INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e de Gestão

Sistemas Distribuídos

Eleição Distribuída

Trabalho Prático nº3

Pedro Ribeiro N 37557

Mário Machado N 37549

- Ano Letivo 2018/2019 -

Professor Rui Lopes

Índice

Introdução

A dependência de um servidor centralizado pode levantar alguns constrangimentos à operação continuada dos sistemas atuais. De facto, com o aumento de escala, a capacidade do servidor pode-se tornar rapidamente insuficiente, comprometendo todos os serviços que dele dependem.

Um dos procedimentos comuns nas aplicações modernas é a exigência de um processo coordenador, iniciador ou que desempenhe uma tarefa específica. Como é bem sabido, a coexistência de mais que um processo deste tipo pode levar a inconsistências e a perda de informação.

Neste sentido, pode haver necessidade de coordenação entre vários processos, para que apenas um desempenhe este papel. Em sistemas distribuídos é necessário implementar algoritmos específicos com esse objetivo. Estes algoritmos são denomidados algoritmos de eleição.

Os algoritmos distribuídos de eleição muitas vezes precisam que um nó se comporte de maneira diferente, desempenhando assim uma função especifica e única para todo sistema (chamado de líder). Para escolher o nó existe um processo chamado de eleição que é executado por um algoritmo de eleição por todos os nós do sistema, dentre todos apenas um torna-se o líder através de um mecanismo de comunicação JGroups.

O líder pode ser escolhido de acordo com vários fatores, dentre eles pode-se citar como exemplo: um endereço IP, endereço físico do nó, quantidade de processamento ou qualquer identificação única, para que cada nó seja distinto. Ao abordar o assunto de algoritmos de eleição vamos abordar com mais enfase o Algoritmo de Bully.

Análise

Algoritmo Bully

O algoritmo de Bully recebe esse nome porque o nó mais robusto/maior vence a eleição e torna-se líder. Antes que a eleição tenha início há duas coisas a ter em conta, cada nó tem uma identificação única e cada nó conhece os vizinhos maiores que ele próprio. Então, um nó ativo qualquer vai perceber se não há nenhum líder ativo para poder iniciar uma eleição. O nó que inicia a eleição envia uma mensagem de "ELEIÇÃO" para todos os nós maiores que ele, ativos ou não e aguarda uma resposta positiva. Caso ele receba ao menos uma resposta, seu processo de eleição é finalizado e um outro nó vai continuar com a eleição. Porém, caso nenhum nó responda, ele torna-se o líder.

Um nó pode receber a mensagem de eleição a qualquer momento, quando este recebe envia um "OK" de volta como resposta e toma o poder de fazer a eleição novamente.

Assim em algum momento todos os nós maiores do que o nó que iniciou a eleição vão obter o poder e ter a possibilidade de iniciar também a eleição, após receberem as respostas "OK", o nó maior em alguma hora vai realizar o mesmo processo e então vencer a eleição e mandar a mensagem "LÍDER" a todos os nós do sistema.

JGroups

É uma biblioteca *Java* para usufruir de comunicação em grupo. Usa um *JChannel* como API (Application Programming Interface) principal para se conectar a um cluster, para poder enviar e receber mensagens. Permite também que haja notificações quando um membro se junta, ou um membro sai ou quando existem falhas.

As mensagens enviadas contêm o endereço do remetente e do recetor. Os endereços são subclasses de org.jgroups.Address, e geralmente contêm um endereço IP além de uma porta.

Em *JGroups* existem mais duas áreas:

- **blocos de construção**: os blocos de construção são as classes que residem no topo de uma *JChannel* que proporcionam um nível de abstração mais elevado.
- **pilha do protocolo:** a pilha de protocolo, sendo a caraterística mais poderosa do *JGroups*, é o intermediário pelo qual cada mensagem passa e permite a personalização completa, os protocolos podem ser configurados, removidos, substituídos, melhorados ou novos protocolos podem ser escritos e adicionados à pilha para coincidirem com as exigências da aplicação e as caraterísticas da rede.

