An Asynchronous Leader Algorithm for Dynamic Network (janeiro 2018)

Bruno Daniel Vieira de Pinho
up201305783@fe.up.pt

Daniel Cardoso Fernandes
up201305789@fe.up.pt

***Abstract*— Uma possível implementação de um algoritmo de eleição de líder em redes assíncronas com alteração dinâmica de topologia é descrita neste documento. A metodologia adoptada foi implementar o algoritmo descrito no artigo "An Asynchronous Leader Election Algorithm for Dynamic Networks" que resulta da combinação de ideias do Temporally Ordered Routing Algorithm (TORA) for mobile ad hoc Networks e do Wave Algorithm. A característica deste algoritmo é garantir que independemente do tipo e número de alterações de topologia que ocorram, assim que estas terminem, todos os nós conectados na rede contém um único líder e serão um DAG (Directed Acyclic Graph) orientado ao líder. É, ainda, garantido que em situações normais não é eleito um novo líder desnecessariamente.**

Keywords*— *Líder, Eleição, Height, Link, Nó, Topologia

I. Introdução

A eleição de líder é uma importante primitiva no contexto de Sistemas Distribuídos, onde é comum existir sistemas que requerem a seleção de um único processo num conjunto de processos para liderar a execução de uma tarefa. Neste contexto, surgem os algoritmos de eleição de líder que são determinantes para levar a cabo essa eleição. Recorre-se aos algoritmos de eleição de líder em duas situações: quando se inicia um sistema distribuído e quando, por algum motivo, o líder anterior fica incapacitado de comunicar com os restantes processos (nós) da rede. Esta segunda situação é muito comum em redes dinâmicas onde os communication links vão a "UP" e a "DOWN" com muita frequência.

II. Definição do Problema

O desafio deste algoritmo de eleição é ser capaz de, assim que ocorrer a perda de comunicação com o líder atual, detetar essa perda, iniciar de imediato uma nova eleição e encontrar um novo líder que seja único entre todos os nós da rede na menor duração de tempo possível. Para que isto seja possível, a topologia deve apresentar um conjunto de características específicas que serão descritas de seguida.

A. *Mecanismo baseado em alturas*

A rede é constituída por um conjunto de nós e as suas ligações (links). Cada nó é identificado na rede pela sua altura que é constituída por uma 7-tuple:

τ - *timestamp* não negativo, que é 0 ou guarda o tempo em que a pesquisa foi iniciada;
oid – valor que é 0 ou o id do nó que início o pesquisa;
r – bit que é 0 quando é iniciada uma pesquisa e 1 quando a pesquisa termina;
δ – distância ao líder;
nlts – *timestamp* não positivo, cujo valor absoluto é o tempo de quando o líder se tornou líder;
lid – id do líder atual;
id – id do nó;

B. *Estrutura de um Nó*

Cada nó é definido com as seguintes componentes:

heights - lista com a própria altura e a altura dos seus vizinhos;
forming - lista que contém o número de vizinhos a serem formados;
N - lista com os vizinhos;
node_id - valor com o id do próprio nó;

C. *Estrutura de links*

A comunicação entre os nós é feita através de links que são definidos com as seguintes componentes:

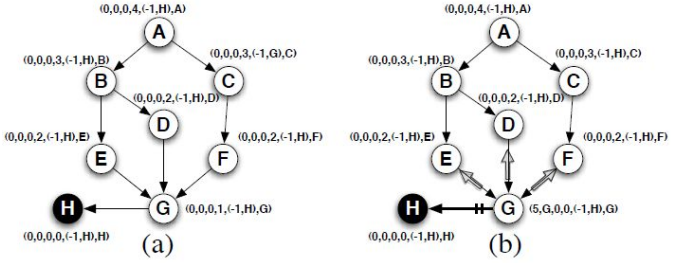
state – que pode ser "Up", "Down", "ComingUp" e "GoingDown";
mqueue_01 – lista com todas as mensagens enviadas do nó 0 para o nó 1;
mqueue_10 – lista com todas as mensagens enviadas do nó 1 para o nó 0;
[ ] id – vetor com duas posições que contém os id's dos nós do link;

III. Descrição da Solução

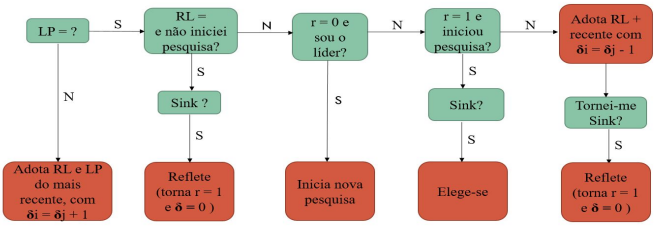
Na implementação deste algoritmo foi utilizado o protocolo UDP, cada nó tem associado a si um endereço IP e uma porta e utiliza um socket cliente/servidor. O cliente está sempre à escuta de novas mensagens e o servidor envia mensagens aos seus vizinhos apenas quando a sua altura for atualizada. Para comunicar com os seus vizinhos, cada nó acede a uma lista onde, através do id desse vizinho, consegue obter informações sobre o seu endereço IP e a sua porta, podendo de seguida enviar-lhe uma mensagem. As mensagens são sempre enviadas em unicast e o seu conteúdo é apenas a altura atual do nó que envia a mensagem.

