Mutua esclusione distribuita

Progetto per il corso di Sistemi Distribuiti e Cloud Computing

Sara Da Canal 0316044

Anno 2021/2022

Obiettivo

Lo scopo di questo progetto è implementare un sistema che simuli l'esecuzione di due algoritmi di mutua esclusione distribuita e un algoritmo di mutua esclusione basato su coordinatore.

Algoritmi scelti

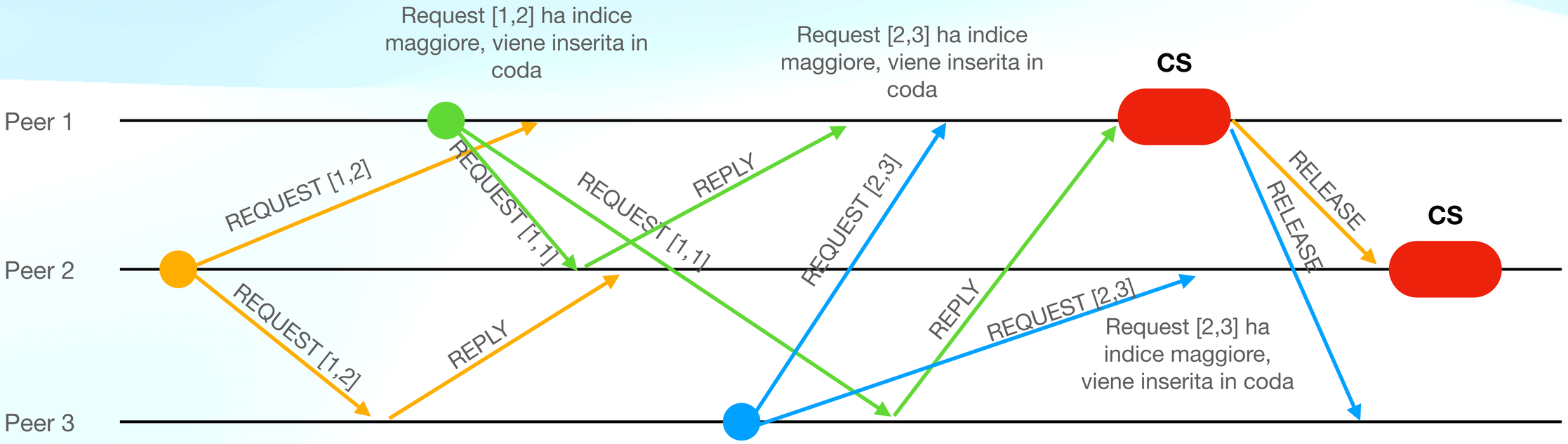
- Ricart-Agrawala
- Maekawa
- Token centralizzato

