Algoritmos Distribuídos de Eleição

Dhruv Babani, Eduardo Bregalda, Eduardo Barcellos

Outubro 2023

Abstract

Algoritmo distribuído de eleição simulado em Go - cenários de sucesso e falha e como essas são tratadas e reportadas.

1 Problema de eleição

Em sistemas distribuídos, nos quais processos em paralelos executam determinadas tarefas, um líder é responsável por tarefas especiais, como processar solicitações do sistema, atribuir tarefas para outros processos, gerenciar dados e reportar estados do sistema.

Na ferramenta Apache ZooKeeper, por exemplo, apenas o líder processa requisições de escritas, propagando a requisição para os outros nodos do sistema. Nesse sentido, quando o sistema é inicializado pela primeira vez ou o líder atual falha, há a necessidade da eleição de um novo líder.

2 Algoritmo em Anel

Há diversos algoritmos destinados à eleição do novo líder, entre eles, o Algoritmo em Anel e o Algoritmo de Garcia-Molina (Bully).

O Algoritmo em Anel, implementado aqui, assume que todos os processos estão conectados em forma de anel unidirecional.

Quando algum nodo percebe a necessidade de uma nova eleição, ele envia uma mensagem para o próximo nodo do anel, e essa é propagada pelo anel por todos os nodos até chegar novamente no primeiro. Nessa primeira etapa, cada nodo inseriu na mensagem o seu PID (Process Identification).

Dessa forma, o nodo que comec¸ou a eleição determina que o novo líder será o nodo com maior PID. Determinando o novo líder, o mesmo nodo propaga uma nova mensagem, informando todos os outros sobre o novo líıder.

3 Implementação

A implementação do Algoritmo em Anel utilizou quatro processos paralelos conectados em formato de anel e um processo Controle para estimular a necessidade de eleição de um novo líder. Implementado em Go, os processos foram interligados por canais bloqueantes (tamanho 0).

O processo Controle envia a mensagem de eleição ao processo 0 e esse torna-se responsável pelas duas etapas do processo de eleição.

A mensagem eleição consiste, essencialmente, em uma string validada ao final do processo, um vetor de PIDS para que os nodos insiram seus PIDS e o PID do novo eleito sendo outros atributos necessários para simulações e comunicação com o processo Controle.

4 Testes, falhas e Nodo Controle

4.1 Contexto

PIDs randômicos são escolhidos para os quatro processos. Dois dos PIDs gerados são enviados ao Nodo Controle, a fim de atuarem como o comandante (responsável pela eleição) e o Nodo Inoperante (na primeira ou segunda etapa).

Além disso, um Nodo Coringa é escolhido, a fim de alterar a mensagem que terá sua consistência verificada ao final do processo.

O Nodo Controle cria a mensagem de eleição que será injetada no Nodo Comandante. A mensagem de eleição possui atributos para os nodos escolhidos tornarem-se inoperante e coringa quando a receberem, impedindo o fluxo do anel e alterando a mensagem, respectivamente.

4.2 Tolerância à falha

4.2.1 Tempo Excedido

Quando um nodo se torna inoperante durante o processo de eleição, o anel entra em estado de deadlock, uma vez que uns aguardam uma mensagem que não será enviada.

Portanto, para não ocorrer deadlock, foram inseridos sistemas de time out para todos os nodos, que finalizam seus processos ao final do tempo.

Em especial, quando o tempo de espera do Nodo Comandante é alcançado, ele reporta ao Nodo Controle sobre a falha e também encerra seu processo.

4.2.2 Consistência da Mensagem

Além disso, para verificação da consistência das informações enviadas inicialmente pela mensagem de eleição, o processo comandante realiza uma operação de ”checksum” em cima de um vetor de bytes - esse que não deve ser alterado ao longo do anel.

Ao final do processo de eleição, reporta o resultado da verificação para o Nodo Controle, identificando assim, uma possível alteração nos dados recebidos.

