Eleição de líder em redes móveis Ad Hoc
Implementação de um algoritmo

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Sistemas Distribuídos 2018/2019
António Pereira — up201404797@fe.up.pt
Carlos Neto — up201405711@fe.up.pt
Tiago Pinto — up201405965@fe.up.pt

***Resumo*—Aquando a deteção de uma falha de um nó, pode-se proceder à reorganização da rede, começando por eleger um líder/coordenador. Desta forma, a rede deve adaptar-se de forma a continuar a operar de acordo com a função para o qual o sistema foi desenvolvido. As aplicabilidades dos algoritmos de eleição são variadas. Entre elas podemos encontrar este tipo de algoritmo em:**

- **Servidores replicados com com replicação passiva (primary-backup);**
- **Sincronização de relógios (um nó dita o tempo).**

Foi implementado um algoritmo de eleição de líder utilizando linuagem Java®, sobre UDP/IP. Além disso, também foi implementado um algoritmo de criação da rede e de emulação da camada de transporte.

***Palavras-chave*—Sistemas Distribuídos, algoritmo, eleição.**

I. Introdução

O problema da eleição de uma máquina inserida num sistema distribuído (doravante designada por nó), como líder ou coordenador já é estudado há vários anos. Um dos principais entusiastas nesta área, H. Garcia-Molina, em 1982 identificou ([1]) diversos problemas que o processo de eleição de um nó como líder pode causar, dos quais se destaca:

> *"Como pode o protocolo de eleição lidar com falhas durante a própria eleição?"*

Esta questão torna-se particularmente pertinente quando se trata de uma rede móvel que é intrinsecamente dinâmica (isto é, com frequentes mudanças de topologia). Apesar de que, tal como identificado em [1], caso as falhas de nós sejam frequentes, o custo da reorganização do sistema e escolha de um novo líder não trará vantagens sobre outras possíveis soluções para o problema. No entanto, existem aplicações no contexto de redes móveis que fazem uso de protocolos de eleição de líder (p.e., [2], [3], [4], [5]).

S. Vasudevan *et al.* [6] propõem um algoritmo de eleição de líder em redes móveis Ad Hoc. Este algoritmo usa um conceito de difusão computacional, o que permite a eleição de um nó tendo em conta a sua aptidão. Esse algoritmo é capaz de tolerar múltiplas eleições ocorrerem de forma concorrente, de forma assíncrona e com alterações arbitrárias da topologia da rede. Além disso, provam que esse algoritmo converge para um estado estável após um período finito de tempo.

II. Objetivos

Este projeto encontra-se inserido no âmbito da unidade curricular de Sistemas Distribuídos do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto.

De forma a estudar o problema apresentado no ponto anterior, o algoritmo descrito em [6] foi implementado e, sobre este, foi realizada uma análise crítica dos resultados obtidos. Assim, definem-se como objetivos:

1) Compreensão do conceito de sistema distribuído;
2) Perceção do problema de eleição de um nó líder ou coordenador;
3) Implementação do algoritmo de eleição;
4) Implementação de uma camada de emulação da rede para modelação do mundo real;
5) Implementação de uma camada capaz de gerar uma topologia de rede fixa ou aleatória tendo em conta a densidade $n_{vizinhos}/n_{nosRede-1}$ e a probabilidade de novos vizinhos (que representa a probabilidade de um nó hipotético apresentar mais do que um vizinho) de cada nó;
6) Apresentação dos resultados obtidos aos testes efetuados;

III. Metodologia

O algoritmo foi implementado com recurso à linguagem de programação Java®, na versão Java *Standart Edition* (SE) 8, permitindo tornar o código independente do sistema operativo utilizado. O ambiente de desenvolvimento integrado (IDE) foi o Eclipse®, na versão 2018-12.

Foi utilizado o software de controlo de versões Git®, e, de forma a permitir o desenvolvimento em paralelo pelos autores, utilizou-se o GitHub® como repositório remoto. Este software foi utilizado integrado no IDE.

