A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems

by M. Maekawa, ACM TOCS, 1985

Related Work

- Ricart/Agrawala
  - unanimous consensus
  - $2(N-1)$ messages
  - full connectivity, fault free network
- Thomas
  - voting scheme
  - majority of votes
  - $\frac{N}{2}$
- Gifford, Skeen, and Garcia-Molina/Barbara
  - weighted voting
  - further reduces complexity

Maekawa's Algorithm

- More powerful
- Distributed memory
- Assumes error free network
- Uses $c\sqrt{N}$ messages, $3 \leq c \leq 5$
- Combines
  - "voting" technique of Thomas
  - "deferral" of Ricart/Agrawala
  - additionally, uses "relinquishment" to avoid deadlocks

Main Idea

- **Arbitrator** node
- Partitions nodes in sets
- Sets **must** satisfy the following properties

(a) $\forall i, j \quad 1 \leq i, j \leq N, \quad S_i \cap S_j \neq \emptyset$

(b) $S_i, \quad 1 \leq i \leq N$, always contains i

(c) $\mid S_1 \mid = \mid S_2 \mid \cdots \mid S_N \mid = K$

(d) any $j, \quad 1 \leq j \leq N$ is contained in the $D \quad S_i's, 1 \leq i \leq N$

Comments about properties

(a) common node acts as arbitrator

(b) reduce messages - don't send to yourself

(c) and (d) guarantee distributed algorithm

(d) all nodes same responsibility

Aside: Ricart/Agrawala's algorithm satisfies properties for $K = N$ and $D = N$

Choice of $S_i^\prime s$

- Not unique
- Want to minimize the size of sets (i.e., K)
- Problem equivalent to finding a projection plane of N points
- Set of $S_i^\prime s$ is optimal if $(K-1)$ is a power of a prime
- Otherwise, got to relax conditions (c) and (d)
- Eventually we express $N = K(K-1)+1$
- Therefore, $K = \sqrt{N}$

Example...

What if $(K-1)$ is not a power of a prime?

- Two possibilities
  - use the previous algorithm, with some modifications
  - use a “redundant” algorithm with larger sets

First Algorithm (example...)

Suppose we need the sets for$N = 5$, use next larger $M = 7$ that satisfies the necessary condition...

$S_1 = \{1,2,3\}$
$S_2 = \{1,4,5\}$
$S_6 = \{1,6,7\}$
$S_2 = \{2,4,6\}$
$S_5 = \{2,5,7\}$
$S_7 = \{3,4,7\}$
$S_3 = \{3,5,6\}$

first $6 \to 4$ and $7 \to 5$,
then delete redundant

Problem here? loose symmetry...

Second Algorithm (example...)

Works if N is a square of an integer.

Suppose 9 nodes then:

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

and select:

$S_1 = \{1, 2, 3, 4, 7\}$ row 1, column 1

$S_2 = \{1, 2, 3, 5, 6\}$ row 1, column 2

$S_3 = \{1, 2, 3, 8, 9\}$ row 1, column 3

$S_4 = \{1, 4, 5, 6, 7\}$ row 2, column 1

etc ...

Advantage: Symmetry

Drawback: Size is $2\sqrt{N} - 1$

Messages

- **REQUEST**: original request sent to everybody else in your group:
  "I want your key"
- **LOCKED**: reply to request granting lock
  "Here is my key"
- **FAILED**: reply sent back to requester saying a process of higher priority already has lock
  "Sorry, my key has been given to a higher priority request"
- **INQUIRE**: message sent to node who has lock when a request of higher priority arrives (I have a new higher priority request)
  "Are you going to get all your keys? "
  "Give me back my key"

- **RELINQUISH**: message sent from receiver of INQUIRE to sender releasing lock:
  "Here is your key back since I'm not going to get all of mine"
  "But I'm still going to need it"
  Sent only if receiver has received a FAILED
- **RELEASE**: "also here's your key back, I'm finished"

The Algorithm

Send REQUEST to all set members

When a node gets REQUEST...

  If node not locked

    become locked and send LOCKED to requester

  If locked ...

    Put requester on queue

    If requesters priority $<$ one locked for

    OR one in the queue

      send FAIL to requester

    else

      send INQUIRE to locker

      (one with the key)

The Algorithm (cont.)

When a node receives an INQUIRE

If node received a FAILED

return a RELINQUISH

else

defer

When a node receives a RELINQUISH

Put this locker in queue

Send LOCKED to highest priority in queue

When get LOCKED from everyone in set,

do critical section

When done with critical section, send RELEASE

to all members of your set

Proofs (by contradiction)

- Mutual exclusion
  - property (a) helps...
- Deadlock-free
  - circular waiting is not possible
    due to sequence numbers
  - property (a) helps again...
- Starvation
  - OOOPS! Maekawa's proof has hole!
  - Hint: wrong assertion...

Message Traffic

- Light traffic
  - K-1 REQUESTS
  - K-1 LOCKS
  - K-1 RELEASES
- Heavy traffic
  - add K-1 FAILED
- Skewed traffic
  - add K-1 RELINQUISH

Recap...

- $c\sqrt{N}$ messages
- symmetric, fully parallel
- fully distributed
  (i.e., each node serves as an arbitrator for the same number of nodes)
- which algorithm to pick?
  depends on network topology and size, performance, reliability, extensibility...