分布式系统

06-分布式系统的互斥/选举 Mutual Exclusion/Selection in Distributed Systems

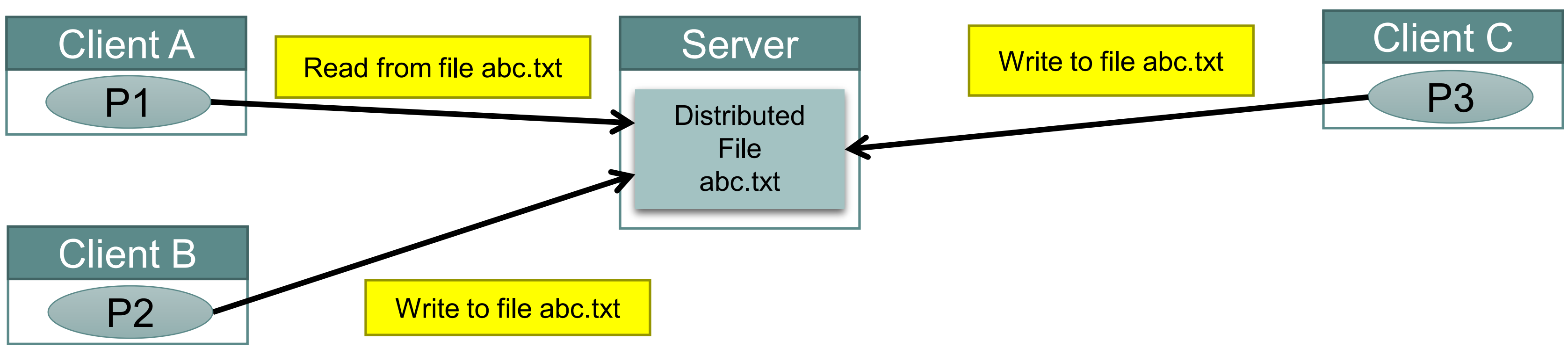
Weixiong Rao 饶卫雄
Tongji University 同济大学软件学院
2023 秋季
wxrao@tongji.edu.cn

Overview

- Time Synchronization
  - Clock Synchronization
  - Logical Clock Synchronization
- Mutual Exclusion
- Election Algorithms

Need for Mutual Exclusion

- Distributed processes need to coordinate to access shared resources
- Example: Writing a file in a Distributed File System



In uniprocessor systems, mutual exclusion to a shared resource is provided through shared variables or operating system support.

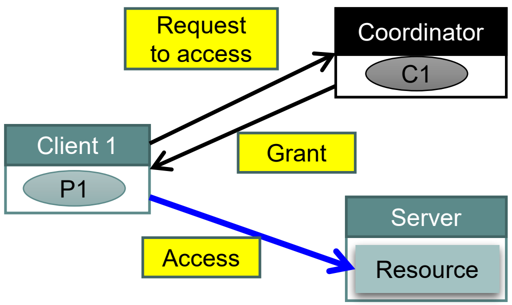
However, such support is insufficient to enable mutual exclusion of distributed entities

In Distributed System, processes coordinate access to a shared resource by passing messages to enforce *distributed mutual exclusion*

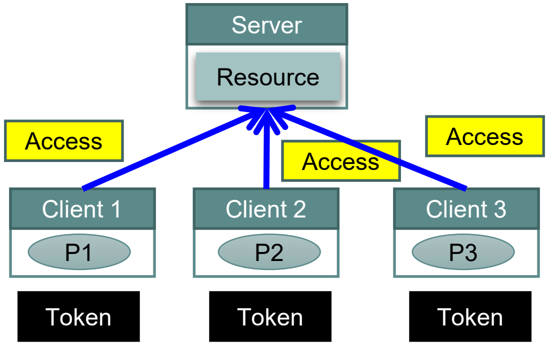
Types of Distributed Mutual Exclusion

- Mutual exclusion algorithms are classified into two categories

1. Permission-based Approaches
   - A process, which wants to access a shared resource, requests the permission from one or more coordinators



2. Token-based Approaches
   - Each shared resource has a token
   - Token is circulated among all the processes
   - A process can access the resource if it has the token