Tecnologie adottate

- Golang
- Docker
- Docker-compose
- EC2

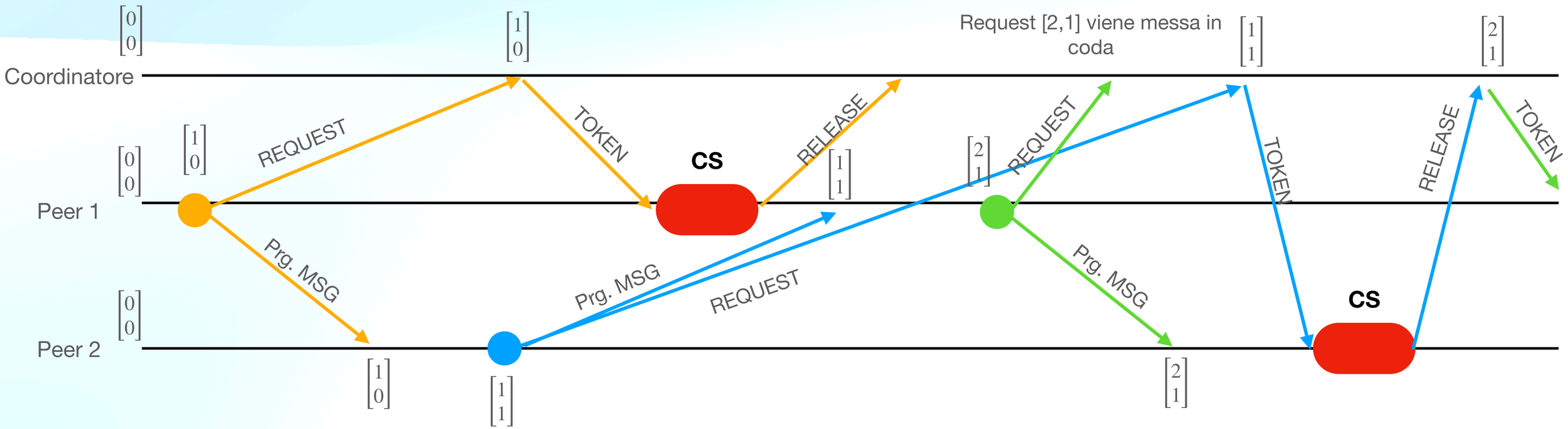
Algoritmo di Ricart-Agrawala

Algoritmo basato su autorizzazione: ogni peer deve ottiene il permesso da tutti gli altri per entrare in sezione critica. Usa clock logico scalare per l'ordinamento delle richieste.



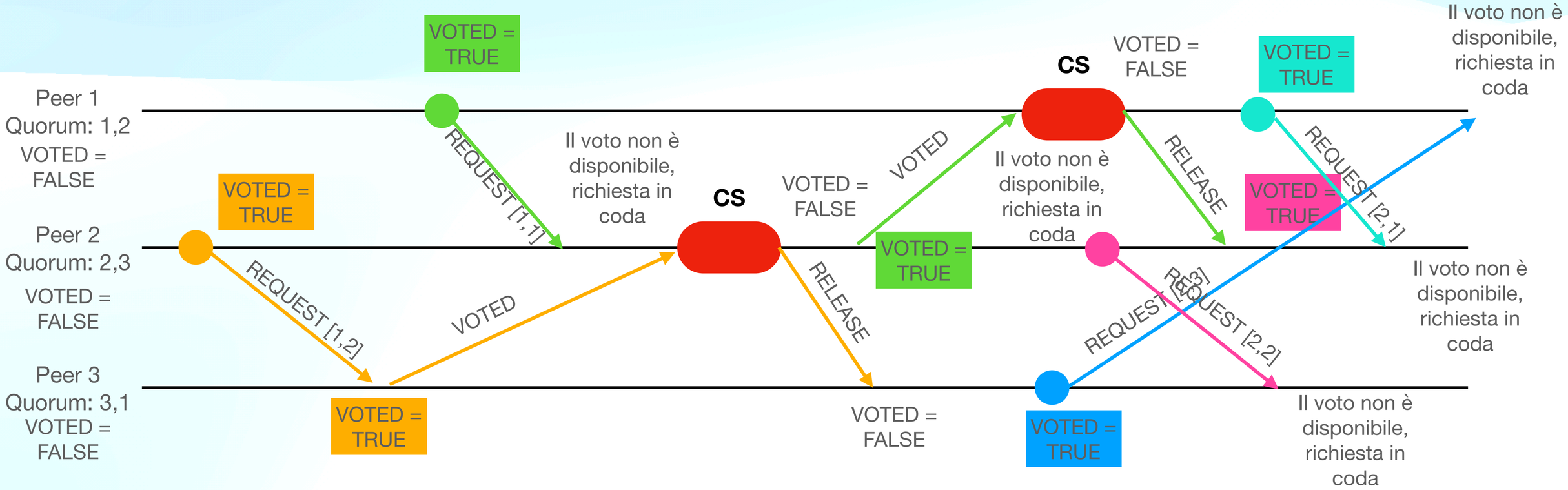
Algoritmo token centralizzato

Algoritmo basato su token: un peer deve possedere il token per entrare in sezione critica. Il token è gestito dal coordinatore, che lo manda a chi lo richiede. Usa clock logico vettoriale per l'ordinamento delle richieste.



Algoritmo di Maekawa

Algoritmo basato su quorum: ogni peer deve ottenere i voti di tutti i componenti del suo quorum. Ogni peer può dare un solo voto per volta.



