 30/03/2020

Recap

- Explored the primitives of clock synchronization
- Learned the purpose of logical clocks in distributed systems

Agenda

- **Goal:** To study the concept of mutual exclusion and election algorithms
- **Content:**
  - Mutual exclusion
    - Centralized algorithm
    - Decentralized algorithm
    - Distributed algorithm
    - Comparison of mutual exclusion algorithms
  - Election algorithms
    - Traditional election algorithms
    - Election in Wireless Sensor Networks (WSN)

After this lecture, you should be able to:

- Understand the importance of mutual exclusion

Mutual exclusion

Problem

A number of processes in a distributed system want exclusive access to some resource.

Goal

- **Starvation:** every process gets a chance to access the resource
- **Deadlocks:** several processes are waiting for each other to proceed

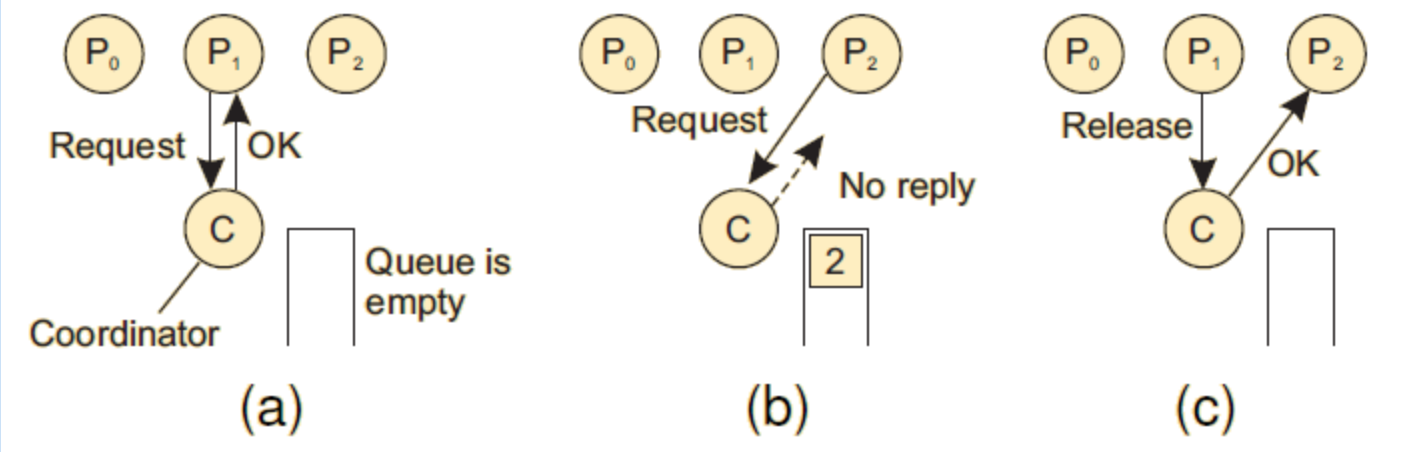
Basic solutions

- **Permission-based:** A process wanting to enter its critical section, or access a resource, needs permission from other processes.
- **Token-based:** A token is passed between processes. The one who has the token may proceed in its critical section, or pass it on when not interested.

Permission-based, centralized

Simply use of a coordinator

- One process is elected as the coordinator, which only lets one process at a time to access the resource



a) Process P_1 asks the coordinator for permission to access a shared resource. Permission is granted.
b) Process P_2 then asks permission to access the same resource. The coordinator does not reply.
c) When P_1 releases the resource, it tells the coordinator, which then replies to P_2.

Permission-based, centralized

Observations

- Pros
  - Fairness
  - No starvation
  - Simplicity, only three messages (request, grant, release)

- Cons
  - Coordinator is a single point of failure and performance bottleneck
  - Distinguishing crashed coordinator from permission denied

A distributed algorithm (Ricart & Agrawala 1981)

The same as Lamport except that acknowledgments are not sent

- Return a response to a request only when:
  - The receiving process has no interest in the shared resource; or
  - The receiving process is waiting for the resource, but has lower priority (known through comparison of timestamps).
- In all other cases, reply is deferred, implying some more local administration

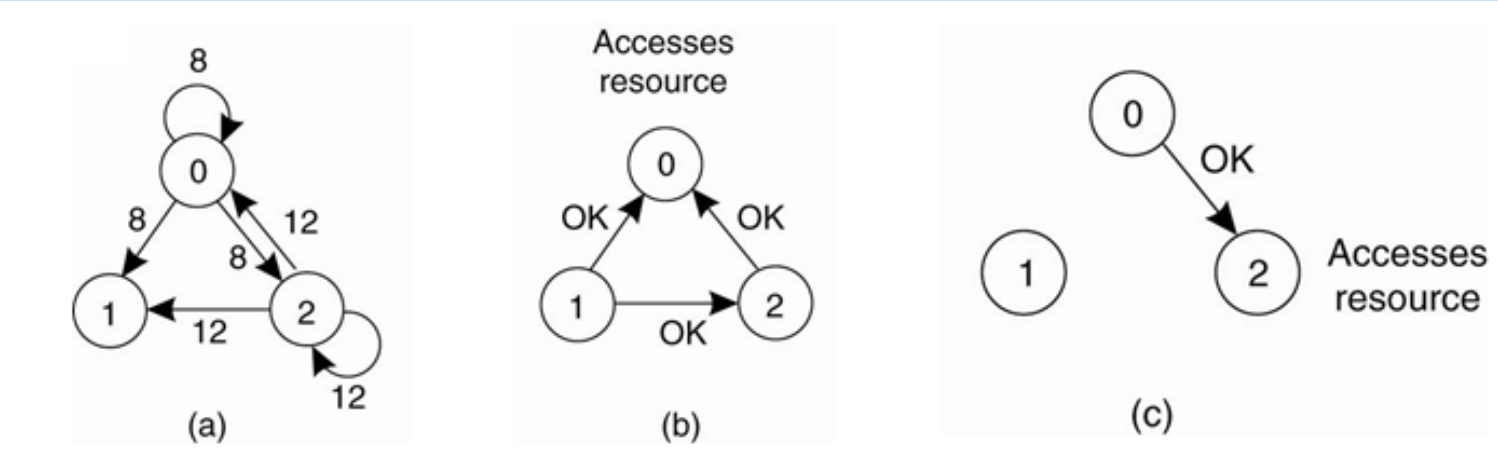
A distributed algorithm (Ricart & Agrawala 1981)

Observations

- Based on total ordering of all events in the system
- A process wanting to access a resource builds a message containing the name of the resource, its process number and the current (logical) time
- The process sends the message to all other processes
- A process receiving the request has three alternatives:
  1. If the receiver is not accessing the resource and does not want to access it, it sends back an OK message to the sender.
  2. If the receiver already has access to the resource, it simply does not reply. Instead, it queues the request.
  3. If the receiver wants to access the resource as well but has not yet done so, it compares the timestamp of the incoming message with the one contained in the message that it has sent everyone. The lowest one wins. If the incoming message has a lower timestamp, the receiver sends back an OK message. If its own message has a lower timestamp, the receiver queues the incoming request and sends nothing.
- The process waits until everyone has given permission

Mutual exclusion: Ricart & Agrawala

Essence



a) Two processes want to access a shared resource at the same moment.
b) P_0 has the lowest timestamp, so it wins.
c) When process P_0 is done, it sends an OK also, so P_2 can now go ahead.