Overview

- Time Synchronization
  - Clock Synchronization
  - Logical Clock Synchronization
- Mutual Exclusion
  - Permission-based Approaches
  - Token-based Approaches
- Election Algorithms

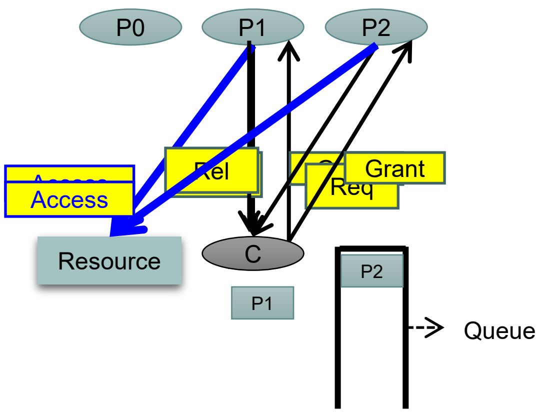
Permission-based Approaches

- There are two types of permission-based mutual exclusion algorithms
  a. Centralized Algorithms
  b. Decentralized Algorithms

- We will study an example of each type of algorithm

a. A Centralized Algorithm

- One process is elected as a coordinator (C) for a shared resource
- Coordinator maintains a **`Queue`** of access requests
- Whenever a process wants to access the resource, it sends a request message to the coordinator to access the resource
- When the coordinator receives the request:
  - If no other process is currently accessing the resource, it grants the permission to the process by sending a "grant" message
  - If another process is accessing the resource, the coordinator queues the request, and does not reply to the request
- The process releases the exclusive access after accessing the resource
- The coordinator will then send the "grant" message to the next process in the queue



Discussion about Centralized Algorithm

- Blocking vs. non-blocking requests
  - The coordinator can block the requesting process until the resource is free
  - Otherwise, the coordinator can send a “permission-denied” message back to the process
    - The process can poll the coordinator at a later time, or
    - The coordinator queues the request. Once the resource is released, the coordinator will send an explicit “grant” message to the process

- The algorithm guarantees mutual exclusion, and is simple to implement

- Fault-tolerance:
  - Centralized algorithm is vulnerable to a single-point of failure (at coordinator)
    - Processes cannot distinguish between dead coordinator and request blocking

- Performance bottle-neck:
  - In a large system, single coordinator can be overwhelmed with requests

b. A Decentralized Algorithm

- To avoid the drawbacks of the centralized algorithm, Lin *et al.* [1] advocated a decentralized mutual exclusion algorithm

- Assumptions
  - Distributed processes are in a Distributed Hash Table (DHT) based system
  - Each resource is replicated **`n`** times
    - The **`i`**th replica of a resource **`rname`** is named as **`rname-i`**
  - Every replica has its own coordinator for controlling access
    - The coordinator for **`rname-i`** is determined by using a hash function

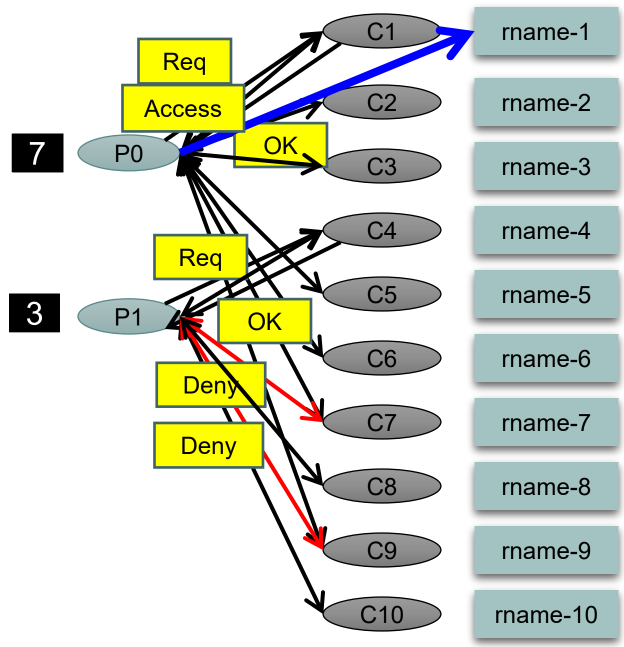
- Approach:
  - Whenever a process wants to access the resource, it will have to get a **majority vote** from **`m > n/2`** coordinators
  - If a coordinator does not want to vote for a process (because it has already voted for another process), it will send a "permission-denied" message to the process

