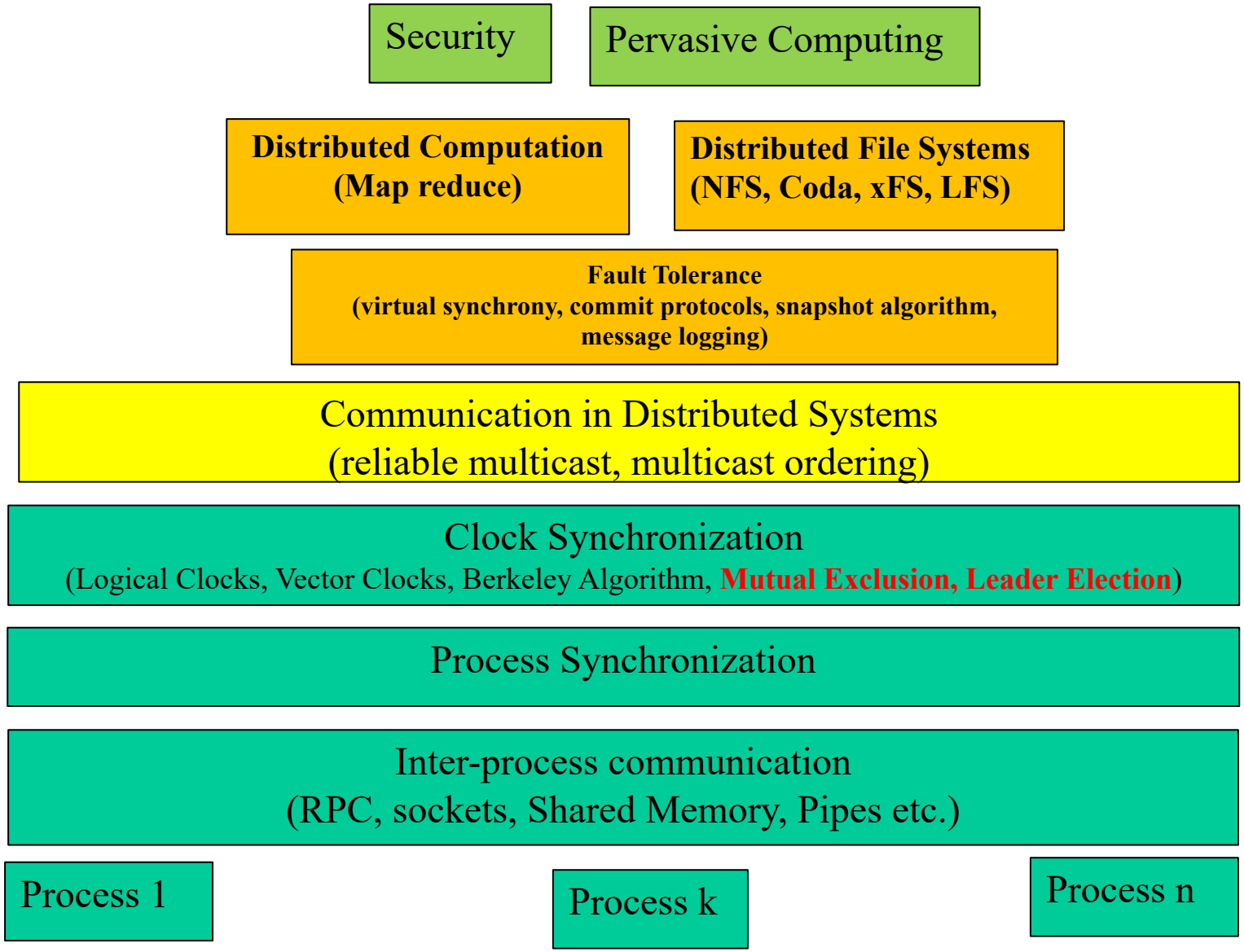
CMSC621: Advanced Operating Systems

Nilanjan Banerjee

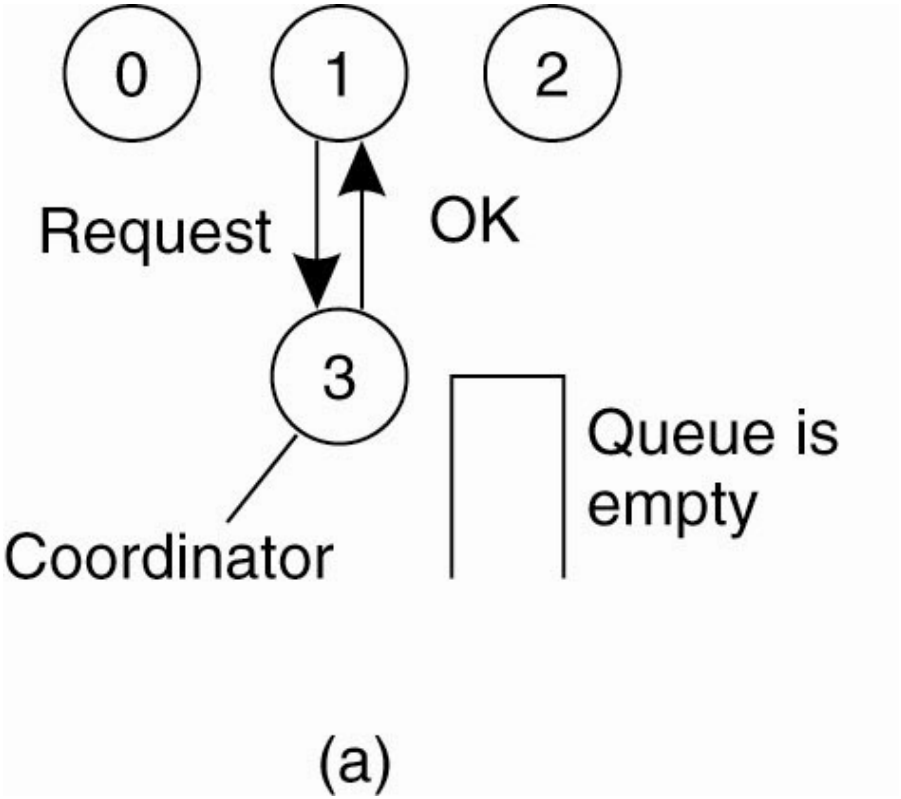
Professor, University of Maryland
Baltimore County
nilanb@umbc.edu
http://www.csee.umbc.edu/~nilanb/

Overview of the course