Mutual exclusion: Ricart & Agrawala

Observations

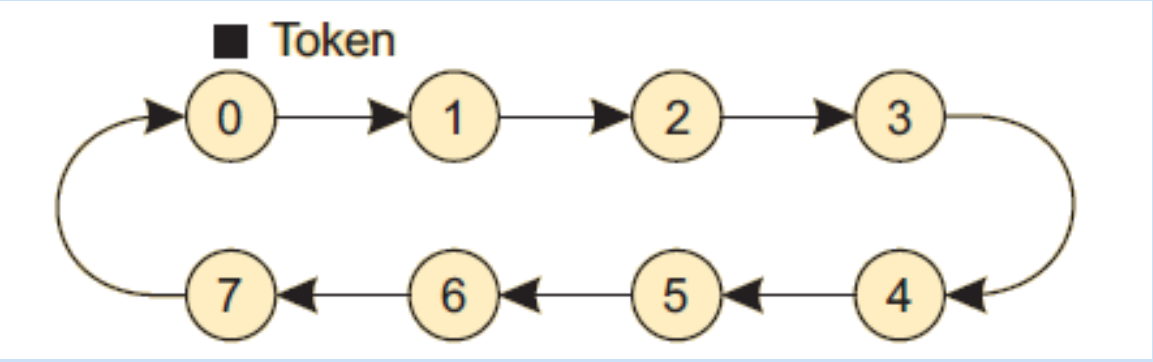
- Pros
  - Mutual exclusion is guaranteed without deadlock or starvation
- Cons
  - N points of failure (crashed process interpreted as denial of access)
  - Requires more communication
  - Low efficiency, as all processes are involved in all decisions (n bottlenecks)

Mutual exclusion:Token ring algorithm

Essence

- Organize processes in a logical ring, a token is introduced, and the idea is to let the token be passed between processes. The one that holds the token is allowed to enter the critical region (if it wants to).

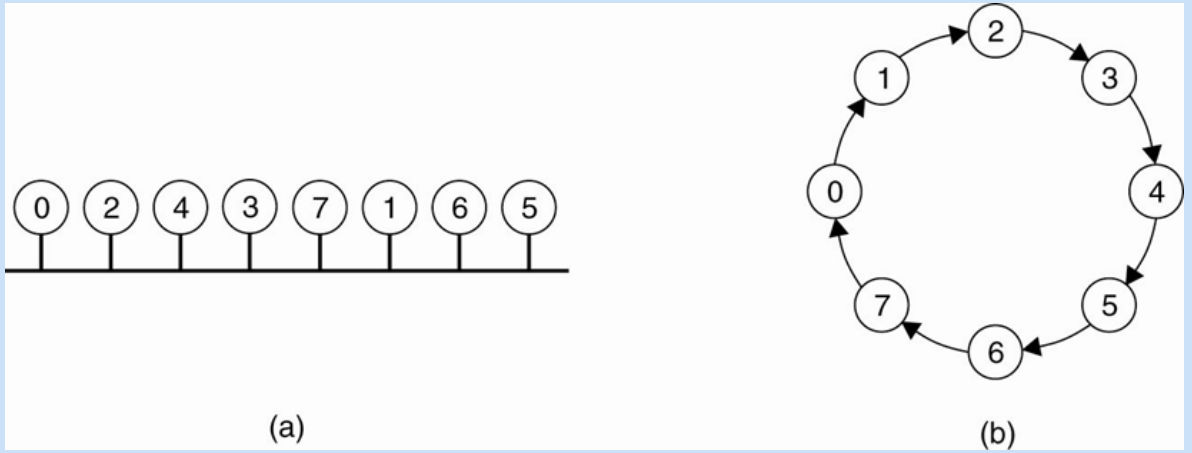
An overlay network constructed as a logical ring with a circulating token



Mutual exclusion:Token ring algorithm

Characteristics

a) An unordered group of processes on a network



- Upon initialization process 0 is given a token
- Token is passed from process k to process k+1 (modulo the ring size) in point-to-point messages
- Token grants access to a shared resource

Mutual exclusion:Token ring algorithm

Observations

- Pros
  - No starvation
  - Relatively easy to recover
- Cons
  - Token can dissapear (be loss)
  - Long delay between successive appearances of the token

A decentralized algorithm (Lin et al. 2004)

Principle

- Extension of the central coordinator
  - Each resource is assumed to be replicated n times
  - Each replica has its own coordinator for controlling access
- Process needs to get majority vote from m > n/2 coordinators to access the resource
- DHT-based implementation

Assumption

When a coordinator crashes, it will recover quickly, but will have forgotten about permissions it had granted.

A decentralized algorithm (Lin et al. 2004)

Observations

- Pros
  - Less vulnerable to coordinator failures
- Cons
  - Starvation, low efficiency

A comparison of the four algorithms

Summary of benefits and drawbacks

- n: # processes
- m: # coordinators
- k: # attempts

Algorithm	Messages per entry/exit	Delay before entry (in message times)	Problems
Centralized	3	2	Coordinator crash
Decentralized	3mk, k = 1,2,...	2 m	Starvation, low efficiency
Distributed	2 (n – 1)	2 (n – 1)	Crash of any process
Token ring	1 to ∞	0 to n – 1	Lost token, process crash

Election algorithms

Principle

An algorithm requires that some process acts as a coordinator. The question is how to select this special process dynamically.

Note

In many systems the coordinator is chosen by hand (e.g. file servers). This leads to centralized solutions → single point of failure.

Teasers

- If a coordinator is chosen dynamically, to what extent can we speak about a centralized or distributed solution?
- Is a fully distributed solution, i.e. one without a coordinator, always more robust than any centralized/coordinated solution?

Basic assumptions

Notion

- All processes have unique id’s
- All processes know id’s of all processes in the system (but not if they are up or down)
- Election means identifying the most suitable process based on different factors, e.g., highests id

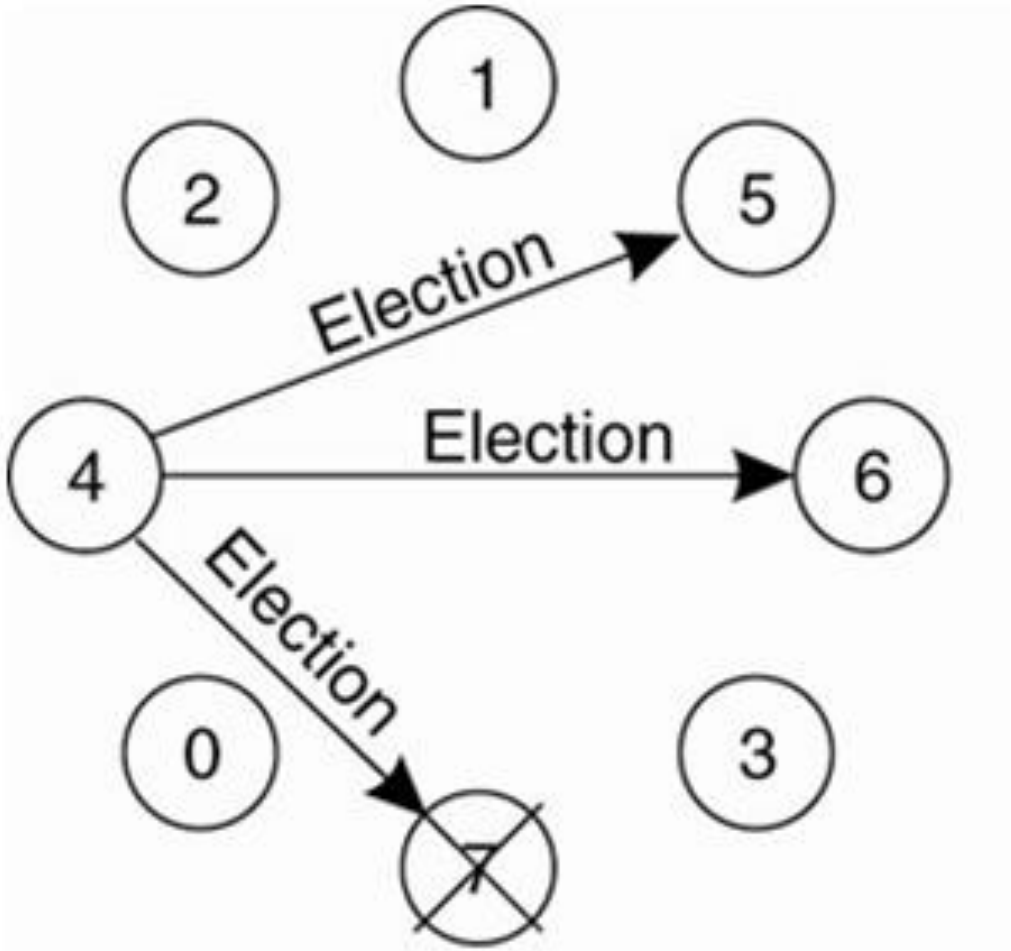
Election by bullying

Principle (Garcia-Molina 1982)

