Mandatory Handin 4 - Distributed Mutual Exclusion

System requirements

R1

The system fulfills the system requirements of using only peer nodes. This is achieved by using one type of node that has both the functionality of a server, it gives peers the ability to connect to itself, and the functionality of a client, it connects to others. The peer structure in our system has a name (for user readability), an address (for connecting to the machine), a port (for listening to other peers), the gRPC mutual exclusion service for responding to requests (see Appendix 1.1), and a function for sending mutual exclusion requests via CLI.

The critical section in our program is a function that changes the peers state from “wanted” to “held” which follows the procedure of the algorithm; “Ricart & Agrawala”. For better user experience print statements are made within the function and sleep time is used for making it possible to queue other peers who ask to enter the critical section (see appendix 1.2).

R2

Only one peer can enter the critical section at a time. This is achieved by using “Ricart & Agrawala algorithm”. To enter the critical section the user must write “mutual” in the peer terminal to start the algorithm. When mutual has been written in the terminal, all other connected peers (which connections are saved in a map <”ip:port”,connection>) will receive a message asking for the authorization to enter the critical section. The peers receiving the question will authorize the asking peer to enter critical section if the peer itself is not in the section. If the asked peer also wants to access the critical section, it will check whether himself or the asker has the lowest Lamport time. If the asked peer has the lowest Lamport time he will put the asking peer in a queue and it will only respond to him once he has performed the critical section. The reason for only one is able to enter the critical section is caused by the peers not answering the other peers with higher Lamport time. This will guarantee that only the one with the lowest Lamport time will enter. When he then exits, he will answer the other peers with higher Lamport time, hence moving them up the queue (see appendix 1.3).

R3

The fact that every peer that wants to enter the critical section will enter it at some point is guaranteed not only by the algorithm itself but also by the fact that in case of waiting for a message response from a peer that is ended causes the wait to be deleted and the peer is no

longer considered among those connected. In this way each thread will always respond sooner or later depending on its development (see Appendix 1.3) and in case of a failed connection the waiting thread will advance thus guaranteeing liveness.

Algorithm discussion

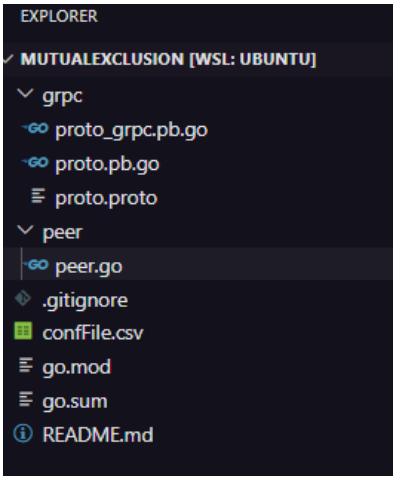
The Ricart & Agrawala algorithm is a distributed mutual exclusion algorithm used in distributed computing systems. Its purpose is to ensure that only one process at a time can access a critical section (shared resource) to prevent conflicts and maintain consistency in the distributed environment. The algorithm relies on a request-reply mechanism between processes, ensuring mutual exclusion while allowing coordinated access to shared resources. Peers can detect network failures, and the system is considered scalable since new peers (or old ones after reboot) can also be added to the system. As you can see from the log's screens, when two threads request to execute a mutual execution at the same time, only one executes it immediately, while the other one waits for a response from the one that is executing it. We also note how then, if one of the peers terminates, the others continue to execute correctly, discarding the missed response from the terminated peer.

Github Repository

https://github.com/chrolle69/MutualExclusion

Appendix

1.1




```go
type Peer struct {
    proto.UnimplementedMutualExlusionServiceServer
    name    string
    address string
    port    int
}
```

1.2

```go
func criticalSection() {
    state = Held
    log.Println("Starting critical section")
    time.Sleep(time.Duration(rand.Intn(4)+10) * time.Second)
    log.Println("Ending critical section")
    state = Released
}
```

1.3

```go
    // Ricart-Agrawala Algorithm
    if (state == Held) || (state == Wanted && (in.Time > int32(lamport_time))) {
        // queue the reply (just wait unline i'm done)
        for state == Held || state == Wanted {
            time.Sleep(500 * time.Millisecond)
        }
    }
    log.Printf("Peer [%s] authorized to do mutual exection", peerRef)
    // update lamport time
    setTime(int(in.Time))
    return &proto.Answer{
        Reply: true,
    }, nil
```

Logs:

