Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Add links

Raymond's algorithm

From Wikipedia, the free encyclopedia
(Redirected from Raymond's Algorithm)

Raymond's Algorithm is a lock based algorithm for mutual exclusion on a distributed system. It imposes a logical structure (a K-ary tree) on distributed resources. As defined, each node has only a single parent, to which all requests to attain the token are made.

Algorithm [edit]

Nodal Properties [edit]

1. Each node has only one parent to whom received requests are forwarded
2. Each node maintains a FIFO queue of requests each time that it sees the token;
3. If any node is forwarding privilege to other node and has non-empty queue then it forwards a request message along

Algorithm [edit]

1. If a node *i* wishes to receive the token in order to enter into its critical section, it sends a request to its parent, node *j*.
   - If node *j* FIFO is empty, node *j* shifts *i* into the its FIFO queue; *j* then issues a request to its parent, *k*, that it desires the token
   - If node *j* FIFO queue is *not* empty, it simply shifts *i* into the queue
2. When node *j* receives the token from *k*, it forwards the token to *i* and *i* is removed from the queue of *j*
   - If the queue of *j* is not empty after forwarding the token to *i*, *j* must issue a request to *i* in order to get the token back

Note: If *j* wishes to request a token, and its queue is *not* empty, then it places itself into its own queue. Node *j* will utilize the token to enter into its critical section **if** it is at the head of the queue when the token is received.

Complexity [edit]

Raymond's algorithm is guaranteed to be *O(log n)* per critical section entry if the processors are organized into a *K-ary* tree. Additionally, each processor needs to store at most *O(log n)* bits because it must track *O(1)* neighbors.[1]

References [edit]

1. ^ R. Chow, T. Johnson; *Distributed Operating Systems & Algorithms; Addison-Wesley, 1997.*

See also [edit]

- Ricart-Agrawala algorithm
- Lamport's Bakery Algorithm
- Lamport's Distributed Mutual Exclusion Algorithm
- Maekawa's Algorithm
- Suzuki-Kasami's Algorithm
- Naimi-Trehel's Algorithm

Categories: Concurrency control algorithms