RPC

Server

```
rpc.RegisterName("API", new(Api))
rpc.HandleHTTP()
lis, e := net.Listen("tcp",
“:<PORT>")
http.Serve(lis, nil)
```

Client

```
client, err :=
rpc.DialHTTP(“tcp”,”<IP>:<PORT>”)
err = client.Call("API.<name>",
&args, &reply)
```

API template:

Func (api *Api) ApiName (args *struct, reply *struct) error {}

Sistema di registrazione

Il sistema di registrazione prende in input il tipo di algoritmo e il numero di peer. Aspetta di aver raggiunto il numero di peer richiesto e invia ai peer che si sono registrati le seguenti informazioni:

```
type Registration_reply struct
{
    Peer  []Peer
    Alg   Algorithm
    Index int
    Mask  []int
}
```

```
type Peer struct {
    IP   string
    Port int
}
```

- Lista dei peer (ip e porta)
- Algoritmo da usare
- Indice del peer
- Maschera di bit per il quorum

Se l'algoritmo selezionato è centralizzato, il sistema di registrazione avvia anche il coordinatore

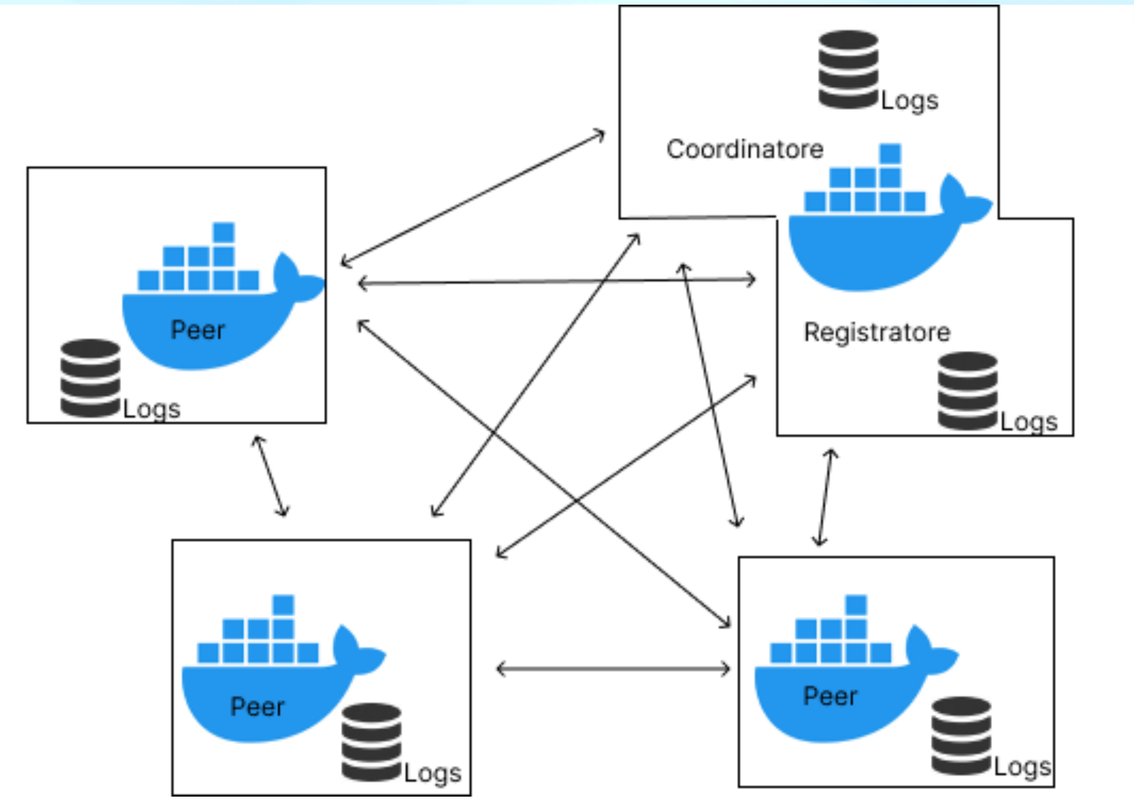
Architettura

I componenti del sistema distribuito vengono eseguiti all'interno di container Docker.
Ci sono due tipi di container:

- Peer
- Registerer

Il coordinatore per l'algoritmo basato su token centralizzato si trova nello stesso container del sistema di registrazione.