IV. Algoritmo de eleição

O algoritmo apresentado por Vasudevan *et al.* em [6] consiste na eleição de um nó, considerado o mais apto, através de operações assíncronas, sendo capaz de tolerar mudanças na topologia da rede. Os autores apresentam este algoritmo e modelam o sistema como um grafo não direcionado (em que cada elemento representa um nó e cada ligação representa a capacidade de dois nós comunicarem diretamente). Além disso, fazem as seguintes assunções:

- **Valor do nó** - cada nó tem um valor associado que indica a sua adaptação ao papel de líder. Este valor pode depender de qualquer característica relevante para o sistema, como p.e., a autonomia ou o poder computacional;
- Cada nó possui um **identificador único**;
- **Comunicação bidirecional e ordenada**;
- **Comportamento arbitrário** do nó;
- Caso dois **nós** capazes de comunicar entre si se **mantenham ativos**, é **certa** a **troca de mensagens**.
- Um nó é capaz de receber mensagens garantindo **espaço suficiente** para **não** sofrer ***overflow*** **do** ***buffer*** **de receção**.

A. *Descrição do algoritmo*

Numa rede em que se assume que os nós podem falhar, seja por avaria ou incapacidade de comunicar, quando um nó móvel se distancia suficientemente para se considerar desligado (isto é, incapaz de comunicar), o algoritmo é executado da seguinte forma:

1) Um nó inicia a execução do algoritmo de eleição de líder começando por construir uma *spanning tree* para a referida eleição. A *source* começa por difundir a mensagem **Election** para os seus vizinhos. Um nó, após a receção da mensagem **Election**, entra em estado de eleição e define o remetente da mensagem recebida como seu pai na *spanning tree*, reenviando a referida mensagem para todos os seus vizinhos, exceptuando o seu pai;
2) Quando um nó recebe uma mensagem **Election** de um vizinho que **não** seja o seu pai, responde imediatamente com uma mensagem **Ack**. Contudo, um dado nó não retorna imediatamente uma mensagem **Ack** para o seu pai. Em vez disso, espera até receber todas as mensagens **Ack** dos seus vizinhos (exceptuando o seu pai); A mensagem **Ack** que é retornada para o pai contém informação acerca do melhor candidato a ser *leader*, que tenha sido comunicado a este. Isto é alcançado através da atualização da informação do candidato a líder (par composto por: **valor computacional** e **ID**), sempre que seja recebida uma mensagem **Ack** que contenha um valor computacional superior ao valor computacional do candidato guardado no nó;
3) Para contemplar a falha de um nó e evitar que outro nó fique bloqueado à espera de um **Ack** do nó que falhou, o mecanismo de **Probe** e **Reply** responde a esse problema. A mensagem **Probe** é enviada após um *timeout* definido e a não receção, durante esse tempo, da mensagem **Ack**. Caso o nó vizinho não tenha falhado e apenas estiver à espera das respetivas mensagens **Ack** dos seus vizinhos, responde **Reply** ao seu nó pai para indicar que ainda se encontra presente na rede. Caso o nó vizinho tenha, de facto, falhado, o nó que se encontra à espera da mensagem de **Ack** remove o nó que falhou da lista de vizinhos e envia a mensagem de **Ack** em conjunto com o par **valor computacional** e **ID** para o seu nó pai.
4) Após a *source* receber todos as mensagens *Ack* dos seus vizinhos, anuncia o líder eleito para a rede, difundindo a mensagem **Leader** para os seus vizinhos; Estes reencaminharão a mensagem por todos os seus vizinhos, com exceção do remetente anterior, anunciando o líder assim a todos os nós na rede.
5) Uma vez concluído o processo de eleição, o nó líder deve anunciar periodicamente a rede a sua presença, através de mensagens **HeartBeat**. Caso essa mensagem não seja recebida após um *timeout*, o nó que detetar falha no líder inicia uma nova difusão computacional, i.e., uma nova eleição.

De realçar que, apesar da execução ser apresentada numa lista numerada dando a ideia de ordem de execução, na realidade a execução é assíncrona. Ou seja, a ação executada pelo nó é determinada como resposta a um evento, consoante o estado em que o nó se encontrar - entende-se estado como uma condição de guarda em que o nó despoleta uma ação de resposta de entre dum conjunto de ações possíveis ao evento que ocorreu.

V. Ambiente de simulação

Visando a criação de um ambiente de simulação, é necessário implementar um algoritmo de criação de uma rede aleatória. Este algoritmo tem em conta:

1) a probabilidade de um nó apresentar mais do que um vizinho - uma vez que apresentar um vizinho, pelo menos, é imposto pelo artigo que motiva este projeto;
2) a densidade de vizinhos de cada nó - representativa do número de vizinhos relativo ao número total de nós presentes na rede $N_{vizinhos}/N_{total-1}$.

