An Asynchronous Leader Election Algorithm for Dynamic Networks

Implementação do algoritmo e testes de desempenho

1.º Nuno Adrego
up201403529@fe.up.pt

2.º Pedro Cova
up201403635@fe.up.pt

***Abstract*—This article was developed in the context of the Distributed Systems course, with a special focus on the implementation and analysis of a leader election algorithm. For this, an emulator was developed that implements the algorithm described in the article "An Asynchronous Leader Election Algorithm for Dynamic Networks" in order to test its performance against different metrics.**

***Index Terms*—distributed systems, leader election, dynamic networks**

I. Resumo

Em muitos algoritmos de sistemas distribuídos é necessário a existência de um nó com um papel especial de coordenação em relação aos restantes. Para tal, é necessário que esse nó seja eleito líder e que o seu estatuto seja conhecido e reconhecido por todos os nós.

Eleger autonomamente um líder num sistema distribuído não é fácil, mas o verdadeiro desafio aparece quando a rede é dinâmica, uma vez que estas têm a particularidade das ligação entre nós da rede serem ligados e desligados com muita frequência. Um exemplo típico de redes dinâmicas são as redes sem fios, uma vez que facilmente um utilizador pode sair e entrar da área de alcance.

Pretende-se então implementar o algoritmo para eleição de líder proposto no artigo "An Asynchronous Leader Election Algorithm for Dynamic Networks" [1] e avaliar o seu desempenho recorrendo a implementação de um emulador. Este emulador cria e manipula uma rede constituída por nós e links, onde os nós irão executar o algoritmo de eleição com o intuito de elegerem entre si um líder reconhecido por todos.

II. Definição do problema

A. Link

Cada link representa uma máquina de estados. Supondo que os nós "u" e "v" estão ligados por um link, existem seis estados possíveis: $[Up]$, $[GoingDown_u]$, $[GoingDown_v]$, $[Down]$, $[ComingUp_u]$ e $[ComingUp_v]$. As transições entre estado ocorrem por eventos que podem ser $\{LinkUp_u\}$, $\{LinkUp_v\}$, $\{LinkDown_u\}$, $\{LinkDown_v\}$.

Existem também duas filas de mensagens, $mqueue_{u,v}$ (do nó u para o nó v) e $mqueue_{v,u}$ (do nó v para o nó u). Sempre que um dos nós pretende enviar uma mensagem, esta é armazenada na fila se o link estiver no estado $[ComingUp]$ e só são enviadas quando o link está no estado $[Up]$. Em qualquer outro estado as mensagens são rejeitadas pelo link nenhum nó é notificado da perda de mensagens.

É importante salientar que os links são bi-direccionais, embora virtualmente o algoritmo possa atribuir uma direcionalidade.

B. Nó

Cada nó contém um vetor de *tuples* (estrutura de dados) em que cada posição desse vetor corresponde a um nó vizinho.

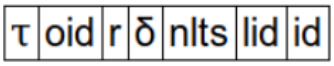

Fig. 1. Organização da estrutura *tuple*

Cada estrutura contém 7 elementos:

- τ - Timestamp de nova pesquisa de líder
- oid - Zero ou o ID do nó que começou a pesquisa
- r - Zero quando a pesquisa é iniciada ou um quando chega a um fim sem saída
- δ - distância até ao líder
- nlts - Timestamp da eleição do atual líder
- lid - ID do nó líder
- id - ID do nó

Nota: Uma das posições do vetor de *tupples* corresponde ao próprio nó

A discrição do algoritmo em si, encontra-se no seu relativo artigo [1], que já foi referido previamente.

Tendo em conta os aspetos anteriormente descritos, pretende-se implementar um método para avaliar o desempenho do algoritmo em diferentes topologias de rede. Perceber como varia o tempo de eleição e o número de mensagens trocadas com a dimensão da rede e com o número de ligações entre os vários nós da rede.