[1] Shi-Ding Lin, Qiao Lian, Ming Chen and Zheng Zhang, "A Practical Distributed Mutual Exclusion Protocol in Dynamic Peer-to-Peer Systems", Lecture Notes in Computer Science, 2005, Volume 3279/2005, 11-21, DOI: 10.1007/978-3-540-30183-7_2

A Decentralized Algorithm – An Example

- If **n=10** and **m=7**, then a process needs at-least **7** votes to access the resource

majority vote



rname-x = xth replica of a resource rname; Cj = Coordinator j; Pi = Process i; n = Number of votes gained

Fault-tolerance in Decentralized Algorithm

- The decentralized algorithm assumes that the coordinator recovers quickly from a failure
- However, the coordinator would have reset its state after recovery
  - Coordinator could have forgotten any vote it had given earlier
- Hence, the coordinator may incorrectly grant permission to the processes
  - Mutual exclusion cannot be deterministically guaranteed
  - But, the algorithm *probabilistically* guarantees mutual exclusion

Probabilistic Guarantees in Decentralized Alg.

- What is the minimum number of coordinators who should fail for violating mutual exclusion?
  - At least $2m$-n coordinators should fail
- Let the probability of violating mutual exclusion be P_v
- Derivation of P_v
  - Let T be the lifetime of the coordinator
  - Let $p=\Delta t/T$ be the probability that coordinator crashes during time-interval Δt
  - Let $P[k]$ be the probability that k out of m coordinators crash during the same interval

$$P[k] = \binom{m}{k} p^k \, p^{m-k}$$

  - We compute the mutual exclusion violation probability P_v by:

$$P_v = \sum_{k=2m-n}^{n} P[k]$$

- In practice, this probability should be very small
  - For T=3 hours, Δt=10 s, n=32, and m=0.75n : P_v=10^{-40}

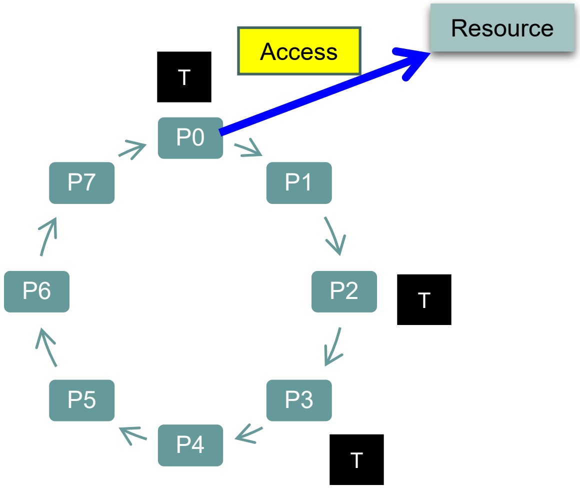
Overview

- Time Synchronization
  - Clock Synchronization
  - Logical Clock Synchronization
- Mutual Exclusion
  - Permission-based Approaches
  - Token-based Approaches
- Election Algorithms

Token Ring

- In the Token Ring algorithm, each resource is associated with a *token*
- The token is circulated among the processes
- The process with the token can access the resource

- Circulating the token among processes:
  - All processes form a logical ring where each process knows its next process
  - One process is given a *token* to access the resource
  - The process with the token has the right to access the resource
  - If the process has finished accessing the resource OR does not want to access the resource:
    - it passes the token to the next process in the ring



Discussion about Token Ring

- ✓ Token ring approach provides deterministic mutual exclusion
  - There is one token, and the resource cannot be accessed without a token
- ✓ Token ring approach avoids starvation
  - Each process will receive the token
- ✗ Token ring has a high-message overhead
  - When no processes need the resource, the token circulates at a high-speed
- ✗ If the token is lost, it must be regenerated
  - Detecting the loss of token is difficult since the amount of time between successive appearances of the token is unbounded
- ✗ Dead processes must be purged from the ring
  - ACK based token delivery can assist in purging dead processes

