Distributed Mutual Exclusion - A golang implementation

Lauritz Andersen **lana@itu.dk**
Johan Sandager Bennedsen **jsbe@itu.dk**
Jonas Kramer **kram@itu.dk**

February 21, 2024

1 System requirements

1.1 R1: Implement a system with a set of peer nodes, and a Critical Section, that represents a sensitive system operation. Any node can at any time decide it wants access to the Critical Section. Critical section in this exercise is emulated, for example by a print statement, or writing to a shared file on the network.

The system has been implemented with a set of peer nodes, which we refer to as "clients" in our code. We do not have a concrete, existing Critical Section as such, as for example a shared file that can be written to. We have instead emulated a Critical Section with a print statement. When a user provides some input, it signifies its decision to access the Critical Section. The user can continuously provide input, thereby ensuring that the requirement "Any node can at any time decide it wants access to the Critical Section" is met. The elected coordinator can grant access to the user, and the input provided by the user will be printed to the terminal. This action symbolises the coordinator granting access to the Critical Section. How the coordinator and elections are implemented can be read in section 2.

1.2 R2: Safety: Only one node at the same time is allowed to enter the Critical Section

Access to the critical section is granted by the coordinator (Bully algorithm). The way we have implemented it, although inefficient is as follows:
A client struct will have the boolean IsAvailable, by defualt set to false. The altercation of this, only happens when the client is also the coordinator. When the coordinator receives a request for resource access, it first checks if it is indeed the coordinator. If so, it will then check if the resource is available. If so, it will grant access for a set amount of time, by setting IsAvailable to false. After this time, it will be true again, and another client can have access. The trick here is that both the requesting client and the coordinator have to agree on this pause, so a client does not keep using the resource after the time has run out.
This, we have not implemented gracefully, but rather hardcoded. An alternative would be to set the time in the ResrouceRequestResponseMessage.

1.3 R3: Liveliness: Every node that requests access to the Critical Section, will get access to the Critical Section (at some point in time)

The way our implementation handles access to the critical section is by appointing a coordinator using voting (Bully Algorithm), and then making a access request to said coordinator. The implementation as it is, can either decide to grant that access, or not. If it is denied (it is in use), a client can make another request. The access model here is "Først til mølle" (first come, first serve), which of course weakens the sense of fairness, but a client is always allowed to make another request, therefore allowing it to eventually enter the cirtical section.

2 Discussion of Algorithm

We have chosen to implement the Bully algorithm. Our implementation goes as follows:
When a new client joins, it will make a RPC on itself, calling an election. This election will either verify the current coordinator, or make a new one coordinator. This is done by first checking whether the client itself holds the largest port, if so it will AssertCoordinator as itself. If not the case it will be invoking the CallElection call on all clients larger the current one, upon receiving an answer it will rest, since it know some client is larger than itself. If no answer is received, it will assume all above dead, and declare itself the coordinator.
When a client wishes to access the critical section, it will call RequestResource on the current coordinator. The current coordinator will then check if the resource is available (a boolean stored internally, not the smartest solution we know) and if so grant access and reserve the resource for some set time period. This time period is hardcoded, but could alternatively be sent with the ResourceRequestResponseMessage.
We are aware that the implementation has some sub-optimal design, especially in relation to the access to the critical section, but it works nonetheless.

2.1 Appendix

2.1.1 Log

Client 1

```
2023/11/13 20:56:51 Hey I'm at port: 5001
2023/11/13 20:56:51 Listening at: 192.***.**.**:5001
2023/11/13 20:56:51 Calling election, coordinator before election:
2023/11/13 20:56:51 Election called!
2023/11/13 20:56:51 No response means I'm the leader muhahaha
2023/11/13 20:56:51 Telling my subjects I'm the boss around here, subject:
5001
2023/11/13 20:56:51 Someone thinks they are coordinator, this guy eh: 5001
2023/11/13 20:56:51 Coordinator after election: 5001
2023/11/13 20:56:53 Someone thinks they are coordinator, this guy eh: 5002
2023/11/13 20:56:56 Someone thinks they are coordinator, this guy eh: 5003
I want access
2023/11/13 20:57:00 Awww man, aint no access to resource for me
2023/11/13 20:57:20 Someone thinks they are coordinator, this guy eh: 5002
```

Client 2

```
2023/11/13 20:56:53 Hey I'm at port: 5002
2023/11/13 20:56:53 Listening at: 192.***.**.**:5002
2023/11/13 20:56:53 Calling election, coordinator before election:
2023/11/13 20:56:53 Election called!
2023/11/13 20:56:53 No response means I'm the leader muhahaha
2023/11/13 20:56:53 Telling my subjects I'm the boss around here, subject:
5002
2023/11/13 20:56:53 Someone thinks they are coordinator, this guy eh: 5002
2023/11/13 20:56:53 Telling my subjects I'm the boss around here, subject:
```

```
5001
2023/11/13 20:56:53 Coordinator after election: 5002
2023/11/13 20:56:56 Someone thinks they are coordinator, this guy eh: 5003
I want access
2023/11/13 20:57:06 Access to print input granted. Input: I want access
I'm still here
2023/11/13 20:57:20 Election called!
2023/11/13 20:57:20 No response means I'm the leader muhahaha
2023/11/13 20:57:20 Telling my subjects I'm the boss around here, subject: 5002
2023/11/13 20:57:20 Someone thinks they are coordinator, this guy eh: 5002
2023/11/13 20:57:20 Telling my subjects I'm the boss around here, subject: 5001
```

Client 3

```
2023/11/13 20:56:56 Hey I'm at port: 5003
2023/11/13 20:56:56 Listening at: 192.***.**.**:5003
2023/11/13 20:56:56 Calling election, coordinator before election:
2023/11/13 20:56:56 Election called!
2023/11/13 20:56:56 No response means I'm the leader muhahaha
2023/11/13 20:56:56 Telling my subjects I'm the boss around here, subject: 5003
2023/11/13 20:56:56 Someone thinks they are coordinator, this guy eh: 5003
2023/11/13 20:56:56 Telling my subjects I'm the boss around here, subject: 5002
2023/11/13 20:56:56 Telling my subjects I'm the boss around here, subject: 5001
2023/11/13 20:56:56 Coordinator after election: 5003
I want access
2023/11/13 20:57:10 Awww man, aint no access to resource for me
```