Ricart Agrawala Distributed Mutual Exclusion
Distributed Systems 2g1509

Ali Ghodsi  aligh@imit.kth.se
Distributed ME – Ricart/Agrawala [1/5]

**Idea:**

- To enter the critical section the requesting process has to get the approval of all processes (including itself).
- Processes only give the approval if they themselves are not in the critical section.
- If a process wants to enter the critical section itself and it has received a request from another process, it approves the process with the lowest process identifier.
A process $P_1$ receiving a request from a process $P_2$

- If $P_1$ does not want to enter the critical section and it is not in the critical section already, it sends back an accept message to $P_2$.
- If $P_1$ is already in the critical section, it buffers the incoming request in a FIFO queue.
- If $\text{IN\_CS}$ is not yet set, but $P_1$ wants to join, it does one of the following:
  - If $P_2$ has a lower process ID than $P_1$, it sends back an ACCEPT message to $P_2$.
  - If $P_1$ has a lower process ID than $P_2$, it queues $P_2$'s request in a local FIFO queue and enters the critical section itself.
Requesting node

- Sends a request message to all processes $p_i$ (including self)
- Waits for the receipt of an accept message from every process $p_i$
- Upon receipt of $n$ accepts, it enters the critical section
- After exiting the critical section it sends an ACCEPT message to the first process in its local FIFO queue
Clarification

- Why break ties by comparing process identifiers when a race condition occurs?
  - Any deterministic rule can be used to resolve ties, e.g. Using logical vector timestamps
  - If the rule is not deterministic, deadlocks might occur

If both P1 and P2 would accept the other process’ request, deadlock would occur!
Distributed ME – Ricart/Agrawala [5/5]

- **Advantages**
  - Completely distributed and decentralized

- **Disadvantages**
  - Every node is a single-point of failure
  - Costly and slow, $O(N)$ messages needed for each request