Comparison of Mutual Exclusion Algorithms

Algorithm	Delay before a process can access the resource (in message times)	Number of messages required for a process to access and release the shared resource	Problems
Centralized	2	3	• Coordinator crashes
Decentralized	2mk	2mk + m; k=1,2,…	• Large number of messages
Token Ring	0 to (n-1)	1 to ∞	• Token may be lost • Ring can cease to exist since processes crash

- Assume that:

  n = Number of processes in the distributed system

  For the Decentralized algorithm:

  m = minimum number of coordinators who have to agree for a process to access a resource

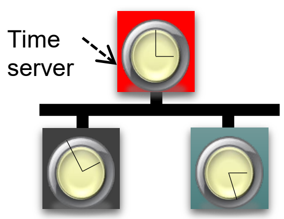
  k = average number of requests made by the process to a coordinator to request for a vote

Overview

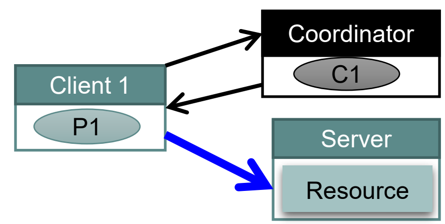
- Time Synchronization
  - Clock Synchronization
  - Logical Clock Synchronization
- Mutual Exclusion
  - Permission-based Approaches
  - Token-based Approaches
- Election Algorithms

Election in Distributed Systems

- Many distributed algorithms require one process to act as a coordinator
  - Typically, it does not matter which process is elected as the coordinator
- Example algorithms where coordinator election is required



Berkeley Clock Synchronization Algorithm



A Centralized Mutual Exclusion Algorithm

Election Process

- Any process P_i in the DS can initiate the election algorithm that elects a new coordinator
- At the termination of the election algorithm, the elected coordinator process should be unique
- Every process *may* know the process ID of every other processes, but it does not know which processes have crashed
- Generally, we require that the coordinator is the process with the largest process ID
  - The idea can be extended to elect *best* coordinator
    - Example: Election of a coordinator with least computational load
      - If the computational load of process P_i denoted by $load_i$, then coordinator is the process with highest $1/load_i$. Ties are broken by sorting process ID.

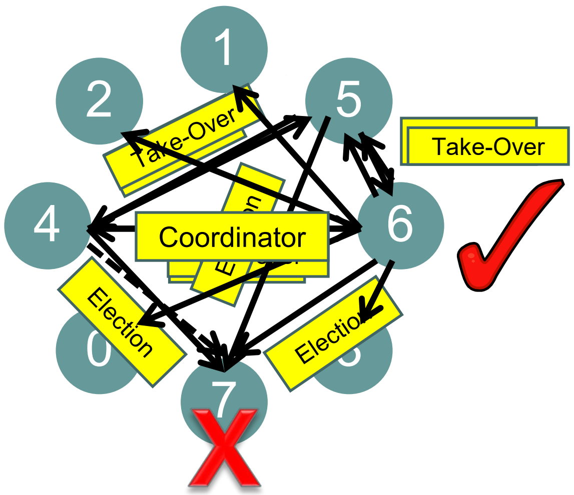
Election Algorithms

- We will study two election algorithms
  1. Bully Algorithm
  2. Ring Algorithm

1. Bully Algorithm

- A process initiates election algorithm when it notices that the existing coordinator is not responding
- Process P_i calls for an election as follows:

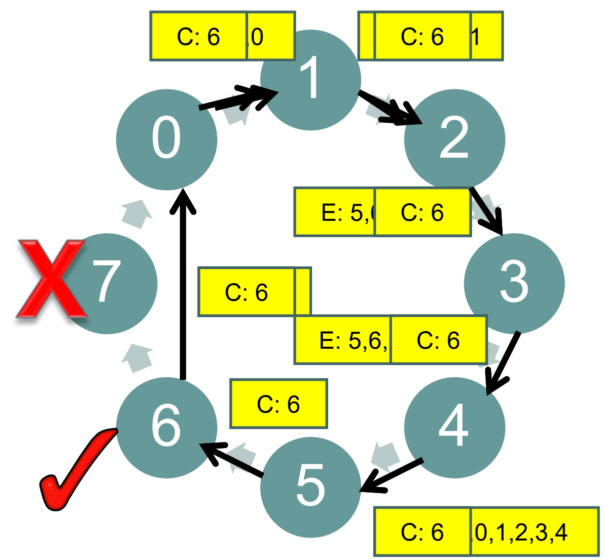
1. P_i sends an "Election" message to all processes with higher process IDs
2. When process P_j with $j>i$ receives the message, it responds with a "Take-over" message. P_j no more contests in the election
   i. Process P_j re-initiates another call for election. Steps 1 and 2 continue
3. If no one responds, P_i wins the election. P_i sends "Coordinator" message to every process