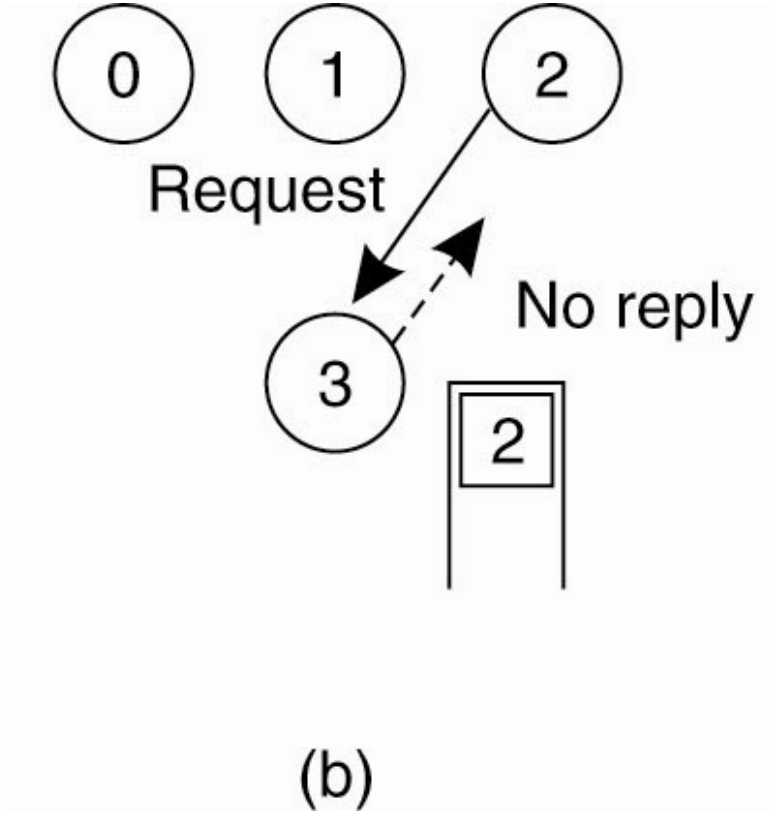
Take a look at some of the classic problems in Distributed systems.

Mutual Exclusion: A Centralized Algorithm (1)



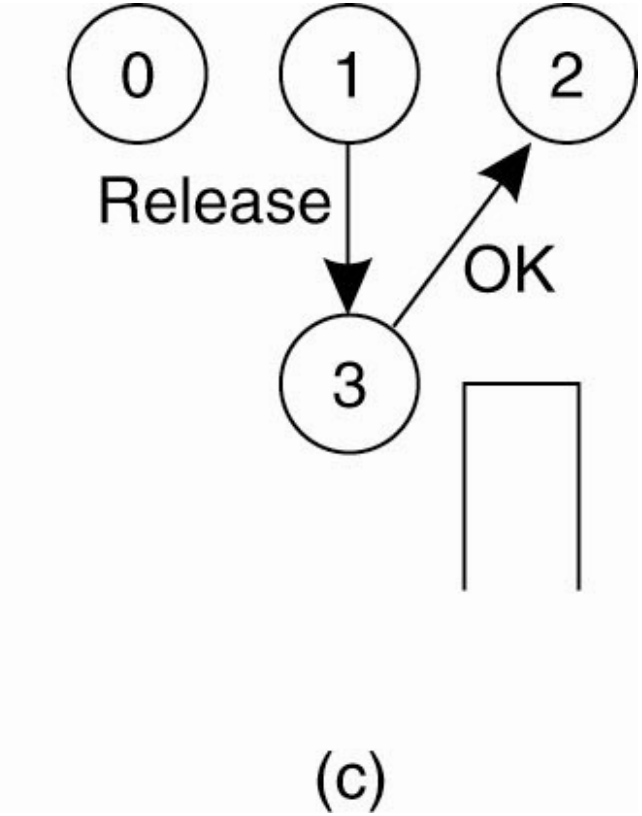
Process 1 asks the coordinator for permission to access a shared resource. Permission is granted.