Em suma, *JGroups* é um *middleware* para comunicação confiável *multicast* em *Java*. *JGroups* fornece primitivas de comunicação de baixo nível tais como o transporte de mensagens e membros do grupo. As funções de comunicação de alto nível são a troca de mensagens síncronas ou exclusão mútua distribuída.

Diagramas

- Modo de Eleição

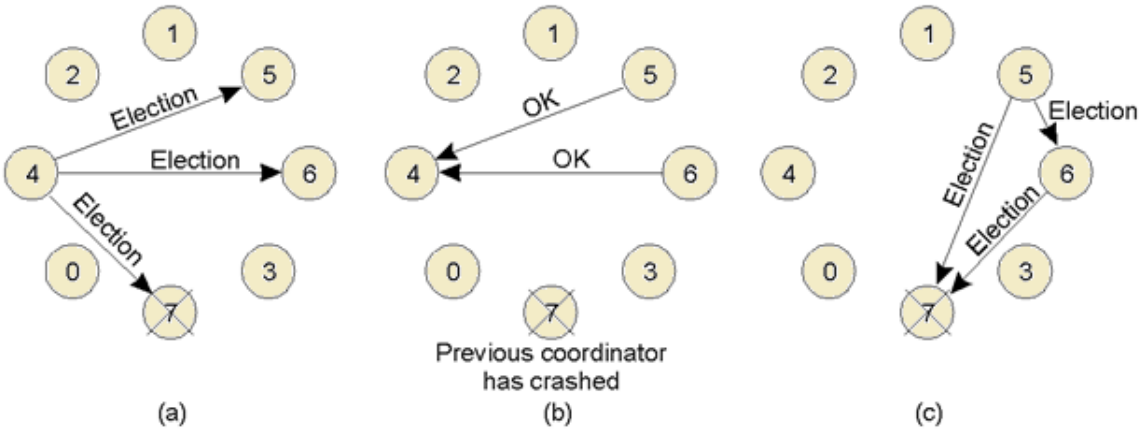

Figura 5 – Modo de Eleição

- Diagrama de Exemplificação

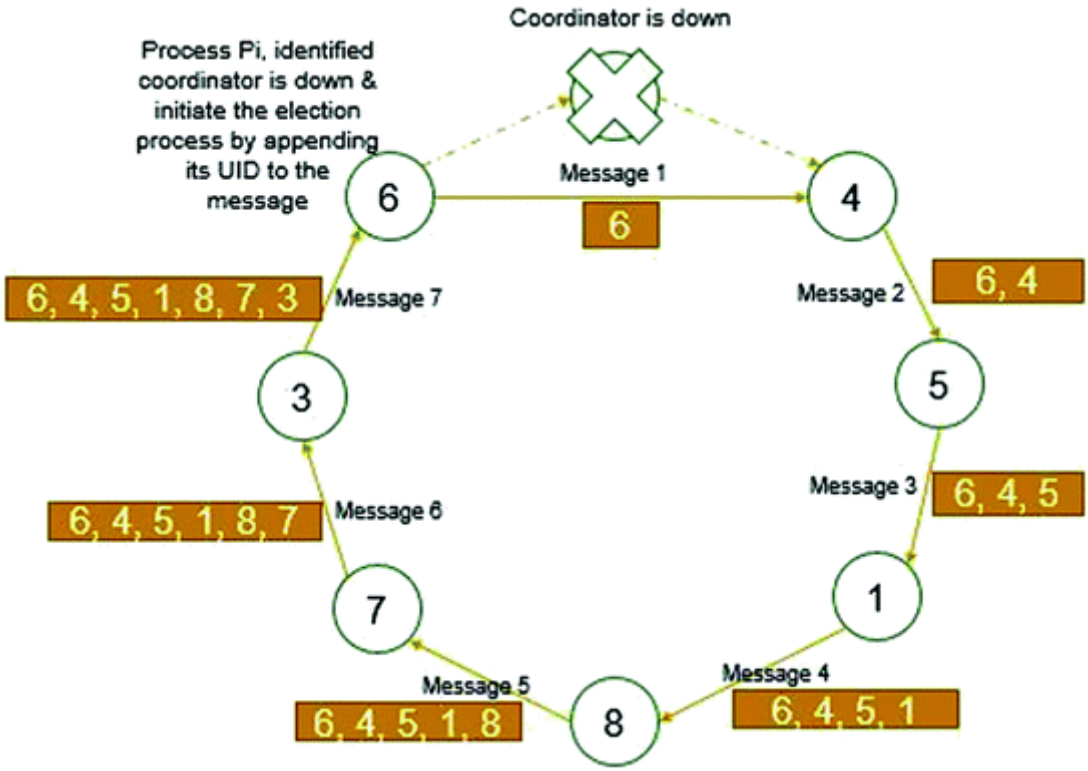

Figura 4 – Diagrama de Exemplificação

Implementação e Resultados

A execução do sistema inicia-se com a criação de um cluster em *JGroups* e a adesão ao mesmo de cada *peer* que seja executado. Cada *peer* é iniciado a partir do momento que se corre a aplicação.