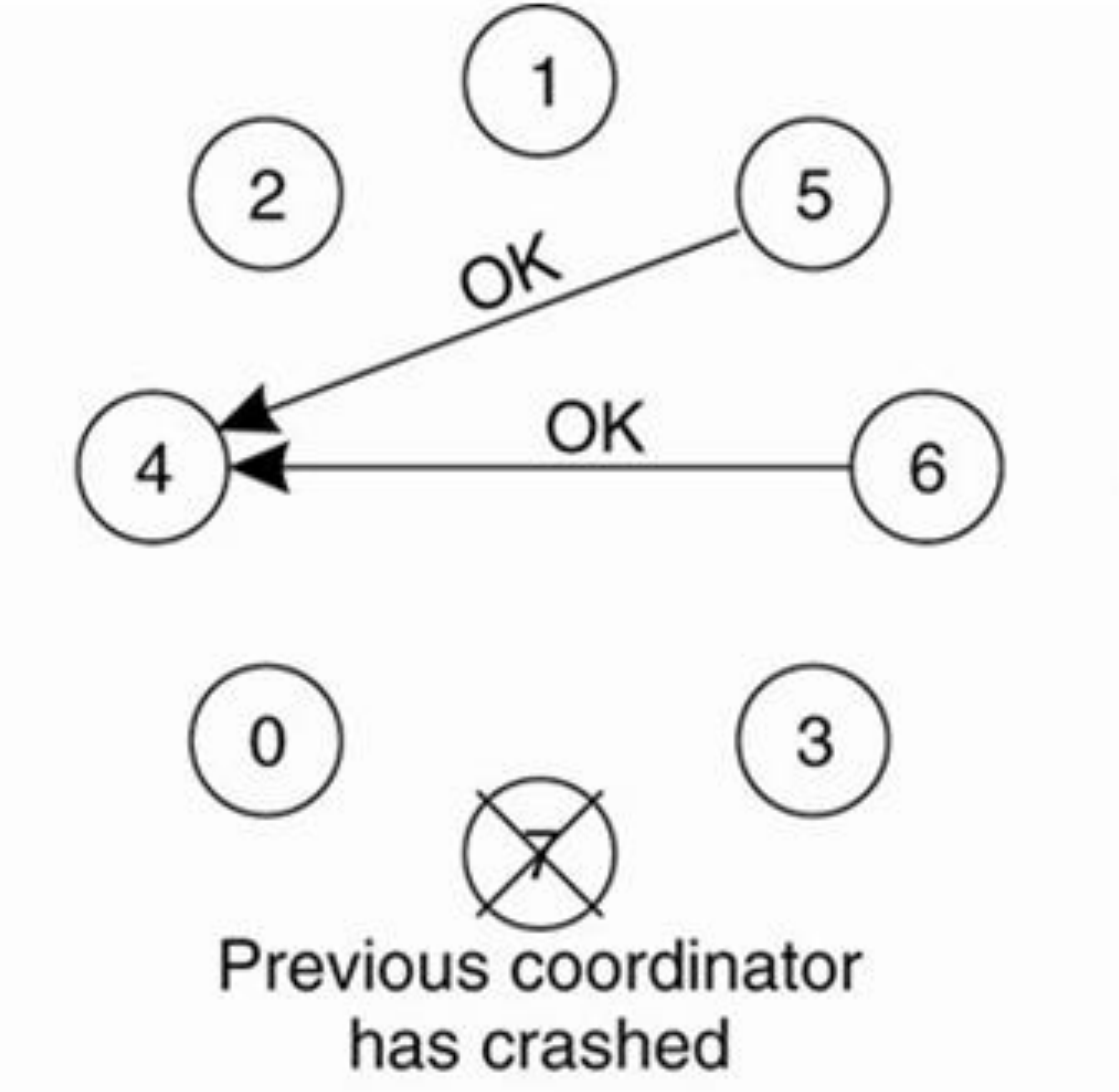
Consider N processes $\{P_0 \ldots P_{N-1}\}$ and let $id(P_k) = k$. When a process P_k notices that the coordinator is no longer responding to requests, it initiates an election:

1. P_k sends an ELECTION message to all processes with higher identifiers: $P_{k+1}; P_{k+2} \ldots P_{N-1}$.
2. If no one responds, P_k wins the election and becomes coordinator.
3. If one of the higher-ups answers, it takes over and P_k 's job is done.