Mutual Exclusion: A Centralized Algorithm (2)



Process 2 then asks permission to access the same resource. The coordinator does not reply.

Mutual Exclusion: A Centralized Algorithm (3)



When process 1 releases the resource, it tells the coordinator, which then replies to 2

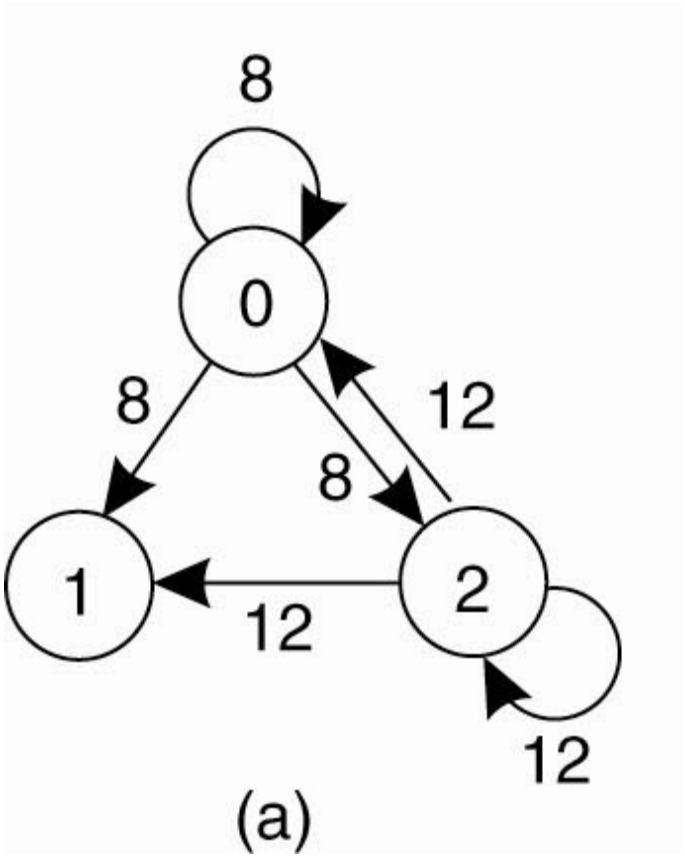
A Distributed Algorithm (1)

Three different cases:

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.

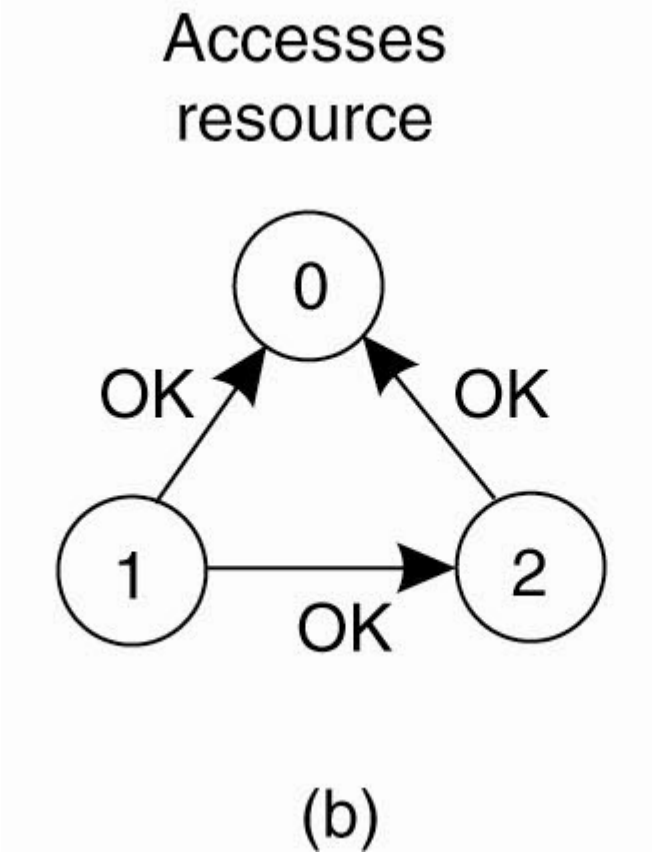
A Distributed Algorithm (2)

Two processes want to access a shared resource at the same moment.



