CSCI 5105

Instructor: Abhishek Chandra

Today

- Distributed Mutual Exclusion
- Leader Election

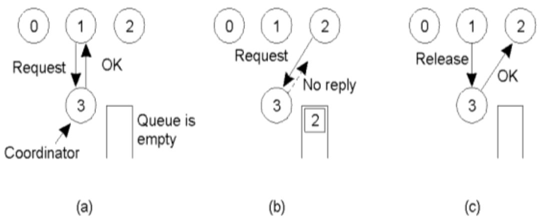
Distributed Mutual Exclusion

- Multiple processes on different machines may need to access a critical section
- Shared-memory systems:
  - Semaphores, mutexes, etc.
  - Typically implemented in shared memory
  - Processes share same blocking queues
- How to implement mutual exclusion in distributed systems?

Centralized Algorithm

- A coordinator grants access to critical section
  - Maintains a local queue
  - Can be elected using an election algorithm
- A process sends request to coordinator
  - If nobody in critical section, grant access
  - Otherwise, put process in queue
- When process done:
  - Send release to coordinator
  - Coordinator grants access to next process in queue

Centralized Algorithm



5

Centralized Algorithm: Properties

- Simple and efficient:
  - Requires only 3 messages per request grant
- No starvation or deadlock
- Problem:
  - What happens when coordinator crashes?

6

Decentralized Algorithm: Replicated Coordinator

- Have n replicas of the coordinator
  - A coordinator grants only one request at a time
- Need to get a majority m of permissions
  - Otherwise backoff and retry after random time
- Resource release:
  - Send release message to each of the m coordinators

7

Replicated Coordinator: Problems

- Problem 1: What if a coordinator fails and resets its state?
  - Problem only if a majority fail at the same time: What are the chances?
- Problem 2:What if there is a lot of resource contention?

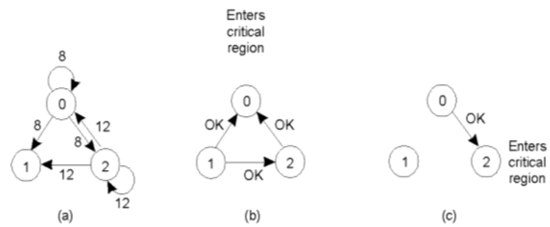
8

Distributed Algorithm: Timestamp-based Algorithm

- All events are totally ordered
- To gain access:
  - Send a request to all processes with timestamp
- On receipt of request:
  - If don't care, send OK
  - If already in critical section, queue the request
  - If want to enter the critical section, compare timestamp of request to own request: Send OK or queue based on timestamp value
- Access granted: When all processes send OK

9

Timestamp-based Algorithm



10

Timestamp-based Algorithm: Problems

- Requires 2(n-1) messages per access
- Any node becomes point of failure/bottleneck
  - Dependent on all nodes
  - Higher probability of failure than central algorithm
- Requires group communication
- Modifications:
  - Get permission from majority of processes
  - Get permission from overlapping subsets (~√n size)

11

Token Ring Algorithm

- Processes arranged in a ring
- Token passes around the ring
  - Token holder has access to critical section
- If process wants to enter critical section:
  - Wait for the token
  - Enter the critical section while holding the token
  - Pass on the token when done
- If process does not want to enter critical section:
  - Pass the token to neighbour

12

Token Ring Algorithm: Properties

- Fairness: Each process gets chance in turn
- Worst-case wait: O(n)
- Problems:
  - How to detect a lost token?
  - What if a process crashes?

13

Mutual Exclusion Algorithms: Comparison

Algorithm	Messages per entry/exit	Delay before entry (no. of messages)	Problems
Centralized	3	2	Coordinator crash
Decentralized	2mk+m, k=1,2,...	2mk	Starvation, inefficiency
Timestamp	2 (n – 1)	2 (n – 1)	Crash of any process
Token ring	1 to ∞	0 to n – 1	Lost token, Process crash

14

Leader Election

- Why do we need it?
  - Many systems require a coordinator, monitor, initiator, central server, etc.
  - It may not matter who the leader is
- Examples?