2. Ring Algorithm

- This algorithm is generally used in a ring topology
- When a process P_i detects that the coordinator has crashed, it initiates an election algorithm

1. P_i builds an "Election" message (E), and sends it to its next node. It inserts its ID into the Election message
2. When process P_j receives the message, it appends its ID and forwards the message
   i. If the next node has crashed, P_j finds the next alive node
3. When the message gets back to the process that started the election:
   i. it elects process with highest ID as coordinator, and
   ii. changes the message type to "Coordination" message (C) and circulates it in the ring



Comparison of Election Algorithms

Algorithm	Number of Messages for Electing a Coordinator	Problems
Bully Algorithm	$O(n^2)$	• Large message overhead
Ring Algorithm	2n	• An overlay ring topology is necessary

- Assume that:

  n = Number of processes in the distributed system

Summary of Election Algorithms

- Election algorithms are used for choosing a unique process that will coordinate an activity
- At the end of the election algorithm, all nodes should uniquely identify the coordinator
- We studied two algorithms for election
  - Bully algorithm
    - Processes communicate in a distributed manner to elect a coordinator
  - Ring algorithm
    - Processes in a ring topology circulate election messages to choose a coordinator

References

- [1] Shi-Ding Lin, Qiao Lian, Ming Chen and Zheng Zhang, “A Practical Distributed Mutual Exclusion Protocol in Dynamic Peer-to-Peer Systems”, Lecture Notes in Computer Science, 2005, Volume 3279/2005, 11-21, DOI: 10.1007/978-3-540-30183-7_2
- [2] http://en.wikipedia.org/wiki/Causality

https://research.shanghai.nyu.edu/cn/centers-and-institutes/datascience/people/zheng-zhang

Election in Large Scale Networks

- Bully Algorithm and Ring Algorithm does scales poorly with the size of the network
  - Bully Algorithm needs $O(n^2)$ messages
  - Ring Algorithm requires maintaining a ring topology and requires 2n messages to elect a leader
- In large networks, these approaches do not scale
- We discuss a scalable election algorithm for large scale peer-to-peer networks

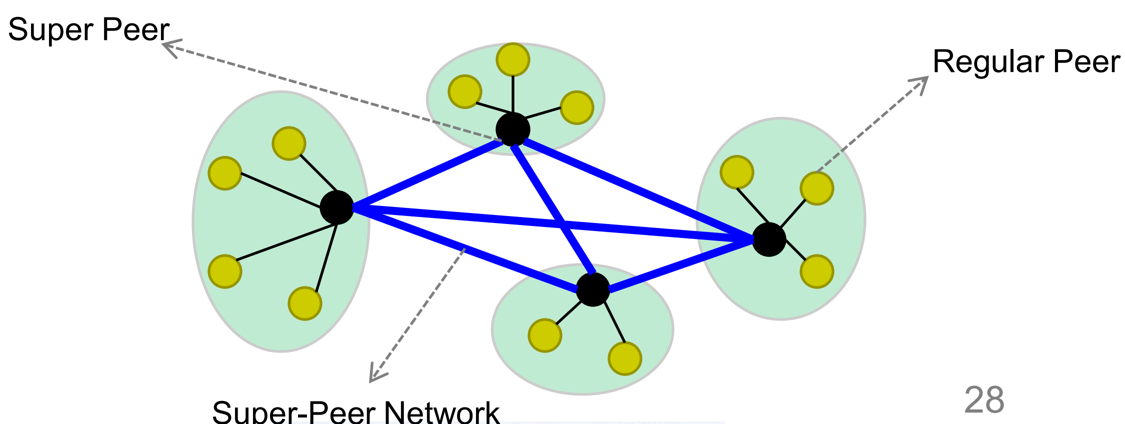
Election in Large Scale Peer-to-Peer Networks