III. Descrição da solução

A. Abordagem escolhida

Para a resolução do problema apresentado, desenvolveu-se um programa que emula uma rede dinâmica e esta rede respeita as características definidas no artigo [1].

Existe então um processo que inicializa os nós e os *links* através da leitura de um ficheiro com uma estrutura que representa uma matriz de adjacências, e desencadeia eventos de $\{LinkUp\}$, ou seja, ativa ligações entre nós consoante o que estiver definido nessa mesma matriz de adjacências.

Cada nó está contido numa *Thread* diferente e fica a escuta de pacotes UDP num endereço IP único e numa porta que é igual em todos os nós. Quando um nó (a) precisa de enviar uma mensagem para outro nó (b) coloca essa mensagem na fila $mqueue_{a,b}$ do $Link_{ab}$, a qual será enviada quando o estado do *Link* estiver $[Up]$.

Os nós reagem apenas a três eventos, aos eventos de $\{LinkUp\}$ e $\{LinkDown\}$, onde pode ser despoletado o envio de mensagens para os vizinhos, e reagem ainda a recepção de mensagens provenientes dos seus vizinhos. Excluindo a inicialização da rede, os eventos de $\{LinkUp\}$ e $\{LinkDown\}$ são despoletados por um processo independente, designado de C&C.

Foi implementada uma funcionalidade de estatística, em que cada evento era registado num ficheiro com um *timestamp* e um identificador, para que posteriormente pode-se ser analisado e obtidas as estatísticas necessárias para avaliar o desempenho do algoritmo.

Achou-se ainda necessário desenvolver um programa para gerar diferentes topologias de rede e guardar esta topologia num ficheiro de texto como uma matriz de adjacências.

B. Abordagens alternativas

Dadas as limitações de hardware ao correr a simulação na nossa máquina local, surgiram as seguintes ideias para contornar essas mesmas limitações.

1) Data center: Um salto para escalar os testes seria a execução numa rede controlada de maior dimensão. Nessa rede seria possível ter um maior número de nós a funcionar, se utilizássemos um *Data Center*. No *Data Center* deveria ser criada uma rede com uma máscara mais pequena, por exemplo /16, e por cada nó colocado em cada máquina seria necessária uma interface virtual.

2) Resolução de IDs: Para conseguirmos correr numa rede não controlada, precisaríamos de resolver / atribuir a cada ID de um nó o seu respectivo IP. Para tal, seria necessário um ponto central atualizado que também introduziria um *overhead* considerável e seria um *bottleneck*.

IV. Resultados

Depois de implementado o emulador e o algoritmo de eleição de líder estar funcional, i.e., todos os nós acordarem quem é o líder, procedeu-se a várias simulações para recolha de resultados.

Estas simulações dividiram-se essencialmente em 4 tipos. O primeiro tipo consistiu em gerar varias topologias de rede variando o número de nós mas mantendo a densidade de ligações constante (15%) e executar 10 simulações por cada topologia. O segundo tipo consistiu no inverso da primeira, ou seja, gerar varias topologias de rede variando a densidade de ligações, mas mantendo o número de nos (100 nós) e novamente executar 10 simulações por cada.

Densidade neste contexto representa o número de ligações diretas que cada nó tem em média e é representado por uma percentagem relativamente ao número total de nós. Por exemplo, numa rede com 100 nós e uma densidade de 15% significa que cada nó em média tem 15 conexões diretas com outros nós. Neste caso o número total de ligações existentes seria 1500 e sabendo que numa rede de 100 nós é possível ter no máximo 10000 ligações únicas (incluindo ligações dos nós para eles mesmos), a densidade de ligações ativas seria o quociente entre as ligações existentes e número máximo de ligações únicas.

$$\frac{1500}{10000} = 0.15$$

O terceiro tipo de simulações consistiu em gerar duas topologias em árvore, em que uma estava balanceada e outra não balanceada, i.e., relativamente ao esperado líder existem muitas mais nós num dos ramos relativamente aos outros, o que pode ser observado na Figura 2. No quarto tipo de simulações também se gerou topologias em árvore, uma com profundidade de 3 e outra de 7, como se pode observar na Figura 3.