I diversi container sono collegati tra loro usando Docker-compose.



Dockerfiles

```
FROM golang:1.17-alpine
ADD . /reg
WORKDIR /reg
COPY . .
RUN go mod download
RUN go build -o ./main ./Registerer
ENTRYPOINT ["./main"]
```

```
FROM golang:1.17-alpine
ADD . /peer
WORKDIR /peer
COPY . .
RUN go mod download
RUN go build -o ./main ./Peer
ENTRYPOINT ["./main"]
```

Docker-compose definisce due tipi di servizio, uno per ogni dockerfile, e una rete.
Ad ogni servizio viene associato un volume di log
Viene usato il parametro —scale per far partire diverse istanze del container contenente il peer

Parametri e avvio

- Avvio: sh launch.sh -n num -a alg -v -d delay
- Spegnimento: sh down.sh

```
Options:

 FLAG   VALUES                        DESCRIPTION
 -n     <number-of-peers>             Number of peers to spawn
 -a     [auth | token | quorum]       Algorithm to use
 -v                                   Verbose modality
 -d     [slow | medium | fast]        Level of net congestion
```

Test

I test sono basati sulla lettura e controllo dei file di log. Vengono fatti tre tipi di controllo:

- Mutua esclusione
- Fairness
- Assenza di deadlock

- Avvio: sh test/testing.sh -n num -a alg -d delay

File di log

```
2022/10/20 09:51:22 Peer client starting in debug mode
2022/10/20 09:51:22 Configuration file successfully opened
2022/10/20 09:51:22 Configuration successfully loaded
2022/10/20 09:51:22 I'll trying to access shared resources
2022/10/20 09:51:22 Registered!
2022/10/20 09:51:22 Ricart Agrawala algorithm client  4  started
2022/10/20 09:51:22 Process listening on ip 172.22.0.7 and port
14541
2022/10/20 09:51:22 Trying to enter critic section
2022/10/20 09:51:22 Request sent to process  0
2022/10/20 09:51:22 Request sent to process  1
2022/10/20 09:51:22 Request sent to process  2
2022/10/20 09:51:22 Request sent to process  3
2022/10/20 09:51:23 Reply arrived from process  0
2022/10/20 09:51:24 Process  0 in queue
2022/10/20 09:51:27 Reply arrived from process  1
2022/10/20 09:51:28 Process  1 in queue
2022/10/20 09:51:29 Reply arrived from process  2
2022/10/20 09:51:29 Process  2 in queue
2022/10/20 09:51:31 Reply arrived from process  3
2022/10/20 09:51:31 Critic section entered
2022/10/20 09:51:32 Process  3 in queue
2022/10/20 09:51:36 Exiting critic section
2022/10/20 09:51:36 Reply sent to processs 3
2022/10/20 09:51:36 Reply sent to processs 2
2022/10/20 09:51:37 Reply sent to processs 1
2022/10/20 09:51:37 Reply sent to processs 0
2022/10/20 09:51:37 Trying to enter critic section
2022/10/20 09:51:37 Request sent to process  0
2022/10/20 09:51:37 Request sent to process  1
2022/10/20 09:51:37 Request sent to process  2
2022/10/20 09:51:37 Request sent to process  3
2022/10/20 09:51:41 Reply arrived from process  0
2022/10/20 09:51:41 Process  0 in queue
2022/10/20 09:51:50 Reply arrived from process  1
2022/10/20 09:51:51 Process  1 in queue
2022/10/20 09:52:00 Reply arrived from process  2
2022/10/20 09:52:00 Process  2 in queue
2022/10/20 09:52:09 Reply arrived from process  3
2022/10/20 09:52:09 Critic section entered
2022/10/20 09:52:09 Process  3 in queue
2022/10/20 09:52:14 Shutdown signal caught, peer service will stop
2022/10/20 09:52:17 Exiting critic section
```

Grazie per l’attenzione