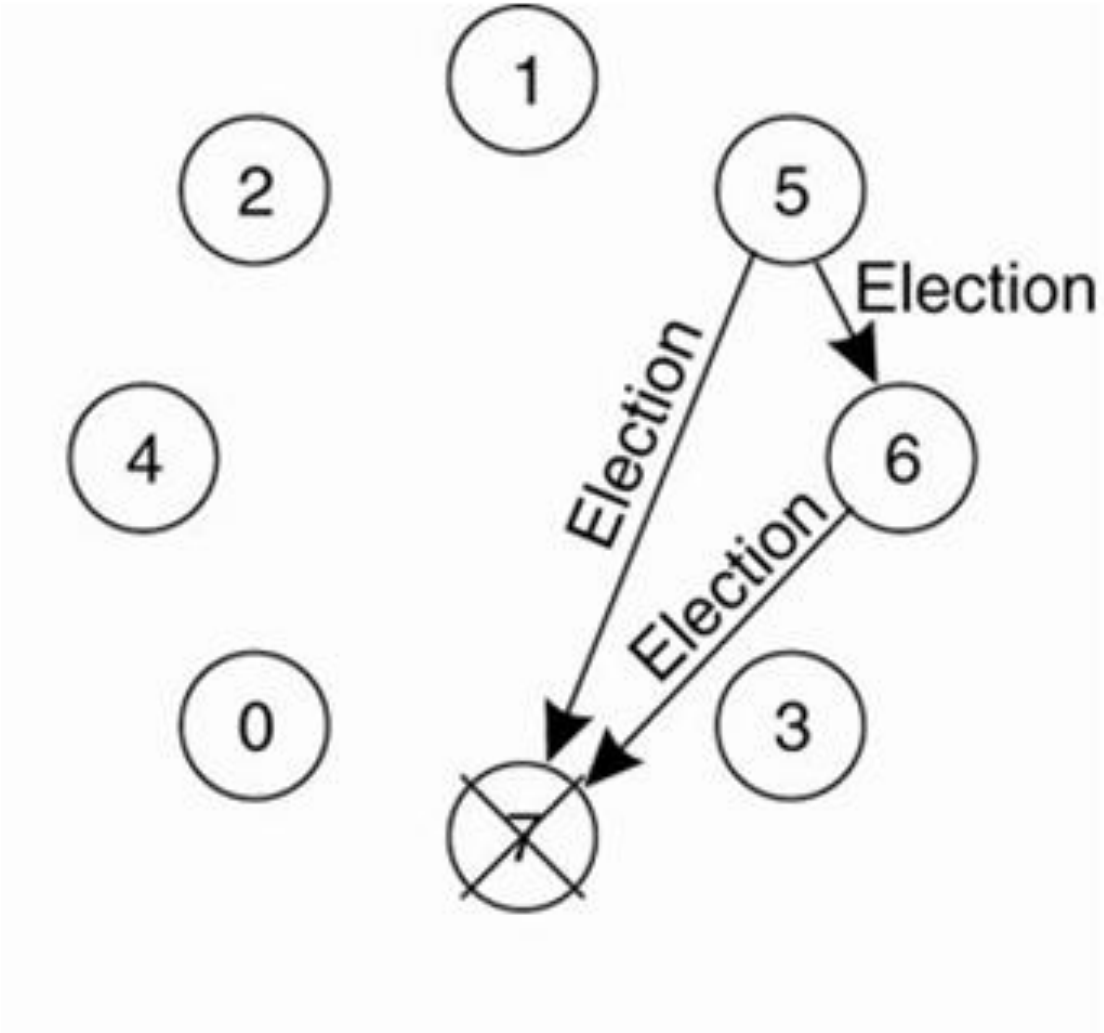
Election by bullying



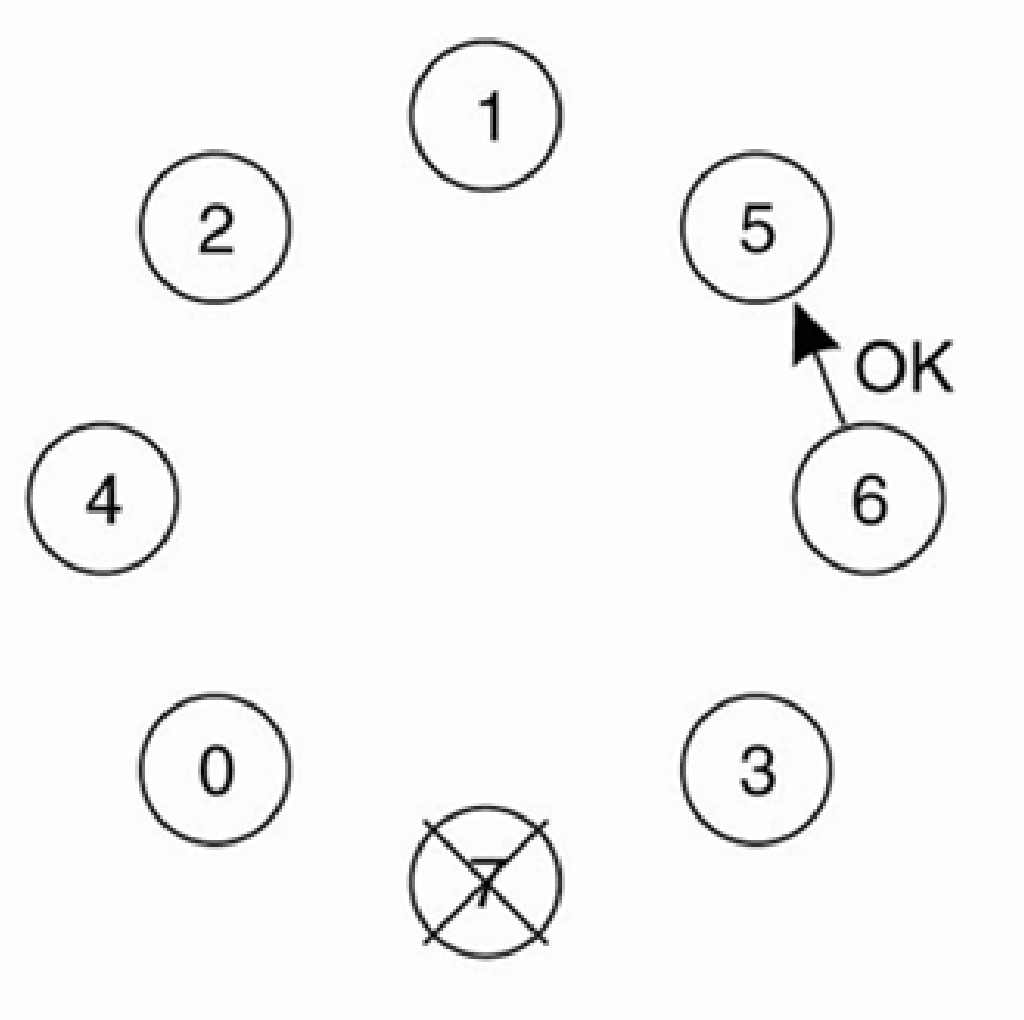
Election by bullying



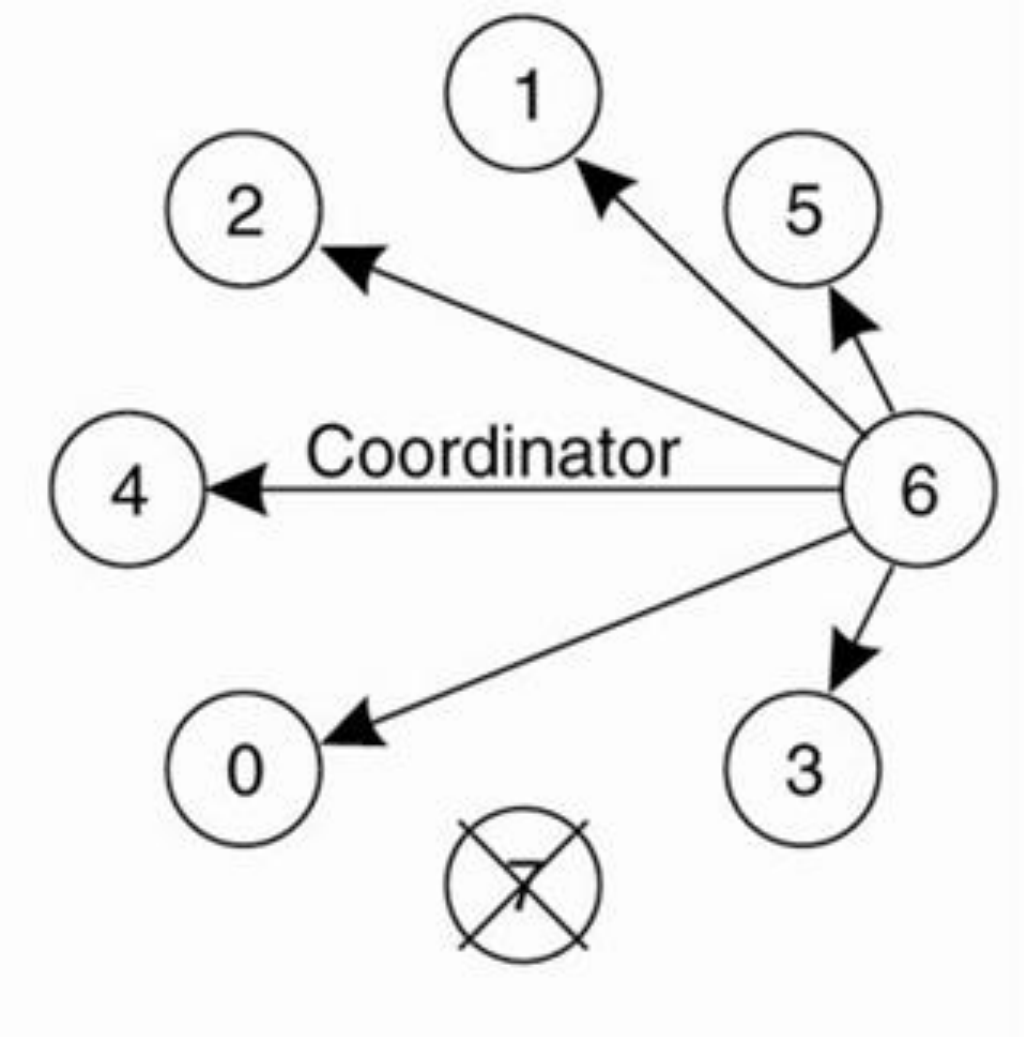
Election by bullying



Election by bullying

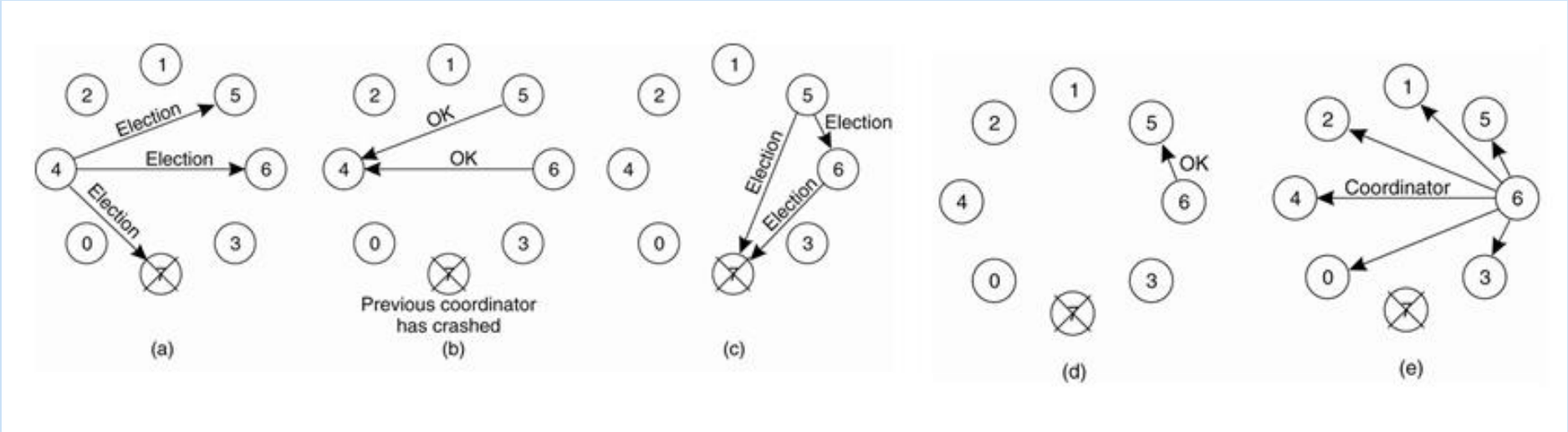


Election by bullying



Election by bullying

Summary



1. (a) Process 4 first notices that coordinator has crashed and sends ELECTION to processes with higher numbers 5,6, and 7
2. (b)-(d) Election proceeds, converging into process 6 winning
3. (e) By sending COORDINATOR message process 6 announces it is ready to take over
4. If process 7 is restarted, it will send COORDINATOR message to others and bully them into submission