15

Election Algorithms

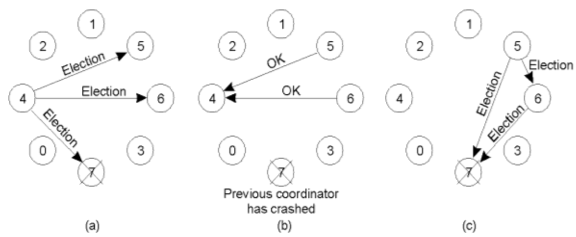
- Goal: All processes must agree on the leader after the election
- Choice of leader
  - Process with the highest ID
  - Process with desired properties, e.g.: resource capacity, location, etc.
- Question: How do we determine the leader?

16

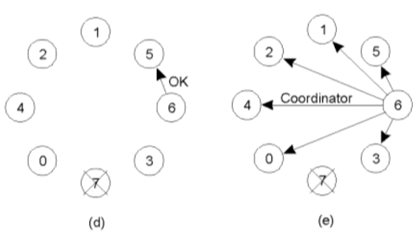
Bully Algorithm

- Process with highest ID "bullies" everyone into accepting it as a leader
- Initiation:
  - A process P sends ELECTION message to all processes with higher ID's
  - If no one responds, P wins the election
  - If someone responds, it takes over the election
- Last process remaining becomes the leader
  - Sends a Victory message to everyone

Bully Algorithm: Initiation



Bully Algorithm: Leader Election



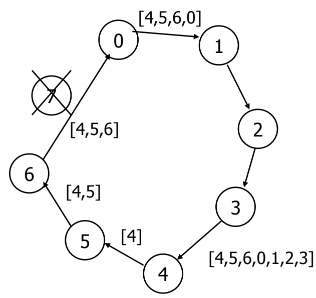
Bully Algorithm: Properties

- Assume n processes initially
- Worst Case:
  - Smallest process initiates election
  - Requires $O(n^2)$ messages
- Best Case:
  - Eventual leader initiates election
  - Requires (n-1) messages

Ring Algorithm

- Processes arranged in a ring
  - Each process has a successor
- Initiation:
  - A process sends an ELECTION message to its successor (or next alive process) with its ID
  - Each process adds its own ID and forwards the ELECTION message
- Leader Election:
  - Message comes back to initiator
  - Initiator announces the winner by sending another message around the ring

Ring Algorithm: Initiation



Ring Algorithm: Properties

- If only 1 process initiates election:
  - Requires 2n messages
- Two or more processes might simultaneously initiate elections
  - Still ensures election of the same leader
  - Results in extra messages

Election in Wireless Networks

- Restricted information
  - Nodes do not know everyones' identity
  - Overall topology may not be known
- Want "best" node to be leader
  - E.g.: most battery life, capacity, etc.

Election Tree

- Initiation:
  - One node starts election
  - Send ELECTION message to all neighbors
- On receiving ELECTION message:
  - If first message, assign sender as parent
  - Forward to all other neighbors
  - Otherwise, ACK to sender
- Responding to parent:
  - After getting ACKs from all neighbors
  - Also pass on info on "best" downstream node

25

Election in P2P Systems

- Electing Superpeers
- Goals:
  - Fixed proportion of total no. of nodes
  - Even distribution across the overlay networks
  - Load balanced
- Different solutions for:
  - DHT networks
  - Unstructured networks

26

Election in DHT Networks

- Goal: Reserve a fraction of the key space for superpeers
- Use top k-bits to identify superpeers
  - Superpeer for node p = Node responsible for p&1...10...0 (first k bits 1)
- No. of superpeers ≈ 2^{k-m} N
  - m-bit key space, N total nodes

27

Election in Unstructured Networks

- Goal: Place N superpeers evenly across an m-dimensional geometric space
- N tokens spread across N random nodes
- Each token exerts a repelling force
  - Tokens move away from each other based on the net force
- Gossipping used to spread the forces through the network
  - If the force on a token > threshold, move it away
- Superpeer: node that manages to hold a token for a certain time duration

28