



WIKIPEDIA
The Free Encyclopedia

[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


[Deutsch](#)
[Français](#)

 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)



Maekawa's algorithm

From Wikipedia, the free encyclopedia
(Redirected from [Maekawa's Algorithm](#))



This article **does not cite any references or sources**. Please help [improve this article](#) by [adding citations to reliable sources](#). Unsourced material may be challenged and [removed](#). *(December 2009)*

Maekawa's algorithm is an algorithm for [mutual exclusion](#) on a [distributed system](#). The basis of this algorithm is a quorum like approach where any one site needs only to seek permissions from a subset of other sites.

Contents [\[hide\]](#)

- 1 Algorithm
 - 1.1 Terminology
 - 1.2 Algorithm
 - 1.3 Performance
- 2 See also
- 3 References

Algorithm [\[edit\]](#)

Terminology [\[edit\]](#)

- A *site* is any computing device which is running the Maekawa's Algorithm
- For any one request of the critical section:
 - The *requesting site* is the site which is requesting entry into the critical section.
 - The *receiving site* is every other site which is receiving the request from the requesting site.
- ts* refers to the local time stamp of the system according to its [logical clock](#).

Algorithm [\[edit\]](#)

Requesting site:

- A requesting site P_i sends a message `request(ts, i)` to all sites in its quorum set R_i .

Receiving site:

- Upon reception of a `request(ts, i)` message, the receiving site P_j will:
 - If site P_j does not have an outstanding **grant** message (that is, a **grant** message that has not been released), then site P_j sends a `grant(j)` message to site P_i .
 - If site P_j has an outstanding **grant** message with a process with higher priority than the request, then site P_j sends a `failed(j)` message to site P_i and site P_j queues the request from site P_i .
 - If site P_j has an outstanding **grant** message with a process with lower priority than the request, then site P_j sends an `inquire(j)` message to the process which has currently been granted access to the critical section by site P_j . (That is, the site with the outstanding **grant** message.)
- Upon reception of a `inquire(j)` message, the site P_k will:
 - Send a `yield(k)` message to site P_j if and only if site P_k has received a **failed** message from some other site or if P_k has sent a yield to some other site but have not received a new **grant**.
- Upon reception of a `yield(k)` message, site P_j will:
 - Send a `grant(j)` message to the request on the top of its own request queue. Note that the requests at the top are the highest priority.
 - Place P_k into its request queue.
- Upon reception of a `release(i)` message, site P_j will:
 - Delete P_i from its request queue.
 - Send a `grant(j)` message to the request on the top of its request queue.

Critical section:

- Site P_i enters the critical section on receiving a **grant** message from all sites in R_i .
- Upon exiting the critical section, P_i sends a **release**(i) message to all sites in R_i .

Quorum set (R_x):

A quorum set must abide by the following properties:

1. $\forall i \forall j [R_i \cap R_j \neq \emptyset]$
2. $\forall i [P_i \in R_i]$
3. $\forall i [|R_i| = K]$
4. Site P_i is contained in exactly K request sets

Therefore:

$$\bullet |R_i| \geq \sqrt{N-1}$$

Performance [\[edit\]](#)

- Number of network messages: $3\sqrt{N}$ to $6\sqrt{N}$
- Synchronization delay: 2 message propagation delays

See also [\[edit\]](#)

- [Lamport's bakery algorithm](#)
- [Lamport's Distributed Mutual Exclusion Algorithm](#)
- [Ricart-Agrawala algorithm](#)
- [Raymond's algorithm](#)

References [\[edit\]](#)

- Mamoru Maekawa, Arthur E. Oldehoeft, Rodney R. Oldehoeft (1987). Operating Systems: Advanced Concept. Benjamin/Cummings Publishing Company, Inc.
- B. Sanders (1987). The Information Structure of Distributed Mutual Exclusion Algorithms. ACM Transactions on Computer Systems, Vol. 3, No. 2, pp. 145–59.

Categories: [Concurrency control algorithms](#)

This page was last modified on 12 February 2014, at 18:24.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