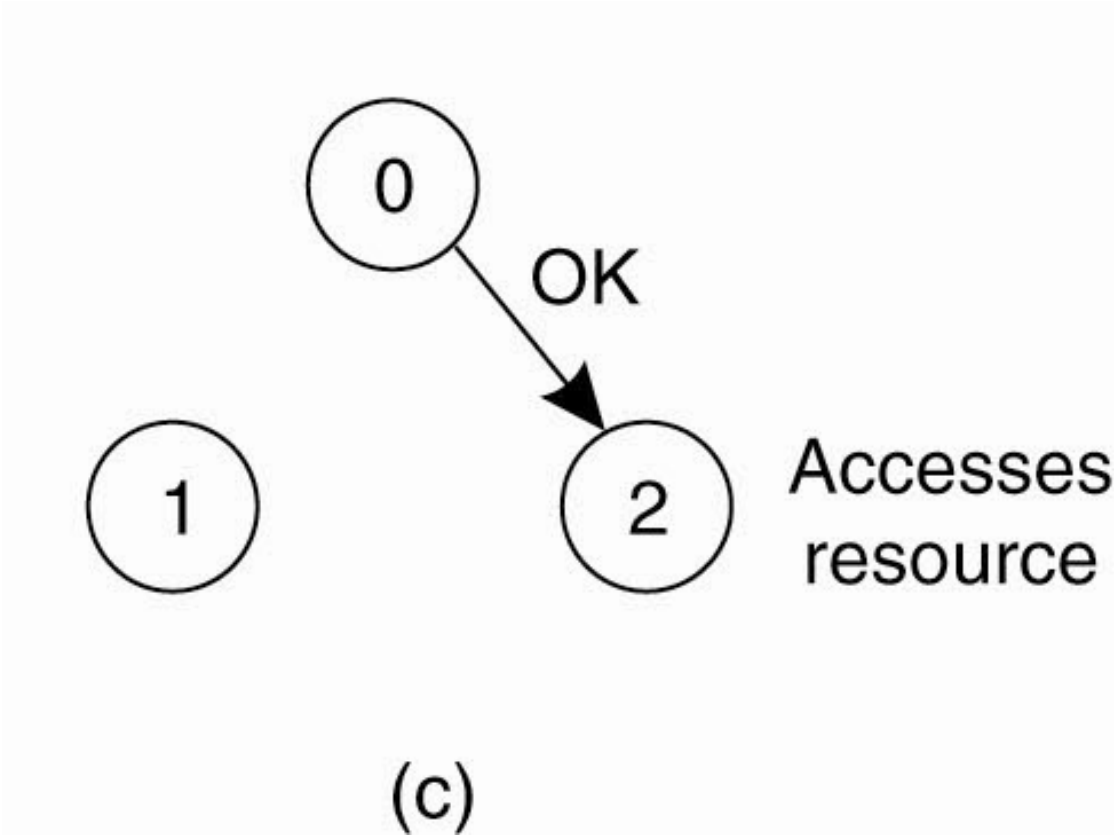
A Distributed Algorithm (3)

Process 0 has the lowest timestamp, so it wins.



A Distributed Algorithm (4)

When process 0 is done, it sends an OK also, so 2 can now go ahead.



Election Algorithms

Leader Election is the problem of finding a node in a distributed system that can act as a coordinator

Applications

- Electing a time server
- Centralized Synchronization
- Distributed Synchronization where you are a group of nodes and each group selects a leader.

