Distributed System with Bully Algorithm

Created

August 16, 2019

Author

Yehia Ahmed ElKhawanky

Contents

Contents 2
1 Introduction 4
2 How to use the application 4
3 Used IPC technique 4
4 Process flow chart 4
5 Some enhancements can be added 5

Revision History

Version	Date	Name	Description
1.0	16/8/2019	Yehia Khawanky	Initial Document
1.1	17/8/2019	Yehia Khawanky	Adding all documentation parts

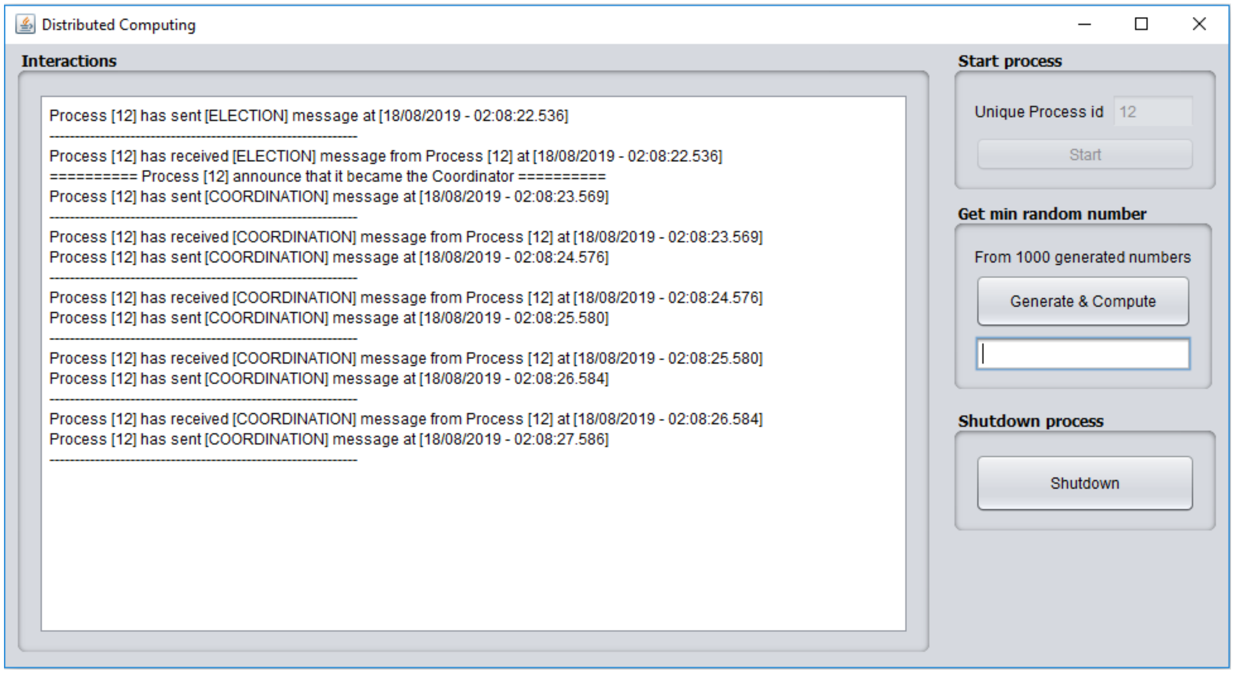
1 Introduction

This is the documentation of the Bully Election Algorithm, with a distributed computing random number generator example.

2 How to use the application

2.1 *Initially please run the executable jar file (just double click)*

1. The below panel opens.
2. Enter a unique integer process Id.
3. Press “start”, you can see any process interaction on the Interactions Area.
4. To run another process, re-run the same jar.
5. The coordinator has the “Generate & Compute” button enabled.
6. If you click “Generate & Compute”, the minimum random number generator process starts and uses all other process to compute as a distributed system.
7. The result will appear in the text field.
8. If no other process is up, it prints that there is no running process are up to compute.
9. To kill any process, you can close the window, or you can click “shutdown” button so the window still there, so you see the logs.
10. Now you can run and play.



```
Process [12] has sent [ELECTION] message at [18/08/2019 - 02:08:22.536]
------------------------------------------------------------
Process [12] has received [ELECTION] message from Process [12] at [18/08/2019 - 02:08:22.536]
========== Process [12] announce that it became the Coordinator ==========
Process [12] has sent [COORDINATION] message at [18/08/2019 - 02:08:23.569]
------------------------------------------------------------
Process [12] has received [COORDINATION] message from Process [12] at [18/08/2019 - 02:08:23.569]
Process [12] has sent [COORDINATION] message at [18/08/2019 - 02:08:24.576]
------------------------------------------------------------
Process [12] has received [COORDINATION] message from Process [12] at [18/08/2019 - 02:08:24.576]
Process [12] has sent [COORDINATION] message at [18/08/2019 - 02:08:25.580]
------------------------------------------------------------
Process [12] has received [COORDINATION] message from Process [12] at [18/08/2019 - 02:08:25.580]
Process [12] has sent [COORDINATION] message at [18/08/2019 - 02:08:26.584]
------------------------------------------------------------
Process [12] has received [COORDINATION] message from Process [12] at [18/08/2019 - 02:08:26.584]
Process [12] has sent [COORDINATION] message at [18/08/2019 - 02:08:27.586]
------------------------------------------------------------
```

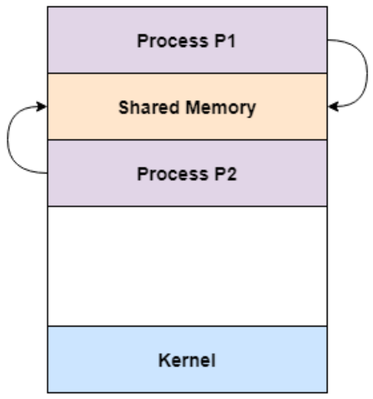
3 Used IPC technique

In this application we used **Share Memory** technique to communicate between different processes.