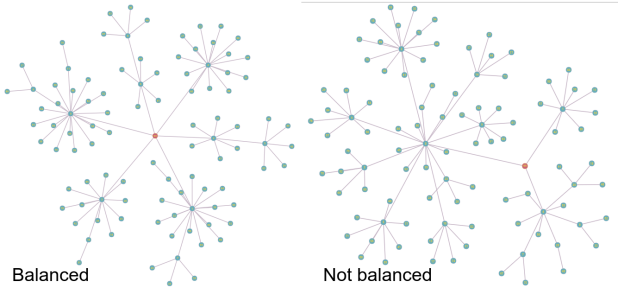

Fig. 2. Rede balanceada e rede não balanceada

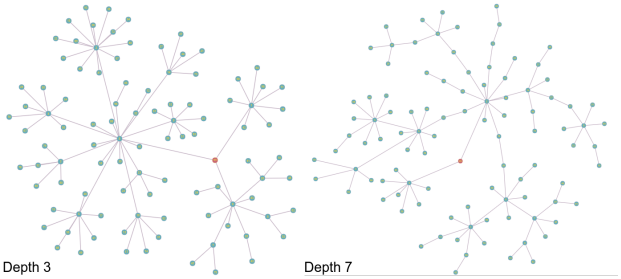

Fig. 3. Redes com profundidade 3 e 7

A partir dos dados das simulações do primeiro tipo obteve-se os gráficos das Figuras 4 e 5 onde se consegue perceber que existe uma relação aproximadamente linear positiva entre o tempo de eleição e o número de nós da rede. Já relativamente a relação entre o número de mensagens trocadas por nó e o número de nós da rede não parece linear e apresenta aumentar menos quanto maior o número de nós da rede.

Os gráficos das Figuras 6 e 7 foram obtidos a partir das simulações do segundo tipo e consegue-se observar que o tempo de eleição pouco varia em relação a densidade de ligações na rede mas relativamente ao número de mensagens trocadas nota-se que existe um aumento mais significativo e aproximadamente linear.