5 Dumps

5.1 Tempo Excedido

```
Control Process says: "Commander is 271."
Control Process says: "Election message sent!"
271 says: "I've received a message of purpose 1: [ -1, -1, -1, -1 ]"
271 says: "I've sent a message to the next node"
259 says: "I've received a message of purpose 1: [ 271, -1, -1, -1 ]"
259 says: "I've sent a message to the next node"
254 says: "I've received a message of purpose 1: [ 271, 259, -1, -1 ]"
254 says: "I've sent a message to the next node"
463 says: "I've received a message of purpose 1: [ 271, 259, 254, -1 ]"
463 says: "I've sent a message to the next node"
271 says: "I've received a message of purpose 1, [ 271, 259, 254, 463 ]"
271 says: "I've gathered all PIds and sent confirmation to control"
271 says: "I've sent foward the elected node for the ring"
259 says: "I've received a message of purpose 2, [ 271, 259, 254, 463 ]"
259 says: "I've just received the new elected node: 463"
259 says: "I've sent forward who is the elected node."
259 says: "My job is done here. Later!"
254 says: "I've received a message of purpose 2, [ 271, 259, 254, 463 ]"
254 says: "I've just received the new elected node: 463"
254 says: "I've sent forward who is the elected node."
254 says: "My job is done here. Later!"
463 says: "I've received a message of purpose 2, [ 271, 259, 254, 463 ]"
463 says: "I've just received the new elected node: 463"
463 says: "Oh yeeah! I'm the elected node!"
463 says: "I've sent forward who is the elected node."
463 says: "My job is done here. Later!"
Control Process says: "All PIds gathered. Waiting for confirmation..."
271 says: "Everyone knows about the new elected. I've sent a successfull confirmation to control."
271 says: "My job is done here. Later!"
Control Process says: "Election confirmed. Elected PId: 463."
```

Figure1:Sucesso na eleição

```
Control Process says: "Commander is 135."
Control Process says: "Commander cannot be dead at gathering!"
```

Figure 2: Nodo Comandante não pode estar morto

```
Control Process says: "Commander is 764."
764 says: "I've received a message of purpose 1: [ -1, -1, -1, -1 ]"
764 says: "I've sent a message to the next node"
832 says: "I've received a message of purpose 1: [ 764, -1, -1, -1 ]"
832 says: "I've sent a message to the next node"
240 says: "My job is done here. Later!"
Control Process says: "Election message sent!"
832 says: "My job is done here. Later!"
764 says: "I've just reported to Control that someone died in PId gathering."
764 says: "My job is done here. Later!"
Control Process says: "Failed in gathering all PIds"
129 says: "My job is done here. Later!"
```

Figure 3: Falha ao coletar PIds

```
Control Process says: "Commander is 272."
272 says: "I've received a message of purpose 1: [ -1, -1, -1, -1 ]"
272 says: "I've sent a message to the next node"
319 says: "I've received a message of purpose 1: [ 272, -1, -1, -1 ]"
319 says: "I've sent a message to the next node"
829 says: "I've received a message of purpose 1: [ 272, 319, -1, -1 ]"
829 says: "I've sent a message to the next node"
938 says: "I've received a message of purpose 1: [ 272, 319, 829, -1 ]"
938 says: "I've sent a message to the next node"
272 says: "I've received a message of purpose 1, [ 272, 319, 829, 938 ]"
Control Process says: "Election message sent!"
Control Process says: "All PIds gathered. Waiting for confirmation..."
272 says: "I've gathered all PIds and sent confirmation to control"
272 says: "I've sent foward the elected node for the ring"
319 says: "I've received a message of purpose 2, [ 272, 319, 829, 938 ]"
319 says: "I've just received the new elected node: 938"
319 says: "I've sent forward who is the elected node."
319 says: "My job is done here. Later!"
829 says: "I've received a message of purpose 2, [ 272, 319, 829, 938 ]"
829 says: "My job is done here. Later!"
272 says: "I've just reported to Control that someone died in election confirmation."
272 says: "My job is done here. Later!"
Control Process says: "Election failed in confirmation"
938 says: "My job is done here. Later!"
```