- Many P2P networks have a hierarchical architecture for balancing the advantages between centralized and distributed networks

- Many peer-to-peer networks are neither completely unstructured nor completely centralized
  - Centralized networks are efficient and, they easily facilitate locate entities and data (e.g., name spaces)
  - Flat unstructured peer-to-peer networks are robust, autonomous and balances load between all peers

Super-peer

- In large unstructured Peer-to-Peer Networks, the network is organized into peers and *super-peers*
- Super-Peers are entities that not only participate as a peer, but also take on additional role of acting as a leader for a set of peers
  - Super-peer acts as a server for a set of client peers
  - All communication from and to a regular peer proceeds through a super-peer
- It is expected that super-peers are long-lived nodes with high-availability

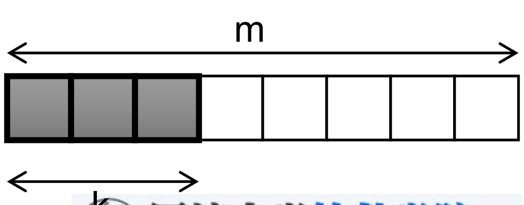


Super-Peer – Election Requirements

- In a hierarchical P2P network, several nodes have to be selected as super-peers
  - Traditionally, only one node was selected as a coordinator
- Requirements for a node being elected as super-peer
  - Super-peers should be evenly distributed across the overlay network
  - There should be a predefined proportion of super-peers relative to the number of regular peers
  - Each super-peer should not need to serve more than a fixed number of regular peers

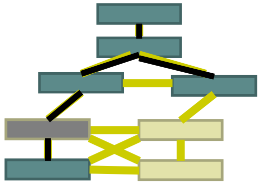
Election of Super-peers in a DHT-based system

- Recall: In a DHT-based system each node receives a random and uniformly assigned m-bit identifier
- We reserve first k-bits to identify super-peers
- For example, let m=8, and k=3
  - If p is the m-bit identifier of a peer, then its super-peer is the successor of (p AND 11100000)
- Proportion of super-peers
  - If we need N super-peers, then $k = \lceil \log_2(N) \rceil$ bits
  - In a network with M nodes, number of super-peers = $2^{k-m}M$

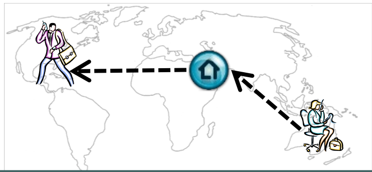


Election Algorithms

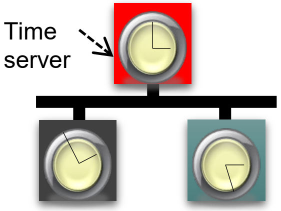
- Many distributed algorithms require one process to act as a coordinator
  - Typically, it does not matter which process is elected as the coordinator



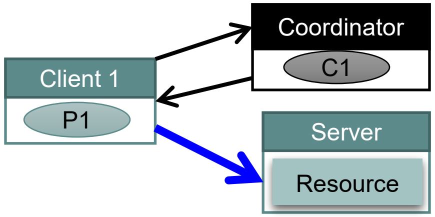
Root note selection in Multicasting



Home Node Selection in Naming



Berkeley Clock Synchronization Algorithm



A Centralized Mutual Exclusion Algorithm

Comparison of Mutual Exclusion Algorithms

Algorithm	Delay before a process can access the resource (in message times)	Number of messages required for a process to access and release the shared resource	Problems
Centralized			
Decentralized			
Token Ring			

- Assume that:

  n = Number of processes in the distributed system

  For the Decentralized algorithm:

  m = minimum number of coordinators who have to agree for a process to access a resource

  k = average number of requests made by the process to a coordinator to request for a vote

Comparison of Election Algorithms

Algorithm	Number of Messages for Electing a Coordinator	Problems
Bully Algorithm		
Ring Algorithm		

- Assume that:

  n = Number of processes in the distributed system