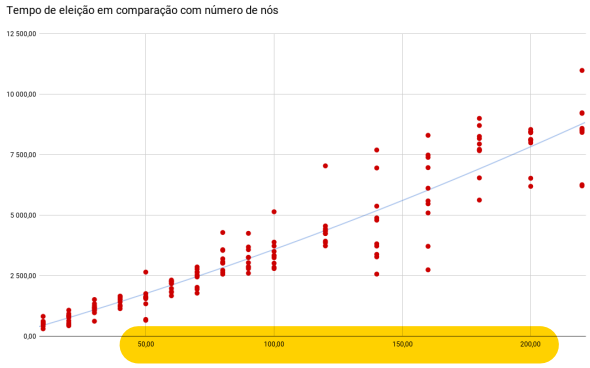

Fig. 4. Gráfico do tempo de eleição em comparação com número de nós

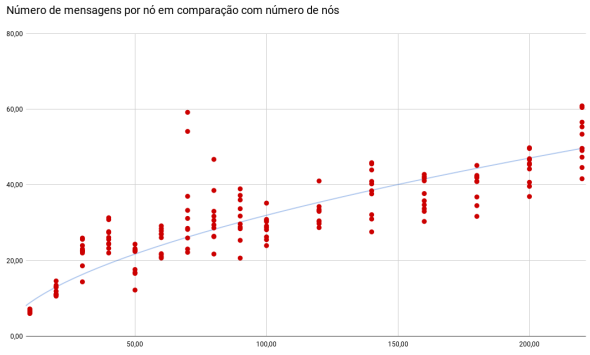

Fig. 5. Gráfico do número de mensagens por nó em comparação com número de nós

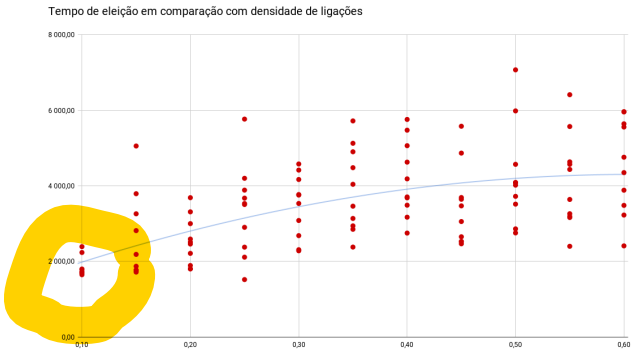

Fig. 6. Gráfico do tempo de eleição em comparação com densidade de ligações

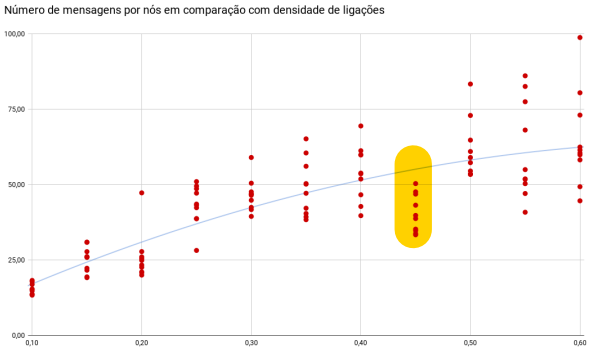

Fig. 7. Gráfico do número de mensagens por nó em comparação com densidade de ligações

As simulações do terceiro e quarto tipo foram repetidas 25 vezes para cada topologia de rede apresentando-se nas Tabelas I e II as médias para o tempo de eleição e número de mensagens trocadas por nó. Analisando as tabelas nota-se que o algoritmo tem um melhor desempenho em redes balanceadas e de pouca profundidade em termos de tempo de eleição. Já relativamente ao número de mensagens trocadas apresentou melhor desempenho em redes não balanceadas.

TABLE I
REDE BALANCEADA VS REDE NÃO BALANCEADA

	Rede balanceada	**Rede não balanceada**
Tempo de eleição (ms)	447,88	490,64
Mensagens trocadas / nó	13,578	10,398

TABLE II
REDE PROFUNDIDADE 3 VS REDE PROFUNDIDADE 7

	Rede profundidade 3	**Rede profundidade 7**
Tempo de eleição (ms)	490,64	561,20
Mensagens trocadas / nó	10,398	11,910

V. Análise crítica

Pode então ser retirado dos dados representados anteriormente que o algoritmo abordado tem um bom desempenho para redes dinâmicas, pois apesar de o tempo até a eleição do líder ter muito forte relação com o número de nós, não tem um crescimento abrupto o que o torna bastante escalável dentro dos limites testados. Conseguiu-se verificar que também tem um baixo *overhead* relativamente ao número de mensagens trocadas, mesmo com um grande aumento do número de nós da rede o aumento do número de mensagens trocadas é baixo, o que mais uma vez é um bom indicio da escalabilidade do algoritmo.

Nota-se que existe uma relação entre a profundidade da rede (desde o líder até ao nó mais distante) e o tempo de eleição, sendo que quanto maior a profundidade maior é o tempo eleição. Mas seriam necessários mais testes em relação a profundidade da rede para se poder fazer uma afirmação com mais certeza. Porem pode-se afirmar que não existe relação evidente entre a profundidade e o número de mensagens trocadas.

O algoritmo demonstra-se assim eficaz em redes dinâmicas e bastante escalável dentro das medidas analisadas.

References

[1] R. Ingram, P. Shields, J. E. Walter and J. L. Welch, ”An asynchronous leader election algorithm for dynamic networks,” 2009 IEEE International Symposium on Parallel & Distributed Processing, Rome, 2009, pp. 1-12.