Figure4:Falha a escolha do eleito

5.2 Consistência da Mensagem

```
Control Process says: "Commander is 185."
Control Process says: "Election message sent!"
185 says: "I've received a message of purpose 1: [ -1, -1, -1, -1 ]"
185 says: "I've sent a message to the next node"
389 says: "I've received a message of purpose 1: [ 185, -1, -1, -1 ]"
389 says: "I've sent a message to the next node"
128 says: "I've received a message of purpose 1: [ 185, 389, -1, -1 ]"
128 says: "I've sent a message to the next node"
579 says: "I've received a message of purpose 1: [ 185, 389, 128, -1 ]"
579 says: "I've sent a message to the next node"
185 says: "I've received a message of purpose 1, [ 185, 389, 128, 579 ]"
185 says: "I've gathered all PIds and sent confirmation to control"
185 says: "I've sent foward the elected node for the ring"
Control Process says: "All PIds gathered. Waiting for confirmation..."
389 says: "I've received a message of purpose 2, [ 185, 389, 128, 579 ]"
389 says: "I've just received the new elected node: 579"
389 says: "I've sent forward who is the elected node."
389 says: "My job is done here. Later!"
128 says: "I've received a message of purpose 2, [ 185, 389, 128, 579 ]"
128 says: "I've just received the new elected node: 579"
128 says: "I've sent forward who is the elected node."
128 says: "My job is done here. Later!"
579 says: "I've received a message of purpose 2, [ 185, 389, 128, 579 ]"
579 says: "I've just received the new elected node: 579"
579 says: "Oh yeeah! I'm the elected node!"
579 says: "I've sent forward who is the elected node."
579 says: "My job is done here. Later!"
185 says: "Everyone knows about the new elected. I've sent a successfull confirmation to control."
185 says: "I'm executing the payload checksum, expected: 0cbc6611f5540bd0809a388dc95a615b current: 0cbc6611f5540bd0809a388dc95a615b."
185 says: "My job is done here. Later!"
Control Process says: "Election confirmed. Elected PId: 579."
Control Process says: "Payload checksum operated successfully!"
```

Figure 5: Sucesso na operação de checksum

```
Control Process says: "Commander is 424."
Control Process says: "Election message sent!"
424 says: "I've received a message of purpose 1: [ -1, -1, -1, -1 ]"
424 says: "I've sent a message to the next node"
396 says: "I've received a message of purpose 1: [ 424, -1, -1, -1 ]"
396 says: "I've sent a message to the next node"
937 says: "I've received a message of purpose 1: [ 424, 396, -1, -1 ]"
937 says: "I've sent a message to the next node"
412 says: "I've received a message of purpose 1: [ 424, 396, 937, -1 ]"
412 says: "I've sent a message to the next node"
424 says: "I've received a message of purpose 1, [ 424, 396, 937, 412 ]"
424 says: "I've gathered all PIds and sent confirmation to control"
424 says: "I've sent foward the elected node for the ring"
Control Process says: "All PIds gathered. Waiting for confirmation..."
396 says: "I've received a message of purpose 2, [ 424, 396, 937, 412 ]"
396 says: "I've just received the new elected node: 937"
396 says: "I've sent forward who is the elected node."
396 says: "My job is done here. Later!"
937 says: "I've received a message of purpose 2, [ 424, 396, 937, 412 ]"
937 says: "I've just received the new elected node: 937"
937 says: "Oh yeeah! I'm the elected node!"
937 says: "I've sent forward who is the elected node."
937 says: "My job is done here. Later!"
412 says: "I've received a message of purpose 2, [ 424, 396, 937, 412 ]"
412 says: "I've just received the new elected node: 937"
412 says: "I've sent forward who is the elected node."
412 says: "My job is done here. Later!"
424 says: "Everyone knows about the new elected. I've sent a successfull confirmation to control."
424 says: "I'm executing the payload checksum, expected: 0cbc6611f5540bd0809a388dc95a615b current: df02a235d48209db07e6b5162528c483."
424 says: "My job is done here. Later!"
Control Process says: "Election confirmed. Elected PId: 937."
Control Process says: "Payload checksum failed!"
```