Assumption: Bully Algorithm

Assumptions

Every process has a process id and the process Id is monotonically increasing

单调地

Every process knows the ID of the other processes in the network

A Plausible Criterion for Leader Election

possibly true

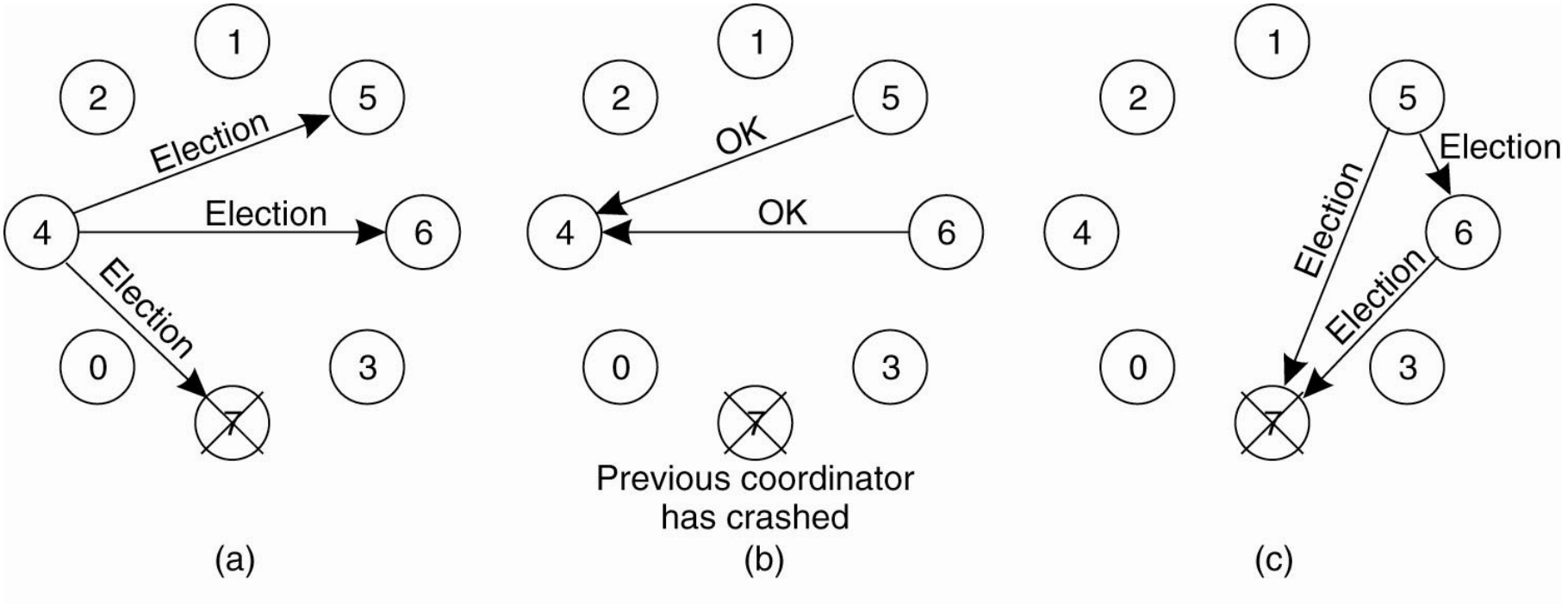
The largest ID node is the leader.

Election Algorithms

The Bully Algorithm

1. *P* sends an *ELECTION* message to all processes with higher numbers.
2. If no one responds, *P* wins the election and becomes coordinator.
3. If one of the higher-ups answers, it takes over. *P*'s job is done.

The Bully Algorithm (1)



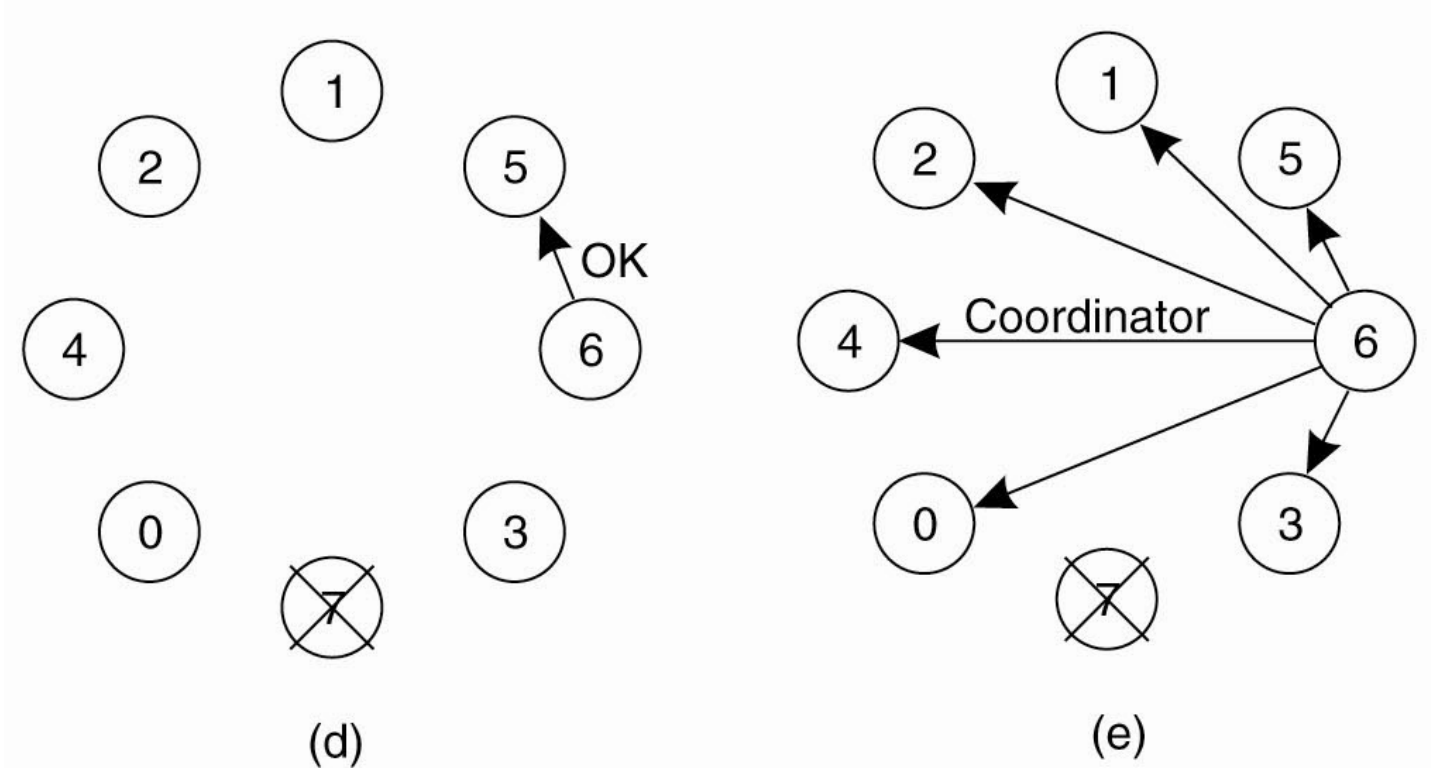
The bully election algorithm.