Peer 1:

```
PS C:\Users\matte\Documents\GitHub\MutualExclusion> go run ./peer/peer.go -row 0
Your settings are : 127.0.0.1 address, 50051 port
2023/11/13 22:24:03 Lamport 1: Started peer receiving at address: 127.0.0.1 and at port: 50051
2023/11/13 22:24:03 Lamport 3: Created TCP connection to the 127.0.0.1 address at port 50052
2023/11/13 22:24:03 Lamport 4: Created TCP connection to the 127.0.0.1 address at port 50053
2023/11/13 22:24:03 Insert 'mutual' to do mutual execution or 'exit' to quit or anything else to increment time [Actual Lamport Time: 4]
2023/11/13 22:24:06 Lamport 6: Peer [127.0.0.1:50053] asked for a mutual exection
2023/11/13 22:24:06 Lamport 6: Peer [127.0.0.1:50053] authorized to do mutual exection
2023/11/13 22:24:10 Lamport 9: Peer [127.0.0.1:50052] asked for a mutual exection
2023/11/13 22:24:10 Lamport 9: Peer [127.0.0.1:50052] authorized to do mutual exection
exit
PS C:\Users\matte\Documents\GitHub\MutualExclusion>
```

Peer 2:

```
PS C:\Users\matte\Documents\GitHub\MutualExclusion> go run ./peer/peer.go -row 1
Your settings are : 127.0.0.1 address, 50052 port
2023/11/13 22:23:49 Lamport 1: Started peer receiving at address: 127.0.0.1 and at port: 50052
2023/11/13 22:23:49 Lamport 3: Created TCP connection to the 127.0.0.1 address at port 50051
2023/11/13 22:23:49 Lamport 4: Created TCP connection to the 127.0.0.1 address at port 50053
2023/11/13 22:23:49 Insert 'mutual' to do mutual execution or 'exit' to quit or anything else to increment time [Actual Lamport Time: 4]
2023/11/13 22:24:06 Lamport 6: Peer [127.0.0.1:50053] asked for a mutual exection
2023/11/13 22:24:06 Lamport 6: Peer [127.0.0.1:50053] authorized to do mutual exection
mutual
2023/11/13 22:24:10 Lamport 9: Asked Peer [127.0.0.1:50051] for permission
2023/11/13 22:24:10 Lamport 11: Got permission from peer [127.0.0.1:50051]
2023/11/13 22:24:10 Lamport 12: Asked Peer [127.0.0.1:50053] for permission
2023/11/13 22:24:18 Lamport 15: Got permission from peer [127.0.0.1:50053]
2023/11/13 22:24:18 Lamport 16: Starting critical section
2023/11/13 22:24:30 Lamport 17: Ending critical section
2023/11/13 22:24:30 Insert 'mutual' to do mutual execution or 'exit' to quit or anything else to increment time [Actual Lamport Time: 17]
mutual
2023/11/13 22:24:33 Lamport 19: Asked Peer [127.0.0.1:50053] for permission
2023/11/13 22:24:33 Lamport 21: Got permission from peer [127.0.0.1:50053]
2023/11/13 22:24:33 Lamport 22: Asked Peer [127.0.0.1:50051] for permission
2023/11/13 22:24:33 Lamport 22: Peer [127.0.0.1:50051] no more available, removed from connected peers
2023/11/13 22:24:33 Lamport 23: Starting critical section
2023/11/13 22:24:43 Lamport 24: Ending critical section
2023/11/13 22:24:43 Insert 'mutual' to do mutual execution or 'exit' to quit or anything else to increment time [Actual Lamport Time: 24]
exit
PS C:\Users\matte\Documents\GitHub\MutualExclusion>
```

Peer 3:

```
PS C:\Users\matte\Documents\GitHub\MutualExclusion> go run ./peer/peer.go -row 2
Your settings are : 127.0.0.1 address, 50053 port
2023/11/13 22:23:55 Lamport 1: Started peer receiving at address: 127.0.0.1 and at port: 50053
2023/11/13 22:23:55 Lamport 3: Created TCP connection to the 127.0.0.1 address at port 50051
2023/11/13 22:23:55 Lamport 4: Created TCP connection to the 127.0.0.1 address at port 50052
2023/11/13 22:23:55 Insert 'mutual' to do mutual execution or 'exit' to quit or anything else to increment time [Actual Lamport Time: 4]
mutual
2023/11/13 22:24:06 Lamport 6: Asked Peer [127.0.0.1:50051] for permission
2023/11/13 22:24:06 Lamport 8: Got permission from peer [127.0.0.1:50051]
2023/11/13 22:24:06 Lamport 9: Asked Peer [127.0.0.1:50052] for permission
2023/11/13 22:24:06 Lamport 10: Got permission from peer [127.0.0.1:50052]
2023/11/13 22:24:06 Lamport 11: Starting critical section
2023/11/13 22:24:10 Lamport 12: Peer [127.0.0.1:50052] asked for a mutual exection
2023/11/13 22:24:18 Lamport 13: Ending critical section
2023/11/13 22:24:18 Insert 'mutual' to do mutual execution or 'exit' to quit or anything else to increment time [Actual Lamport Time: 13]
2023/11/13 22:24:18 Lamport 13: Peer [127.0.0.1:50052] authorized to do mutual exection
2023/11/13 22:24:33 Lamport 19: Peer [127.0.0.1:50052] asked for a mutual exection
2023/11/13 22:24:33 Lamport 19: Peer [127.0.0.1:50052] authorized to do mutual exection
exit
PS C:\Users\matte\Documents\GitHub\MutualExclusion>
```