Figure 6: Identificação da mensagem alterada

6 Códigos-Fonte

```
package main

import (
    "crypto/md5"
    "fmt"
    "math/rand"
    "os"
    "sync"
    "time"
)

var (
    mutex            sync.Mutex
    chans            = []chan election_message{make(chan election_message), make(chan
election_message), make(chan election_message), make(chan election_message)}
    control          = make(chan int)
    wg               sync.WaitGroup
    expected_payload = []byte("Test")
)

type election_message struct {
    purpose              int
    pids                 [4]int
    index                int
    elected              int
    commander            int
    dead_at_gathering    int
    dead_at_confirmation int
    payload              []byte
    joker_id             int
}

func ElectionControler(in chan int, commander int, dead_at_gathering int,
dead_at_confirmation int, payload []byte, joker_id int) {

    // Injeta uma mensagem de eleicao e aguarda o retorno do novo eleito
    // Eleicao consiste em duas etapas: 1 -> coleta de prioridades, 2 -> anuncio do novo
eleito,
    // Se a etapa de coleta for bem sucedida, controle recebe 0, caso contrario, -1
    // Se a etapa de anuncio do novo eleito for bem sucedida, controle recebe o id do novo
eleito, caso contrario, -1

    defer wg.Done()

    var em election_message

    em.purpose = 1
    em.pids[0] = -1
    em.pids[1] = -1
    em.pids[2] = -1
```

```
    em.pids[3] = -1
    em.index = 0
    em.elected = -1
    em.commander = commander
    em.dead_at_gathering = dead_at_gathering
    em.dead_at_confirmation = dead_at_confirmation
    em.payload = payload
    em.joker_id = joker_id

    fmt.Printf("Control Process says: \"Commander is %d.\"\n", em.commander)

    if dead_at_gathering == em.commander {
        fmt.Printf("Control Process says: \"Commander cannot be dead at gathering!\" \n")
        os.Exit(0)
    }

    chans[3] <- em // pedir eleicao para o processo 0
    fmt.Printf("Control Process says: \"Election message sent!\" \n")

    gathering_result := <-in
    if gathering_result == 0 {
        fmt.Printf("Control Process says: \"All PIds gathered. Waiting for
confirmation...\"\n")
        confirmation_result := <-in
        payload_checksum := <-in
        if confirmation_result > 0 {
            fmt.Printf("Control Process says: \"Election confirmed. Elected PId: %d.\"\n",
confirmation_result)
        } else {
            fmt.Printf("Control Process says: \"Election failed in confirmation\"\n")
        }

        if payload_checksum == 1 {
            fmt.Printf("Control Process says: \"Payload checksum operated
successfully!\"\n")
        } else {
            fmt.Printf("Control Process says: \"Payload checksum failed!\"\n")
        }
    } else {
        fmt.Printf("Control Process says: \"Failed in gathering all PIds\"\n")
    }
}

func ElectionStage(PId int, in chan election_message, out chan election_message) {
    defer wg.Done()
    select {
    case pids_collect := <-in:
        if pids_collect.dead_at_gathering != PId {

            fmt.Printf("%2d says: \"I've received a message of purpose %d: [ %d, %d, %d,
%d ]\" \n", PId, pids_collect.purpose, pids_collect.pids[0], pids_collect.pids[1],
pids_collect.pids[2], pids_collect.pids[3])
```

```
            pids_collect.pids[pids_collect.index] = PId
            pids_collect.index += 1

            if pids_collect.joker_id == PId {
                pids_collect.payload = []byte("Hahaha!")
            }

            out <- pids_collect
            fmt.Printf("%2d says: \"I've sent a message to the next node\" \n", PId)

            select {
            case confirmation_msg := <-in:

                fmt.Printf("%2d says: \"I've received a message of purpose %d, [ %d, %d,
%d, %d ]\"\n", PId, confirmation_msg.purpose, confirmation_msg.pids[0],
