Bully algorithm

From Wikipedia, the free encyclopedia

This article has multiple issues. Please help **improve it** or discuss these issues on the **talk page**. [hide]

- This article **may be confusing or unclear to readers**. In particular, is it unclear what is non-trivial about this algorithm. *(January 2015)*
- This article **may be confusing or unclear to readers**. In particular, is the communication model missing, is it all-to-all communication?. *(January 2015)*
- This article or section **appears to contradict itself about the assumption of being synchronous (having a rounds for all processes) and using timeouts.** *(January 2015)*
- This article **may be confusing or unclear to readers**. In particular, is the section about "Election Type" a mix between an algorithm description and a comparison with another algorithm (which one even?). *(January 2015)*

The **bully algorithm** is a programming mechanism that applies a hierarchy to nodes on a system, making a process coordinator or slave. This is used as a method in distributed computing for dynamically electing a coordinator by process ID number. The process with the highest process ID number is selected as the coordinator.

Contents [hide]

Assumptions [edit]

As this algorithm is part from a system model that tries to make a fail-free system (like the solution shown in Lamport paper), we need some assumptions for the model.

- The system is synchronous and uses timeout for identifying process failure. (so you can have Delta and Cmax in order to calculate timeout as opposed to asynchronous systems where you can't calculate a timeout and then you can't distinguish between an omission fail on a process or a delay)
- Allows processes to crash during execution of algorithm. (To=2*Delta+Cmax; so timer knows when omission fails happens)
- Message delivery between processes should be reliable.(Coordinator dilemma,¿is it trustworthy; or suplantation,inyection,replication,DoS may happen?)
- Prior information about other process id's must be known. (This works as Leslie Lamport solution for Byzantine dilemma, where coordinator needs a key and id for each process and where processors hierarchy stipulates nodes as Generals, Commanders and Lieutenants but without a key and with only coordinator and slaves)

Notice that this algorithm can be applied over distributed or centralized systems , because processes can be located on one machine or over severals as you can make multicast calls or system calls or both if your system is hybrid (for example a multithread server working with several clients)

Component calls [edit]

These are the Bully-algorithm components:

- Election Message: Sent to announce faster election

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
Español
فارسی
Italiano
Українська
Edit links

- Answer Message: Respond to the election message
- Coordinator message: Sent to announce the identity of the elected process

Compared with Ring election algorithm:

- Assumes that system is synchronous
- Uses timeout to detect process failure/crash
- Each processor knows which processor has the higher identifier number and communicates with that[1]

Bully algorithm structure [edit]

When a process P determines that the current coordinator is down because of message timeouts or failure of the coordinator to initiate a handshake, it performs the following sequence of actions:

1. P broadcasts an election message (inquiry) to all other processes with higher process IDs, expecting an "I am alive" response from them if they are alive.
2. If P hears from no process with a higher process ID than it, it wins the election and broadcasts victory.
3. If P hears from a process with a higher ID, P waits a certain amount of time for any process with a higher ID to broadcast itself as the leader. If it does not receive this message in time, it re-broadcasts the election message.
4. If P gets an election message (inquiry) from another process with a lower ID it sends an "I am alive" message back and starts new elections.

Note that if P receives a victory message from a process with a lower ID number, it immediately initiates a new election. This is how the algorithm gets its name - a process with a higher ID number will bully a lower ID process out of the coordinator position as soon as it comes online.

See also [edit]

- Distributed Computing#Coordinator election
- Chang and Roberts algorithm

References [edit]

1. ^ Jean Dollimore, Tim Kindberg, George F. Coulouris, "Distributed systems : concepts and design (Third Edition)," in *Distributed systems : concepts and design (Third Edition)*. Addison-Wesley, 2003.

- Witchel, Emmett (2005). "Distributed Coordination". Retrieved May 4, 2005.
- Hector Garcia-Molina, Elections in a Distributed Computing System, IEEE Transactions on Computers, Vol. C-31, No. 1, January (1982) 48-59
- L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem" ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982.

Categories: Distributed algorithms