An Optimal Algorithm for Mutual Exclusion on Computer Networks

Ricart and Agrawala, 1981

Overview

- An algorithm for ensuring mutual exclusion over a network whose nodes communicate through message passing.
- Uses only 2(N-1) messages per "lock", where N is the number of nodes.

How it works

- A node attempting the lock sends out a REQUEST message to all nodes.
- Upon receipt of the REQUEST, each node either sends a REPLY message immediately if the requesting node has priority, otherwise it defers the REPLY until later.
- When requesting node receives a REPLY from all nodes, it enters its critical section.
- A sequence number is sent with each REQUEST. The number is always higher than the highest number received at that time.
- Priority is determined by a sequence number. If seq. numbers are the same, node number is used as a tie-breaker.

Variations

- Broadcast messages – if broadcasting is supported in the network, message traffic an be reduced to N messages, 1 broadcast REQUEST and (N-1) REPLIES.
- Implicit reply – send DEFERRED message and no REPLY message. If a certain amount of time passes on the requesting node without receiving a DEFERRED msg, a reply is assumed. Using this scheme, message traffic can vary between 1(N-1) and 3(N-1). With little contention, it's close to 1(N-1).
- If the above methods are combined, message count can be as low as 1.

More Variations

- Readers and Writers – Can be easily modified to solve readers and writers problem. Readers just REPLY immediately for all requests from other readers. Writers behave as usual.
- Establishing priority – Nodes can have different priorities. Low priority nodes use large sequence number increments, high priority nodes use small increments.

Insertion and removal of nodes

- Insertion of nodes
  - get unique id
  - get node list from other nodes
  - be placed on all other nodes' lists
  - get highest_sequence_number
- Removal of nodes
  - notify all other nodes of intention to leave.
  - can not request M.E.
  - immediately REPLY to all REQUESTs until ACK is received from all other nodes

Handling partial failure

- Node failures
  - if REPLY is not received before a timeout expires, an ARE_YOU_THERE messge is sent to each suspect node.
  - when ARE_YOU_THERE is received, we look at whether we have deferred a response to that node. If not, REPLY immediately. If so, return YES_I_AM_HERE
  - if timeout expires after an ARE_YOU_THERE message is sent, we notify all other nodes of the failure. Nodes waiting on a REPLY from the failed node will pretend it was received.