De forma a iniciar a eleição no sistema, é utilizado o botão *Election* presente na interface gráfica da aplicação. Após o pedido de eleição no sistema este vai percorrer todos os processos presentes comparando-os para ver qual o maior e mostrando na consola o processo coordenador.

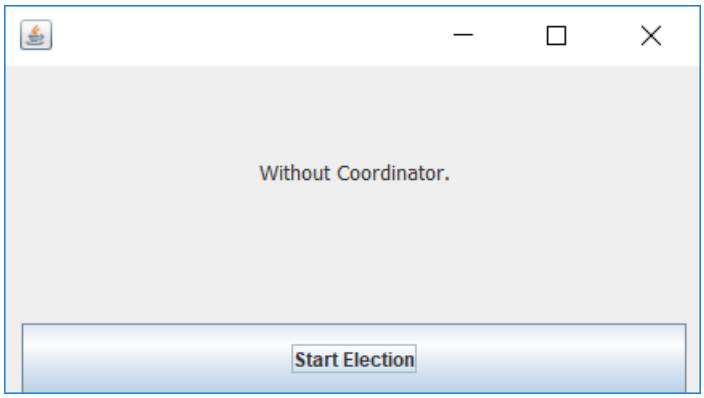

Figura 1 - Inicialização Aplicação

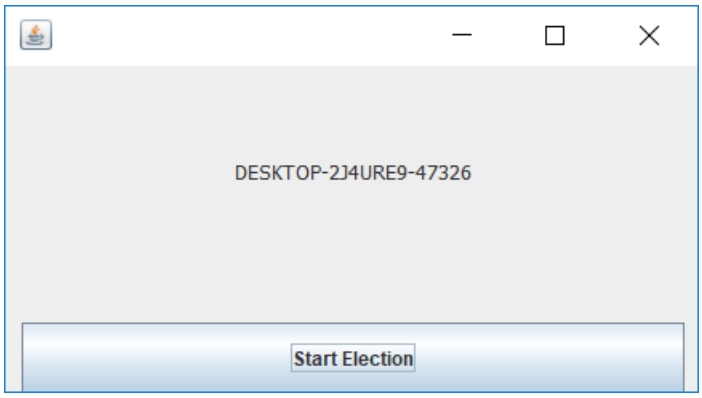

Figura 2 - Depois da sua Eleição

```
-------------------------------------------------------------------
GMS: address=DESKTOP-2J4URE9-47326, cluster=Election, physical address=192.168.1.158:55289
-------------------------------------------------------------------
```

Figura 3 - Consola

Conclusão

A realização deste trabalho ajudou-nos perceber o funcionamento do *JGroups* e do Algoritmo *Bully*, bem como as suas vantagens e desvantagens num Sistema Distribuído. Neste sentido, acreditamos que foram cumpridos os principais objetivos, ainda que existam algumas falhas que não conseguimos resolver.