CISC 434: Synchronization - II

- Synchronizations – how processes in distributed systems synchronize
  - Clock Synchronization – no notion of globally shared clock
  - Logical Clocks – to reach agreement among the processes about correct ordering of events
  - Mutual Exclusion – a synchronization algorithm to limit concurrent access
  - Election Algorithms – selection of a coordinator process for synchronization
Mutual Exclusion

- To ensure a single activity on shared data structures at any point of time – a process enters a critical region to achieve mutual exclusion
  - A Centralized Algorithm
  - A Decentralized Algorithm
  - A Distributed Algorithm
  - A Token Ring Algorithm
Mutual Exclusion - A Centralized Algorithm

a) Process 1 asks the coordinator for permission to enter a critical region - permission is granted
b) Process 2 then asks permission to enter the same critical region - the coordinator does not reply
c) When process 1 exits the critical region, it tells the coordinator and the coordinator replies to 2
Mutual Exclusion - A Centralized Algorithm

- **Fairness** – since requests are granted in the order in which they are received, no process ever waits forever (starvation)
- **Simple to implement** – requires only three messages per use of resource (request, grant, release)
- **The coordinator is a single point of failure**
- **If processes block after making a request, they cannot distinguish between a dead coordinator and a denied permission**
- **For a large system, a coordinator may become a performance bottleneck**
Mutual Exclusion – A Decentralized Algorithm

- Each resource is assumed to be replicated n times, where each replica has its own coordinator.
- When a process wants to access the resource, it will simply need to get a majority vote \( m > n/2 \) coordinators.
- Unlike the centralized algorithm, the coordinator informs the requester even if the permission is not granted.
Mutual Exclusion – A Decentralized Algorithm

- Avoids single point of failure of the centralized algorithm
- The crashed coordinator
  - forgets any vote it gave before the crash
  - resets itself after the recovery
  - may grant permission to the same process again after its recovery
Mutual Exclusion - A Distributed Algorithm

- A process requests a shared resource by sending a message containing resource name, its process id, and current logical time to all other processes including itself.

- Three different cases on receiving the message:
  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 does not reply but queues the request.
  3. If the receiver wants to access the resource as well, it takes the timestamp of the incoming message with the one contained in the message that it has sent everyone - the lowest one wins.
Mutual Exclusion - A Distributed Algorithm

a) Two processes want to enter the same critical region at the same moment.
b) Process 0 has the lowest timestamp, so it wins.
c) When process 0 is done, it sends an OK also, so 2 can now enter the critical region.
Mutual Exclusion - A Distributed Algorithm

- The single point of failure is replaced by $n$ points of failure
  - If any process fails, it will fail to give permission, thus blocking all other processes to enter all critical regions
  - Probability of $n$ processes failing is $n$ times more than a single coordinator failing
  - The above problem can be avoided by enabling the receiver to respond to any request
Mutual Exclusion - A Distributed Algorithm

- All processes are involved in all decisions. If any process is not able to handle the load – bottleneck
- Requires multicast communication primitive or group memberships – works better for small groups that never change their group memberships
- Slower, more complicated, more expensive, and less robust that the original central one
Mutual Exclusion - A Token Ring Algorithm

- When the ring is initialized, process 0 is given a token
- The token circulates around the ring
- Getting a token means getting a permission to enter the critical region
- Ordering does not matter – a process knows who is next in line after itself

(a) An unordered group of processes on a network. (b) A logical ring constructed in software
Mutual Exclusion - A Token Ring Algorithm

- Only one token – only one process can get access at any time
- Token circulates in well-defined order – everybody gets the proper share
- Worst case – a process may have to wait for every other process to use the resource
- If the token is lost, detection and regeneration of the token is difficult
- If a process crashes, the token may be thrown over the head of the dead process
- It requires acknowledgment of the token receipt and maintenance of the current ring configuration
# A Comparison of Mutual Exclusion Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Messages per entry/exit (assume point to point communication channels)</th>
<th>Delay before entry (in message times) (assuming messages are passed sequentially)</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>3 (request, grant to enter, and release to exit)</td>
<td>2</td>
<td>Coordinator crash</td>
</tr>
<tr>
<td>Decentralized</td>
<td>3mk, k=1,2,… (m = no. of majority votes)</td>
<td>2m</td>
<td>Starvation, low efficiency</td>
</tr>
<tr>
<td>Distributed</td>
<td>2(n−1) (request: n-1; grant: n-1)</td>
<td>2(n−1)</td>
<td>Crash of any process</td>
</tr>
<tr>
<td>Token ring</td>
<td>1 to ∞ (1: every process wants to enter ∞: no process interested)</td>
<td>0 to n − 1 (0: token just arrived; n-1: token just departed)</td>
<td>Lost token, process crash</td>
</tr>
</tbody>
</table>
Election Algorithms

- Some general assumptions
  - All processes are exactly the same
  - Each process has a unique identifier
  - Every process knows the process numbers of every other process
- The Bully Algorithm
- A Ring Algorithm
  - Unlike most ring algorithms it does not send a token
Election Algorithms - The Bully Algorithm

1. \( P \) sends an \textit{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
Election Algorithm - The Bully Algorithm

a) Process 4 holds an election  
b) Process 5 and 6 respond, telling 4 to stop  
c) Now 5 and 6 each hold an election
d) Process 6 tells 5 to stop

e) Process 6 wins and tells everyone
Election Algorithm - A Ring Algorithm

- Processes are physically ordered – knows its neighbors
- Process IDs are appended in the message – if the message gets back to the initiating process, that process is considered as a coordinator
- If any successor is down, the sender skips until an active process is found
- Example – 2 and 5 together discovers simultaneously that 7 has died and starts sending election message independently
Summary

- **Clock Synchronization** – no notion of globally shared clock
- **Logical Clocks** – to reach agreement among the processes about correct ordering of events
- **Mutual Exclusion** – synchronization algorithms to limit concurrent access
  - Centralized
  - Decentralized
  - Distributed
  - Token Ring
- **Election Algorithms** – to agree on a coordinator
  - The Bully Algorithm
  - A Ring Algorithm