Election in a ring

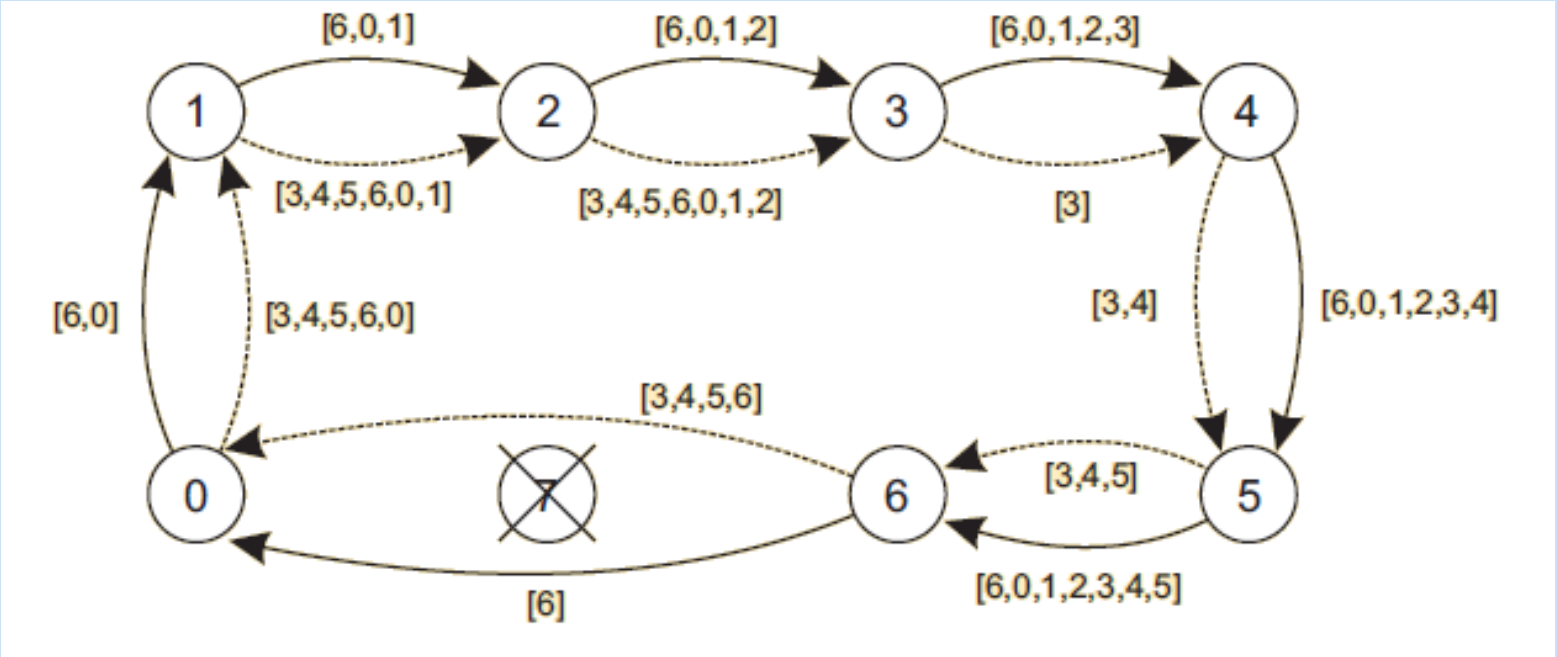
Principle

- Process priority is obtained by organizing processes into a (logical) ring.
- Process with the highest priority should be elected as coordinator.
  - Any process can start an election by sending an election message to its successor. If a successor is down, the message is passed on to the next successor.
  - If a message is passed on, the sender adds itself to the list. When it gets back to the initiator, everyone had a chance to make its presence known.
  - The initiator sends a coordinator message around the ring containing a list of all living processes. The one with the highest priority is elected as coordinator.