Election Process Shared Memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other, during the Election process.

Computing Shared Memory is another memory which is done so that the processes can communicate with each other, during the Computing process.

The following diagram can illustrate the shared memory model of process communication:



Shared Memory Model

I chose this technique for the following reason:

- Easy to be implemented while can fit to implement all the requirements.
- Good fit if the processes run in the same machine (not over the network), and this the suggested design.
- Communication between process is fast.
- Platform independent (can work in Windows, Linux, Mac).
- Does not involve any system calls or protocol-induced overheads.

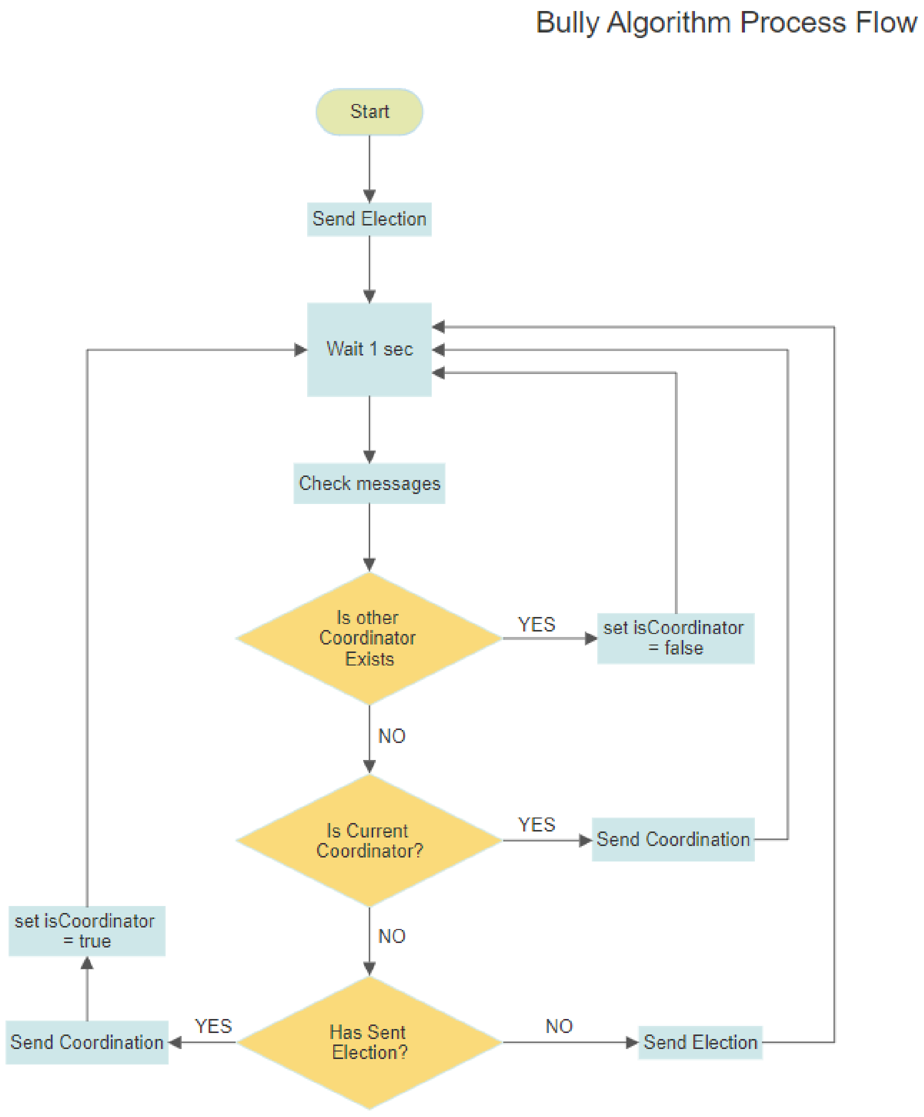
Disadvantage of this technique:

- We should take care of synchronization between different processes in accessing the shared memory.

4 Process flow chart

4.1 *Bully Algorithm flow chart*

Note: this chart demonstrates the Bully Algorithm flow and does not include the random number generator part.



4.2 *Random numbers distributed computing flow chart*

- N/A

5 Some enhancements can be added

5.1 Some missing special scenarios or enhancements

- In the distributed computing may be some special scenarios not covered and tested well, for example when the coordinator or a running process went down while computing the results.
- Another example to let the distributed task generic, not only the random number one.
- And may be give the option to use other IPC techniques to cover if the processes run on completed remote different servers, also, DB logging can be added, validations, exception handling, and etc...
- Main enhancements can be done, like reformatting the code to be clearer and adding some design pattern to make the system more scalable and easier to maintain
- Make some of the fixed values configurable, or collect it from user, for example the count of random numbers to generate and the number range.
- Also, as a ***very important enhancement***, instead of sending and receiving messages as steam of String, it should be enhanced to use a synchronized and serialized Queue that contains all ongoing messages.