Uma rede ilustrativa e resultante da execução do algoritmo pode ser encontrada na figura 1.

Para além da criação da rede, foi necessário criar um camada de simulação capaz de detetar quando um líder foi eleito e simular um número suficiente de vezes para os resultados apresentarem relevância estatística.

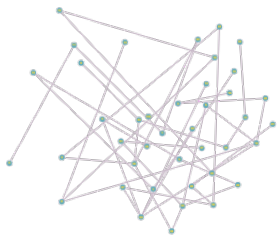

Figura 1. Topologia aleatória com probabilidade=20% e densidade=5%

A. *Emulação da rede*

Foi implementada uma camada de emulação da rede, que permite, com uma certa probabilidade, emular a perda e/ou atrasos em pacotes UDP.

VI. Resultados

De forma a avaliar o desempenho do algoritmo desenvolvido para a eleição de líder para redes moveis, foram avaliadas as seguintes métricas:

1) Tempo de convergência por eleição
2) Número de mensagens trocadas por nó, por eleição

Para cada métrica acima referida, foi feita a variação do número de nós da rede para verificar a evolução das métricas selecionadas com o aumento do número de nós.

Para emular a mobilidade dos nós, foi introduzido um atraso na comunicação entre dois nós, segundo uma distribuição uniforme, sendo que quanto maior o atraso na comunicação maior seria a distância entre dois nós.

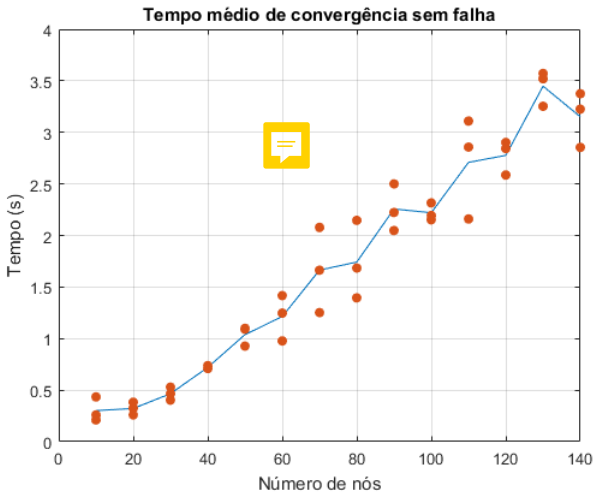

Figura 2. Tempo de convergência sem mobilidade

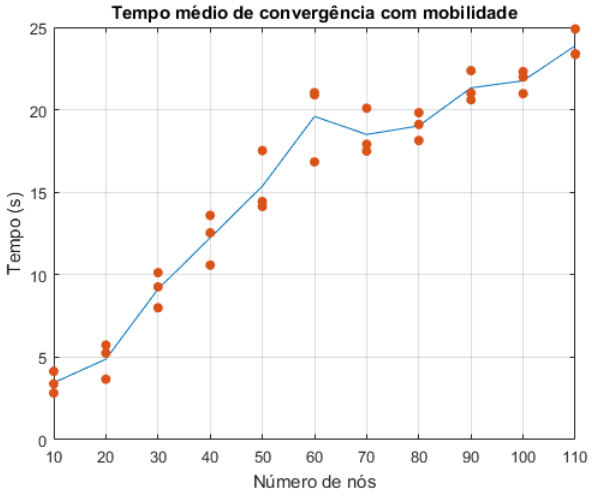

Figura 3. Tempo de convergência com mobilidade

Como é possível constatar nas figuras 2 e 3, o tempo de convergência da eleição aumenta linearmente com o número de nós na rede. Contudo, a mobilidade dos nós tem um impacto significativo no tempo de convergência da eleição devido ao facto da comunicação entre nós ser mais demorada e ser necessário os nós verificarem se os seus vizinhos continuam ativos.