- Process 4 holds an election.
- Processes 5 and 6 respond, telling 4 to stop.
- Now 5 and 6 each hold an election.

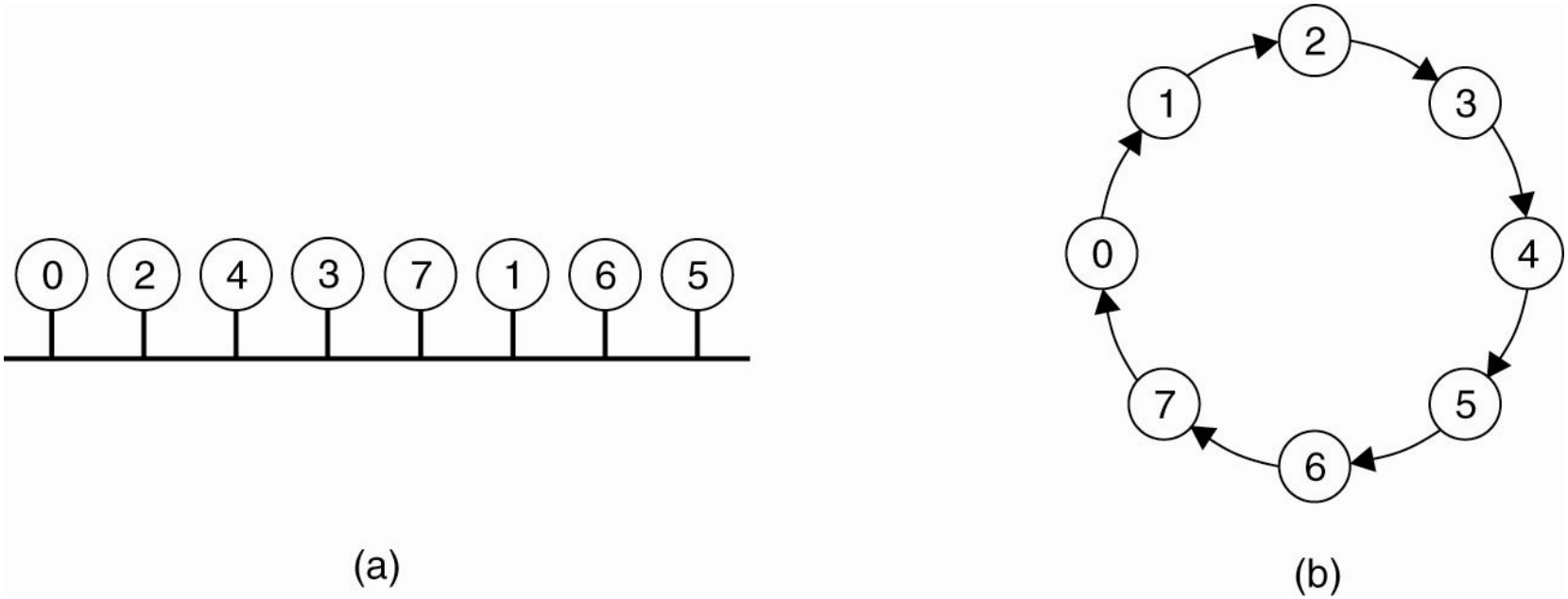
The Bully Algorithm (2)

The bully election algorithm.

- Process 6 tells 5 to stop.
- Process 6 wins and tells everyone.



Leader Election in a Token Ring



(a) An unordered group of processes on a network.
(b) A logical ring constructed in software.