Election in a ring

Example: Election algorithm using a ring

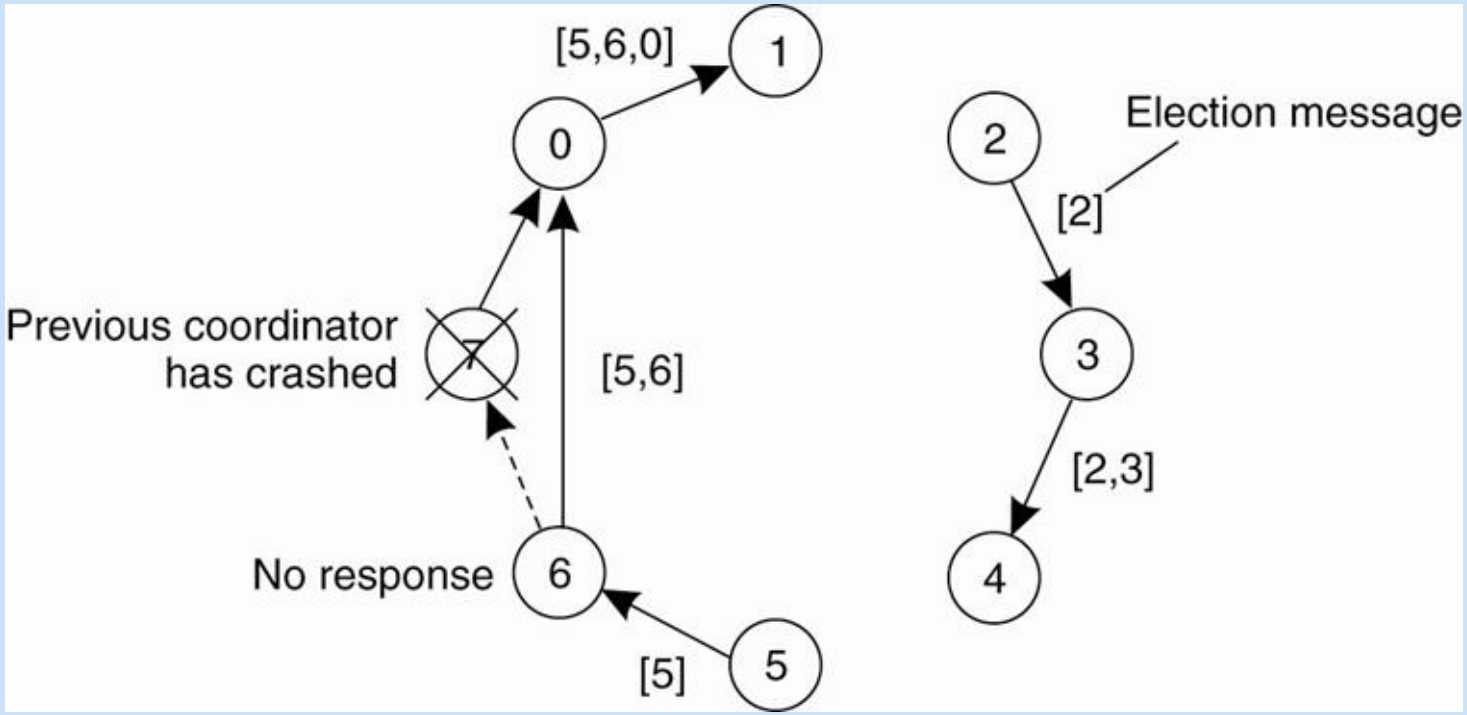
- The solid line shows the election messages initiated by P_6
- The dashed one the messages by P_3



Election in a ring

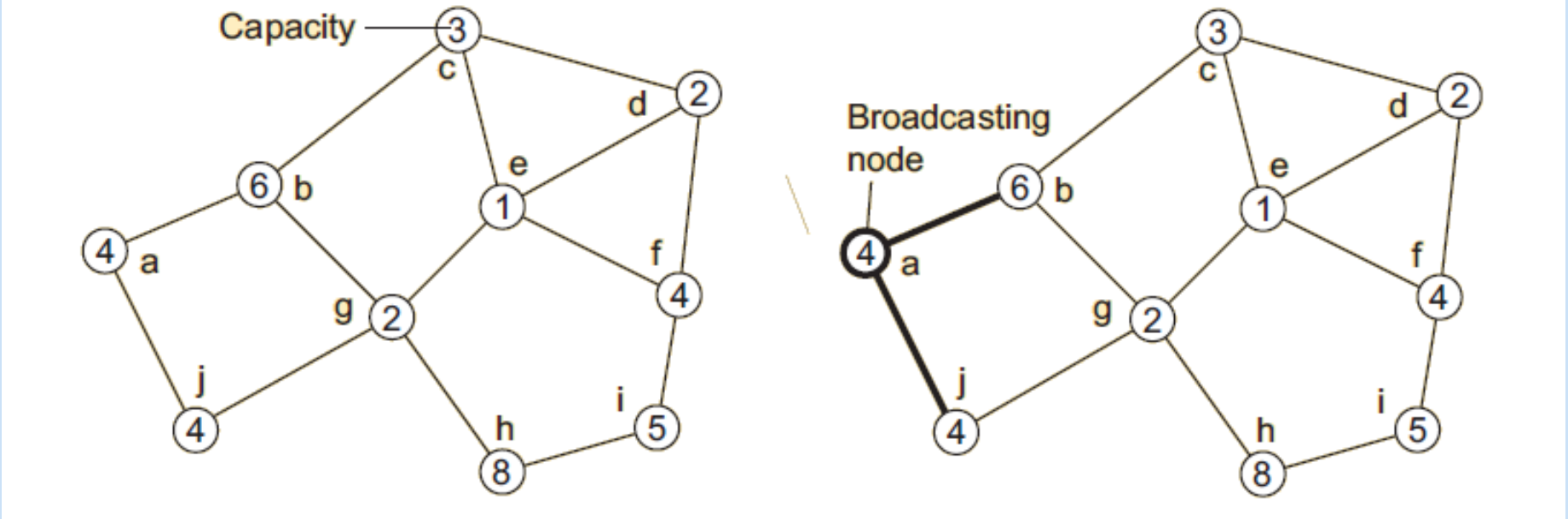
Example: Simultaneous election

(5 and 2 simultaneously in the example) notices the coordinator (7) is not functioning, it sends an ELECTION message containing its number and sends the message to its successor