confirmation_msg.pids[1], confirmation_msg.pids[2], confirmation_msg.pids[3])

                if pids_collect.commander == PId {
                    control <- 0 // Success

                    fmt.Printf("%2d says: \"I've gathered all PIds and sent confirmation
to control\" \n", PId)

                    var elected = -1
                    for i := 0; i < 4; i++ {
                        if confirmation_msg.pids[i] > elected {
                            elected = confirmation_msg.pids[i]
                        }
                    }

                    confirmation_msg.purpose = 2
                    confirmation_msg.elected = elected

                    if confirmation_msg.elected == PId {
                        fmt.Printf("%2d says: \"Oh yeeah! I'm the elected node!\"\n", PId)
                    }

                    out <- confirmation_msg

                    fmt.Printf("%2d says: \"I've sent foward the elected node for the
ring\" \n", PId)

                    select {
                    case finish_msg := <-in:
                        control <- finish_msg.elected // Elected node PId
                        fmt.Printf("%2d says: \"Everyone knows about the new elected. I've
sent a successfull confirmation to control.\" \n", PId)

                        expected_checksum := md5.Sum([]byte(expected_payload))
                        current_checksum := md5.Sum([]byte(finish_msg.payload))

                        payload_checksum := expected_checksum == current_checksum
```

```
                        fmt.Printf("%2d says: \"I'm executing the payload checksum,
expected: %x current: %x.\"\n", PId, expected_checksum, current_checksum)

                        if payload_checksum {
                            control <- 1
                        } else {
                            control <- 0
                        }
                    case <-time.After(5 * time.Second):
                        control <- -1
                        fmt.Printf("%2d says: \"I've just reported to Control that someone
died in election confirmation.\"\n", PId)
                    }
                } else {
                    if pids_collect.dead_at_confirmation != PId {
                        fmt.Printf("%2d says: \"I've just received the new elected node:
%d\"\n", PId, confirmation_msg.elected)
                        if confirmation_msg.elected == PId {
                            fmt.Printf("%2d says: \"Oh yeeah! I'm the elected node!\"\n",
PId)
                        }
                        out <- confirmation_msg
                        fmt.Printf("%2d says: \"I've sent forward who is the elected
node.\"\n", PId)
                    }
                }
            case <-time.After(5 * time.Second):
                if pids_collect.commander == PId {
                    control <- -1
                    fmt.Printf("%2d says: \"I've just reported to Control that someone
died in PId gathering.\"\n", PId)
                }
            }
        }
    case <-time.After(5 * time.Second):
    }
    fmt.Printf("%2d says: \"My job is done here. Later!\"\n", PId)
}

func main() {

    wg.Add(5)

    r := rand.New(rand.NewSource(time.Now().UnixNano()))
    id_1, id_2, id_3, id_4 := 0, 0, 0, 0

    // Avoid same ids
    for {
        id_1, id_2, id_3, id_4 = r.Intn(1000), r.Intn(1000), r.Intn(1000), r.Intn(1000)
        if id_1 != id_2 && id_1 != id_3 && id_1 != id_4 && id_2 != id_3 && id_2 != id_4 &&
id_3 != id_4 {
```

```
            break
        }
    }

    go ElectionStage(id_1, chans[3], chans[0])
    go ElectionStage(id_2, chans[0], chans[1])
    go ElectionStage(id_3, chans[1], chans[2])
    go ElectionStage(id_4, chans[2], chans[3])

    fmt.Println("\n   Anel de processos criado")

    go ElectionControler(control, id_1, -1, -1, expected_payload, -1)

    fmt.Print("\n   Processo controlador criado\n\n")

    wg.Wait()
}
```