O(n^2) Messages LE Algorithm

- send value of own *id* to the left
- when receive an id *j* (from the right):
  - if *j* > *id* then
    - forward *j* to the left (this processor has lost)
  - if *j* = *id* then
    - elect self (this processor has won)
  - if *j* < *id* then
    - do nothing

Analysis of $O(n^2)$ Algorithm

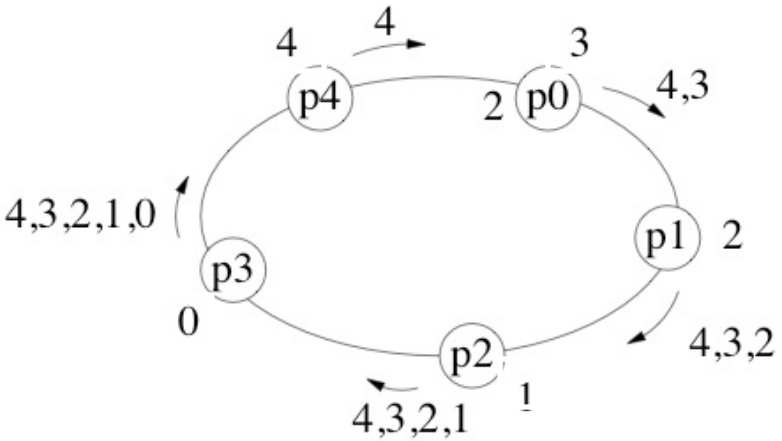
- **Correctness:** Elects processor with largest id.
  - msg containing largest id passes through every processor

- **Time:** $O(n)$

- **Message complexity:** Depends how the ids are arranged.
  - largest id travels all around the ring (n msgs)
  - 2nd largest id travels until reaching largest
  - 3rd largest id travels until reaching largest or second largest

Analysis of $O(n^2)$ Algorithm

- Worst way to arrange the ids is in decreasing order:
  - 2nd largest causes *n - 1* messages
  - 3rd largest causes *n - 2* messages
  - etc.
- Total number of messages is $n + (n-1) + (n-2) + \ldots + 1 = \Theta(n^2)$.



Can We Use Fewer Messages?

- The $O(n^2)$ algorithm is simple and works in both synchronous and asynchronous model.
- But can we solve the problem with fewer messages?
- Idea:
  - Try to have msgs containing smaller ids travel smaller distance in the ring