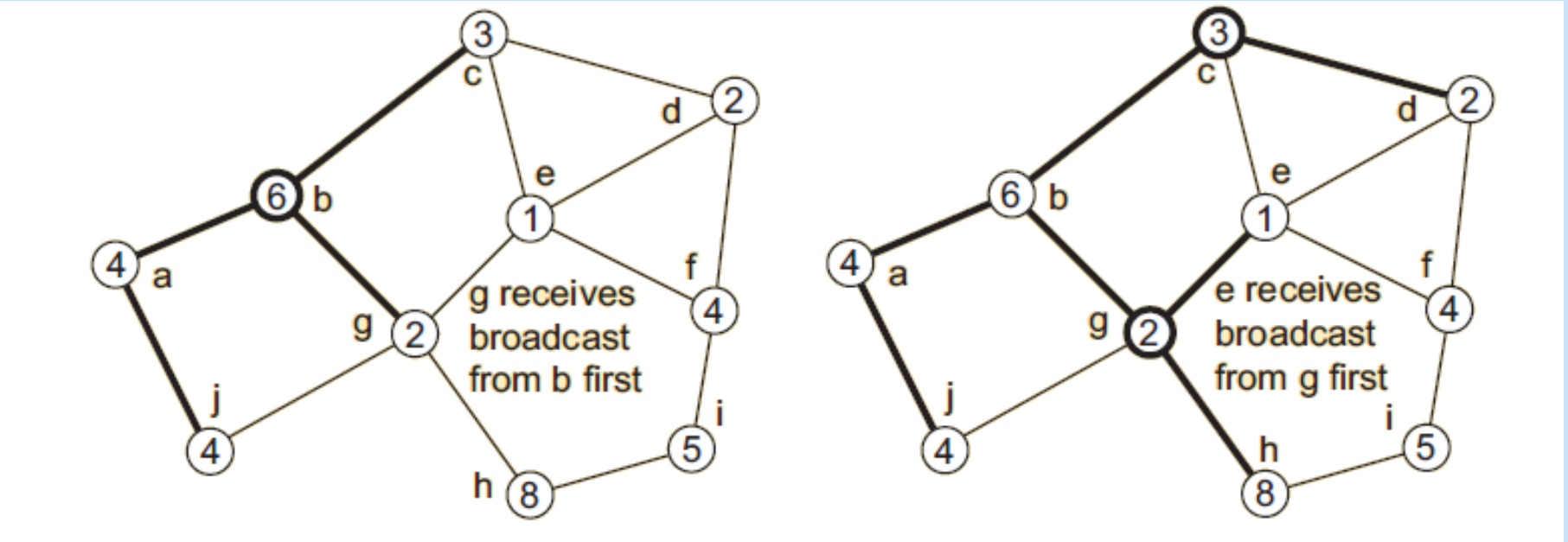
A solution for wireless networks

A sample network



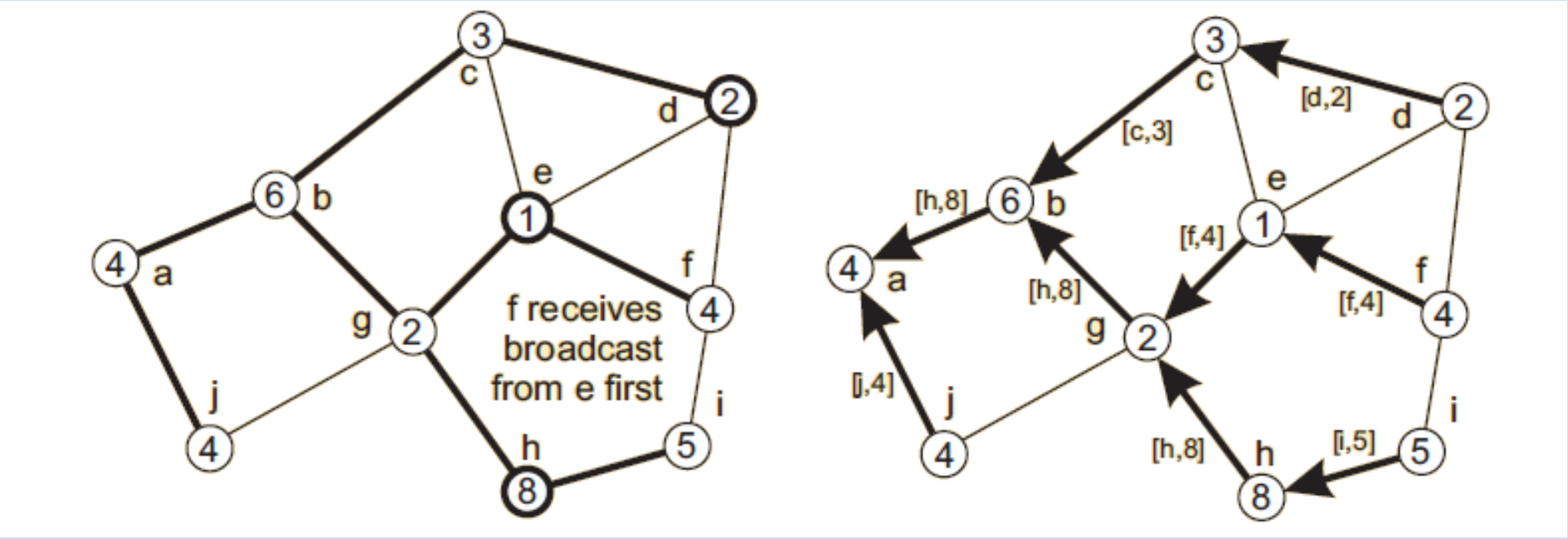
A solution for wireless networks

A sample network



A solution for wireless networks

A sample network



Election in large-scale systems

Observation

- Election algorithms (leader or coordinators) apply only to relatively small distributed systems
- Election algoritms find a single leader/coordinator

Problem

- How to select more than one node/process as leader/coordinator?, e.g., super-peers

Election in large-scale systems

Requirements to select super-peers

- Normal nodes should have low-latency access to super-peers
  - Proximal to each other
- Super-peers should be evenly distributed across the overlay network
- There should be predefined portion of super-peers relative to the total number of nodes in the overlay network
- Each super-peer should not need to server more than a fixed number of normal nodes.

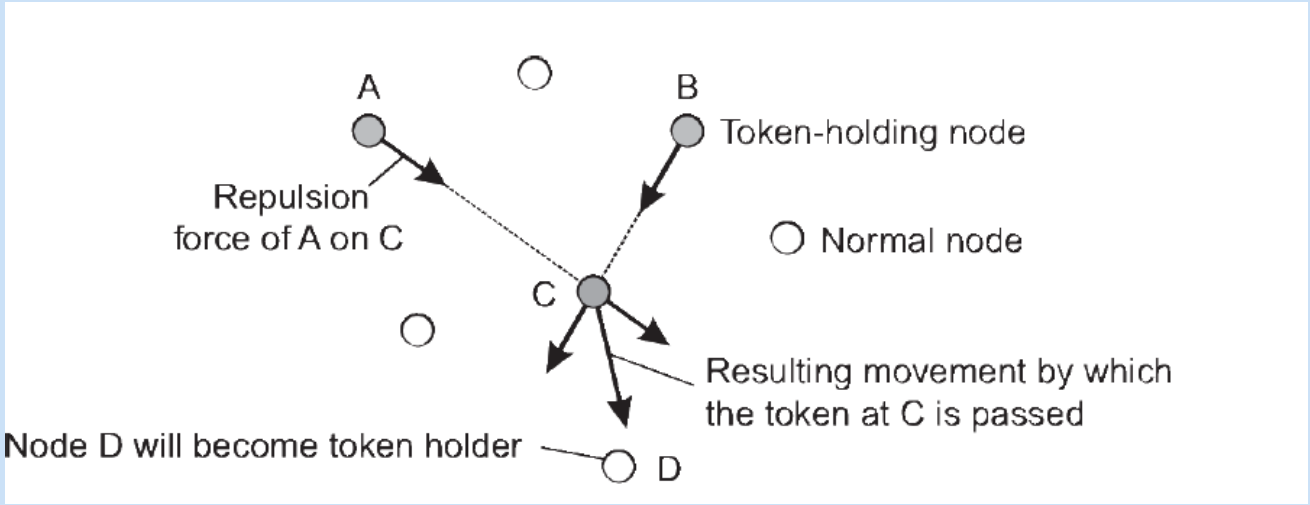
How to achieve this?

It depends on whether the overlay network is structured or unstructured, but a general approach to achieve this is, positioning

Election in large-scale systems

Super-peer election by positioning

- N nodes are located in a m-dimensional geometric space
- It is assume that super-peers need to be located evenly throughout the overlay network
- The basic idea is to introduce a total of N tokens in randomly chosen nodes.
  - No node can hold more than one token
  - Each node represents a repealing force