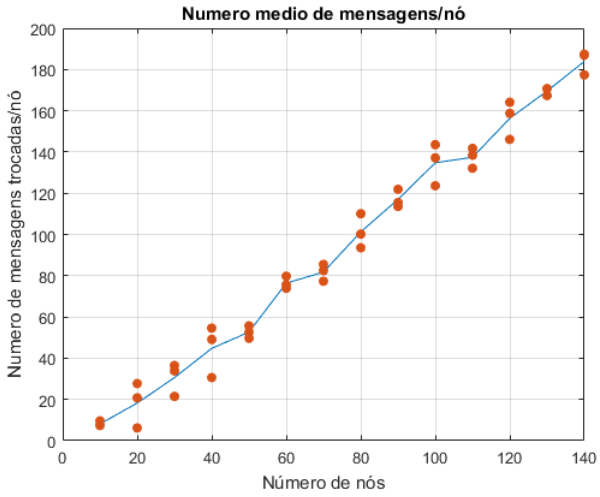

Figura 4. Número de mensagens trocadas por nó sem mobilidade

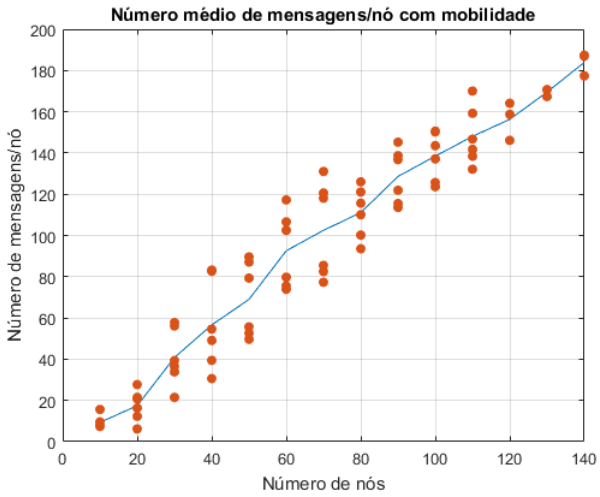

Figura 5. Número de mensagens trocadas por nó com mobilidade

Quanto ao número de mensagens trocadas, é possível constatar que o número de mensagens trocadas por nó aumenta linearmente com o numero de nós na rede independentemente de existir mobilidade ou não dos nós. Porém é importante notar que, quando existe mobilidade de nós, apesar de este também variar linearmente com o aumento de nós na rede, tem um *offset* relativamente à figura 4 que se deve às mensagens relativas ao mecanismo deteção de falha dos nós, nomeadamente as mensagens *Probe* e *Reply*,

VII. Gestão de projeto

A equipa de trabalho dividiu o sistema desenvolvido em três blocos, cada um com um conjunto de tarefas atribuídas:

1) Obtenção de resultados estatísticos;
2) Implementação do algoritmo;
   a) Modelação do nó;
   b) Resposta a mensagens;
   c) Implementação de mecanismos de deteção de falhas em nós. Tanto em processo de eleição (*Probe/Reply*), como quando o líder já foi eleito (*HeartBeat*).
3) Criação e análise da rede;

Graças à fácil comunicação entre os membros da equipa através de WhatsApp®, o que permitiu manter uma perceção geral do trabalho desenvolvido individualmente, e dessa forma, todos os problemas foram mitigados em reuniões de equipa, em que todos os membros trabalharam em conjunto. Assim sendo, a divisão entre tarefas tornou-se menos visível.

Analisando a divisão efetuada, e tendo em conta o ponto anterior, todos os membros declaram que trabalharam de forma idêntica, ou seja:

- António Pereira – 33%;
- Carlos Neto – 33%;
- Tiago Pinto – 33%;

VIII. Referências

[1] H. Garcia-Molina. Elections in a distributed computing system. In *IEEE Transactions on Computers*, volume C-31, pages 48–59, Feb 1982.

[2] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang. Secure group communications for wireless networks. In *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, volume 1, pages 113–117 vol.1, Oct 2001.

[3] Kostas P. Hatzis, George P. Pentaris, Paul G. Spirakis, Vasilis T. Tampakas, and Richard B. Tan. Fundamental control algorithms in mobile networks. In *Proceedings of the Eleventh Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '99, pages 251–260, New York, NY, USA, 1999. ACM.

[4] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8 - Volume 8*, HICSS '00, pages 8020–, Washington, DC, USA, 2000. IEEE Computer Society.

[5] Navneet Malpani, Jennifer L. Welch, and Nitin Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, DIALM '00, pages 96–103, New York, NY, USA, 2000. ACM.

[6] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, pages 350–360, Oct 2004.