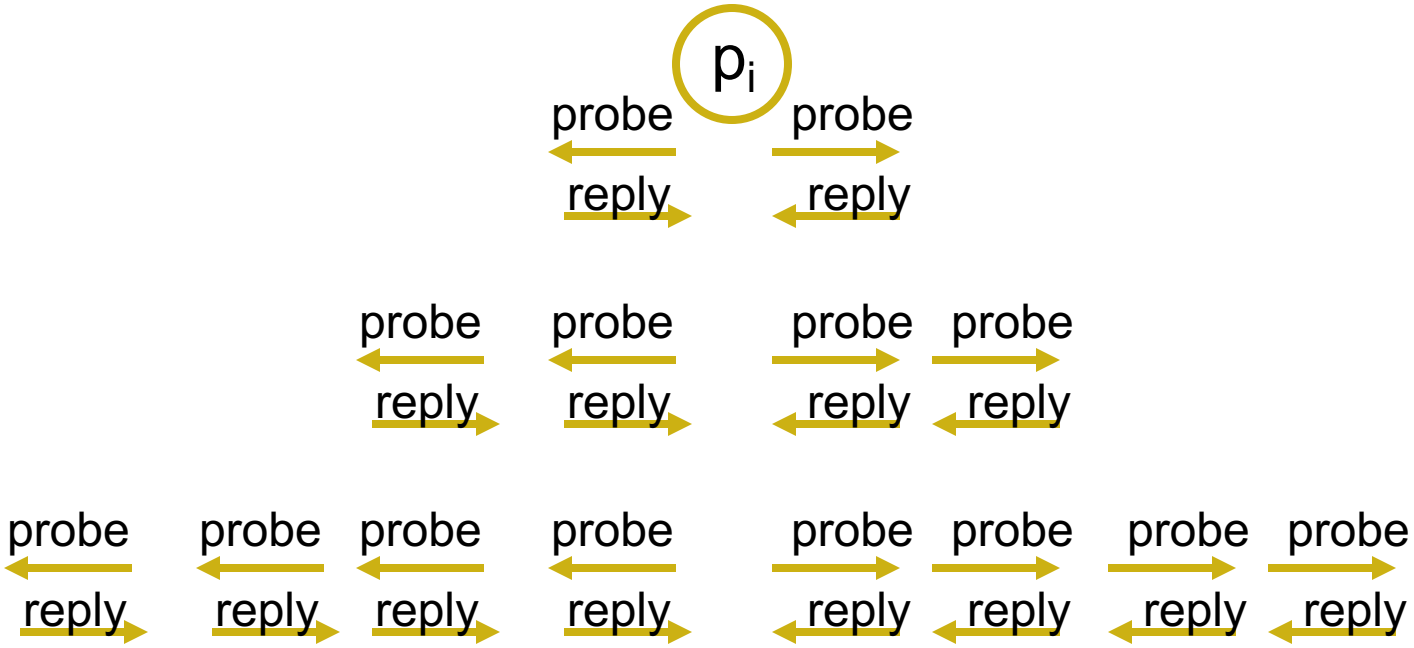
In-class Assignment

If someone can come up with a leader election algorithm (and analysis) in a token ring with a O(nlogn) message complexity, will get 3 extra credits.

O(n log *n)* Leader Election Algorithm

- Each proc. tries to probe successively larger neighborhoods in both directions
  - size of neighborhood *doubles* in each phase
- If probe reaches a node with a larger id, the probe stops
- If probe reaches end of its neighborhood, then a reply is sent back to initiator
- If initiator gets back replies from both directions, then go to next phase
- If proc. receives a probe with its own id, it elects itself

O(n log *n)* Leader Election Algorithm



Analysis of $O(n \log n)$ Leader Election Algorithm

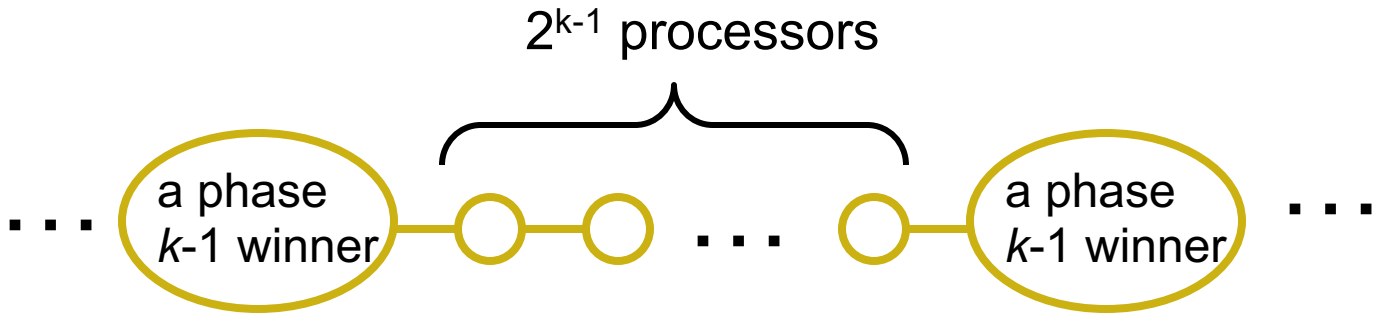
- **Correctness:** Similar to $O(n^2)$ algorithm.
- **Message Complexity:**
  - Each msg belongs to a particular phase and is initiated by a particular proc.
  - Probe distance in phase k is 2^k
  - Number of msgs initiated by a proc. in phase k is at most $4*2^k$ (probes and replies in both directions)

Analysis of $O(n \log n)$ Leader Election Algorithm

- How many procs. initiate probes in phase k ?
- For $k = 0$, every proc. does
- For $k > 0$, every proc. that is a "winner" in phase $k - 1$ does
  - "winner" means has largest id in its 2^{k-1} neighborhood

Analysis of *O(n* log *n)* Leader Election Algorithm

- Maximum number of phase *k - 1* winners occurs when they are packed as densely as possible:



- total number of phase *k - 1* winners is at most

$$n/(2^{k-1} + 1)$$

Analysis of $O(n \log n)$ Leader Election Algorithm

- How many phases are there?
- At each phase the number of (phase) winners is cut approx. in half
  - from $n/(2^{k-1} + 1)$ to $n/(2^k + 1)$
- So after approx. $\log_2 n$ phases, only one winner is left.
  - more precisely, max phase is $\lceil \log(n-1) \rceil + 1$

Analysis of $O(n \log n)$ Leader Election Algorithm

- Total number of messages is sum, over all phases, of number of winners at that phase times number of messages originated by that winner:

$$\leq 4n + n + \sum_{k=1}^{\lceil \log(n-1) \rceil + 1} 4 \cdot 2^k \cdot n/(2^{k-1}+1)$$

$$< 8n(\log n + 2) + 5n$$

$$= O(n \log n)$$

phase 0 msgs

termination msgs

msgs for phases 1 to $\lceil \log(n-1) \rceil + 1$

Can We Do Better?

- The $O(n \log n)$ algorithm is more complicated than the $O(n^2)$ algorithm but uses fewer messages in worst case.
- Works in both synchronous and asynchronous case.
- Can we reduce the number of messages even more?
- Not in the asynchronous model…