Exclusão mútua distribuída

Paulo Augusto Gomes Kataki

paulogkataki@hotmail.com

Instituto de Informática
Universidade Federal de Goiás

12 de Julho de 2019

Exclusão mútua distribuída

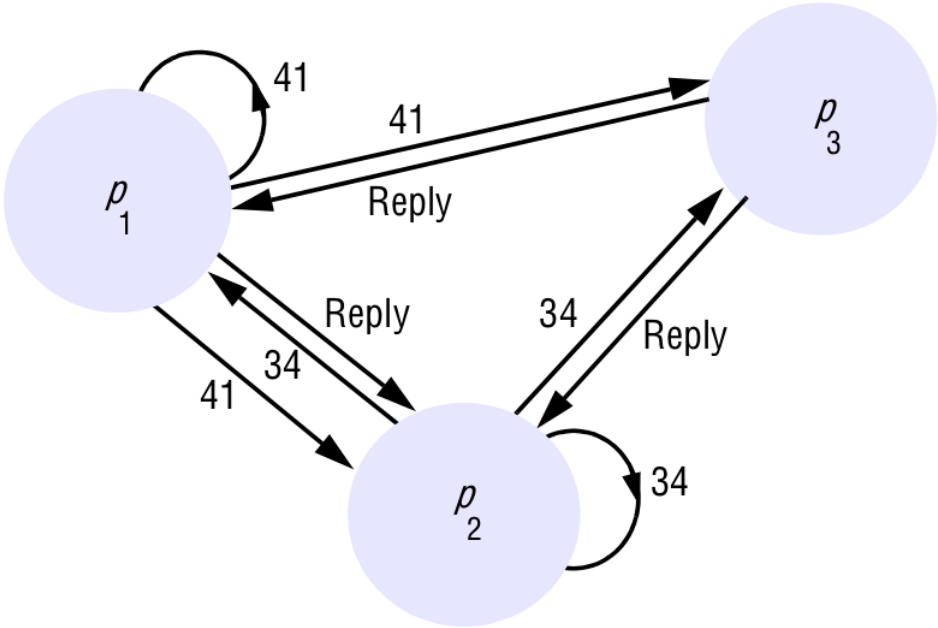
Requisitos essenciais para exclusão mútua

1. No máximo um processo pode acessar a região crítica por vez;
2. Pedidos para entrar e sair da região crítica eventualmente são sucedidos;
3. A ordem cronológica dos pedidos de entrada na região crítica deve ser mantida;

Algoritmo de Ricart e Agrawala

Algoritmo de Ricart e Agrawala

- Utiliza somente troca de mensagem entre os processos;
- Utiliza multicast e relógios lógicos de Lamport;
- Garante todas as 3 restrições;
- Cada processo necessita de $2(N-1)$ mensagens para entrar na seção crítica;
- Tempo de sincronização dos processos é somente na transmissão das mensagens.


41
41
Reply
p_1
p_3
Reply
34
Reply
34
41
p_2
34

Ricart and Agrawala's algorithm

On initialization

state := RELEASED;

To enter the section

state := WANTED;
Multicast *request* to all processes;
T := request's timestamp;

Request processing deferred here

Wait until (number of replies received = $(N-1)$);
state := HELD;

On receipt of a request $<T_i, p_i>$ *at* p_j $(i \neq j)$

if (*state* = HELD *or* (*state* = WANTED *and* $(T, p_j) < (T_i, p_i)$))

then

queue *request* from p_i without replying;

else

reply immediately to p_i;

end if

To exit the critical section

state := RELEASED;

reply to any queued requests;

Implementação

Go

- Linguagem com foco em concorrência;
- Goroutines: threads "leves";
- Channels: canais de sincronização das goroutines por transmissão de mensagem.

Implementação

```
func (p *process) sendMessage(typeMessage int, address string) {
    msg := message{
        Timestamp:        p.timestamp,
        RequestTimestamp: p.requestTimestamp,
        TypeMessage:      typeMessage,
        Address:          p.address,
        Id:               p.id,
    }

    // sending message to address's channel
    p.channels[p.getIndexFromAddress(address)] <- msg
}

func (p process) doMulticast(typeMessage int) {
    p.updateTimestamp(p.timestamp)

    //send message to all processes
    for _, address := range p.processesAddresses {
        if address != p.address {
            go p.sendMessage(typeMessage, address)
        }
    }
}
```

Implementação

Algoritmo de Ricart e Agrawala

- Linguagem utilizada: Go;
- Envio de mensagens: Socket;
- Região crítica: Servidor;
- Utilização de *goroutines*, *channels* e semáforos da linguagem Go.

Exclusão mútua distribuída

Paulo Augusto Gomes Kataki

paulogkataki@hotmail.com

Instituto de Informática
Universidade Federal de Goiás

12 de Julho de 2019