Location systems

Essence

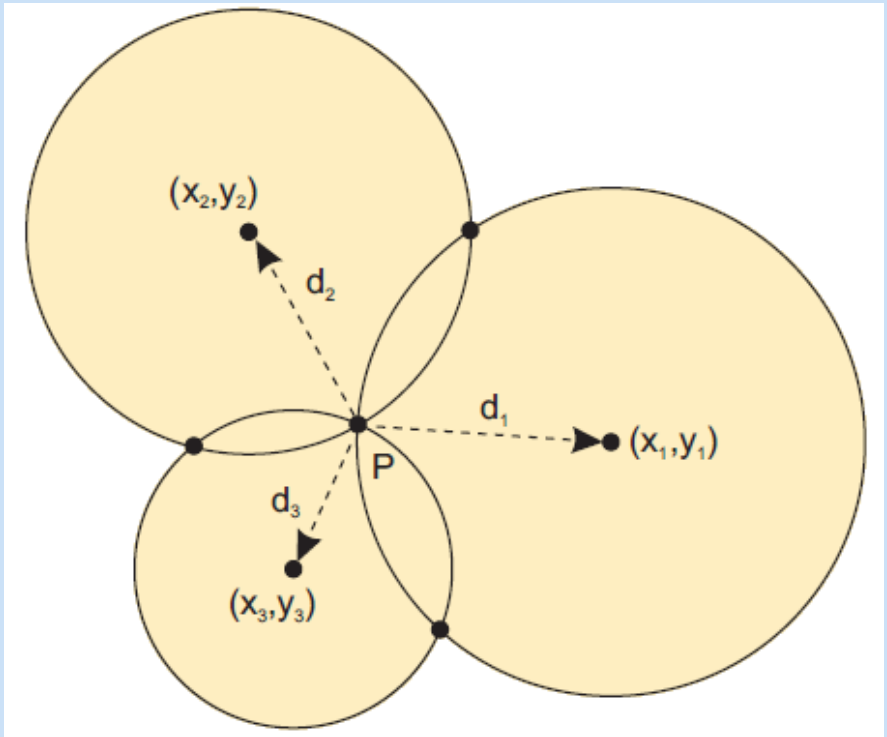
- Location (positioning) is important to improve communication performance, which also enhances coordination between processes/nodes
  - Nodes that are proximal can communicate with less cost rather than nodes that are very far apart
  - Nodes can also be selected as coordinator/leaders based on their location properties

Computing position

Observation

A node P needs d +1 landmarks to compute its own position in a d-dimensional space. Consider two-dimensional case.

Computing in a position in 2D



Solution

P needs to solve three equations in two unknowns (x_P,y_P):

$$d_i = \sqrt{(x_i - x_P)^2 + (y_i - y_P)^2}$$

Global positioning system (revisited)

Assuming that the clocks of the satellites are accurate and synchronized

- It takes a while before a signal reaches the receiver
- The receiver's clock is definitely out of synch with the satellite

Basics

Real distance

$$d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Global positioning system (revisited)

Assuming that the clocks of the satellites are accurate and synchronized

- It takes a while before a signal reaches the receiver
- The receiver's clock is definitely out of synch with the satellite

Basics

- Δ_r: unknown deviation of the receiver's clock.

Real distance

$$d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Global positioning system (revisited)

Assuming that the clocks of the satellites are accurate and synchronized

- It takes a while before a signal reaches the receiver
- The receiver's clock is definitely out of synch with the satellite

Basics

- Δ_r: unknown deviation of the receiver's clock.
- x_r , y_r , z_r: unknown coordinates of the receiver.

Real distance

$$d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Global positioning system (revisited)

Assuming that the clocks of the satellites are accurate and synchronized

- It takes a while before a signal reaches the receiver
- The receiver's clock is definitely out of synch with the satellite

Basics

- Δ_r: unknown deviation of the receiver's clock.
- x_r , y_r , z_r: unknown coordinates of the receiver.
- T_i : timestamp on a message from satellite i

Real distance

$$d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Global positioning system (revisited)

Assuming that the clocks of the satellites are accurate and synchronized

- It takes a while before a signal reaches the receiver
- The receiver's clock is definitely out of synch with the satellite

Basics

- Δ_r: unknown deviation of the receiver's clock.
- x_r , y_r , z_r: unknown coordinates of the receiver.
- T_i : timestamp on a message from satellite i
- $\mathbf{\Delta_i} = (T_{now} - T_i) + \Delta_r$: **measured delay** of the message sent by satellite i.

Real distance

$$d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Global positioning system (revisited)

Assuming that the clocks of the satellites are accurate and synchronized

- It takes a while before a signal reaches the receiver
- The receiver's clock is definitely out of synch with the satellite

Basics

- Δ_r: unknown deviation of the receiver's clock.
- x_r , y_r , z_r: unknown coordinates of the receiver.
- T_i : timestamp on a message from satellite i
- $\mathbf{\Delta_i} = (T_{now} - T_i) + \Delta_r$: **measured delay** of the message sent by satellite i.
- **Measured distance** to satellite **i: c x $\mathbf{\Delta_i}$** (c is speed of light)

Real distance

$$d_i = c\Delta_i - c\Delta_r = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

WiFi-based location services

Basic idea

- Assume we have a database of known access points (APs) with coordinates
- Assume we can estimate distance to an AP
- Then: with 3 detected access points, we can compute a position.

War driving: locating access points

- Use a WiFi-enabled device along with a GPS receiver, and move through an area while recording observed access points.
- Compute the centroid: assume an access point AP has been detected at N different locations $\{\vec{x}_1, \vec{x}_2, \ldots\ldots, \vec{x}_N\}$, with known GPS location.
- Compute location of AP as

$$\vec{x}_{AP} = \frac{\sum_{i=1}^{N} \vec{x}_i}{N}.$$

WiFi-based location services

Problems

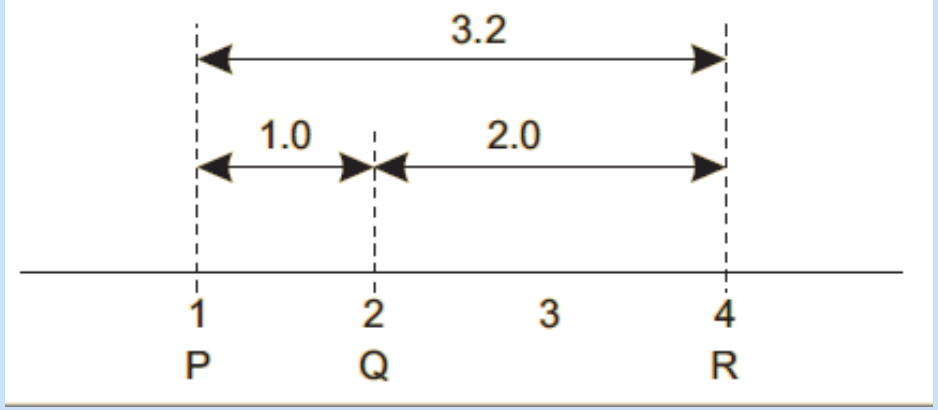
- Limited accuracy of each GPS detection point $\vec{x}$
- An access point has a non uniform transmission range
- Number of sampled detection points N may be too low.

Computing position

Problems

- Measured latencies to landmarks fluctuate
- Computed distances will not even be consistent

Inconsistent distances in 1D space



Computing position

Solution: minimize errors

- Use N special landmark nodes $L_1,\ldots,L_N$.
- Landmarks measure their pairwise latencies $\check{d}(L_i,L_j)$
- A central node computes the coordinates for each landmark, minimizing:

$$\sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(\frac{\tilde{d}(L_i,L_j)-\hat{d}(L_i,L_j)}{\tilde{d}(L_i,L_j)}\right)^2$$

where $\check{d}(L_i,L_j)$ is distance after nodes L_i and L_j have been positioned.

Computing position

Choosing the dimension m

The hidden parameter is the dimension m with N > m. A node P measures its distance to each of the N landmarks and computes its coordinates by minimizing

$$\sum_{i=1}^{N} \left(\frac{\tilde{d}(L_i, P) - \hat{d}(L_i, P)}{\tilde{d}(L_i, P)} \right)^2$$

Observation

Practice shows that m can be as small as 6 or 7 to achieve latency estimations within a factor 2 of the actual value.

Summary

What we learned?

- The principles of mutual exclusion
- The goal of election algorithms in overlay networks

Next lecture

Naming

Questions?

E-mail: huber.flores@ut.ee