Na implementação da topologia existe sempre um *link* entre cada dois nós da rede, mas, tal como já foi referido atrás, esse *link* tem um de 4 estados possíveis: *Up*, *Down*, *ComingUp* e *GoingDown*. Se o *link* tiver o estado *Up* significa que os nós são vizinhos, se o estado for *Down* os nós não são vizinhos, se for *ComingUp* os nós irão ser vizinhos e, consequentemente, o estado *GoingDown* significa que os nós vão deixar de ser vizinhos e as mensagens que estão nas filas serão descartadas. Na perspetiva do nó, o link é considerado incoming quando o seu **δ** é maior do que o **δ** do vizinho, e é considerado outgoing quando o seu **δ** é menor do que o do vizinho.

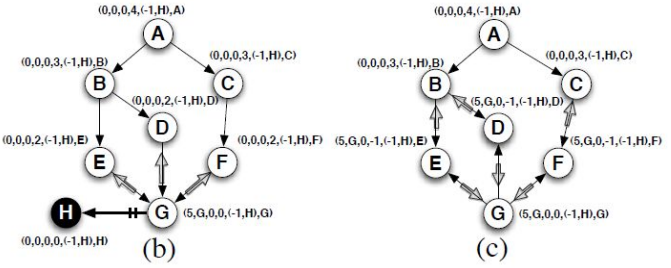
Na execução do algoritmo, assim que um nó vizinho do líder deteta que foi quebrado o link que o ligava ao líder, torna-se num *sink* e inicia uma nova eleição de líder. Por nó *sink,* entenda-se um nó que tem apenas ligações *incoming*, ou seja, não tem ligações *outgoing,* não é líder e os seus vizinhos têm o mesmo LP (*Leader Pair* - nlts e lid). Ao iniciar uma nova eleição, o nó irá alterar os 3 primeiros parâmetros da sua altura, colocando no **τ** o instante de tempo em que a eleição foi iniciada, no **oid** o seu próprio id, coloca o **r** a 0 (significando que uma eleição está a ocorrer) e altera o **δ** para 0 (distância do nó ao líder). Na figura seguinte é mostrada esta parte da execução do algoritmo.



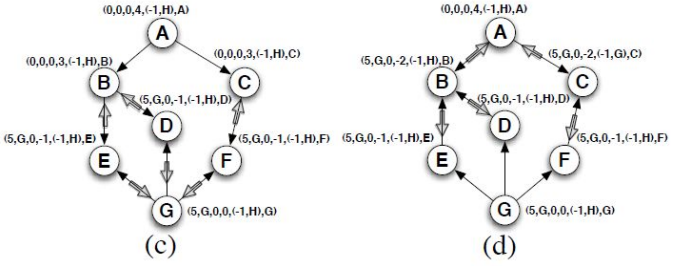
Após o nó G alterar a sua altura e iniciar a eleição envia a sua nova altura para os seus vizinhos, seguindo o exemplo anterior, os seus vizinhos serão o nó E, D e F. Esses nós ao receberem a atualização da altura G, realizam o *update* da sua altura. A função *update* é utilizada ao longo de todo o processo de eleição do novo líder e pode ser compreendida como uma máquina de estados que é apresentada na figura seguinte.



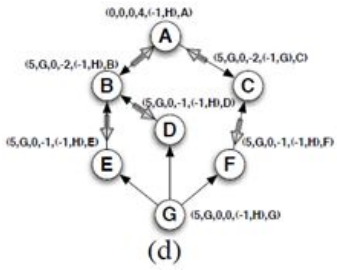
O nó E, D e F quando executam o *update* possuem o mesmo LP que G (nó que enviou a mensagem), não possuem o mesmo RL (entenda-se por RL, *reference level*, ou seja, as 3 primeiras componentes da altura **τ**, **oid** e **r**), nenhum deles é o líder, não foram eles a iniciar a pesquisa, não são *sink* e possuem **r = 0** o que indica que irão adotar o RL atual de G e colocam em **δ** o **δ** de G decrementado de uma unidade. A imagem seguinte, mostra esta execução.



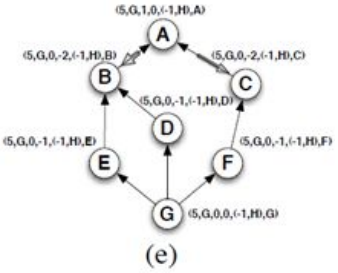
Após realizarem o *update* das suas alturas, os nós E, D e F enviam a sua nova altura para os seus vizinhos e os nós B e C irão receber essa atualização e executar o mesmo procedimento, como podemos verificar na imagem seguinte.



Relativamente ao nó A, este receberá as alturas de B e de C, adotando o seu RL e colocando o seu **δ** a -3. Ao fazê-lo, deixará de ter qualquer link que seja outgoing e irá inserir-se numa situação de sink, significando que, por este caminho, não foi encontrado o líder. Como tal, o nó A irá comunicar isto refletindo a pesquisa, que envolve colocar **r** = 1 e **δ** = 0. Situação exemplificada na imagem seguinte.

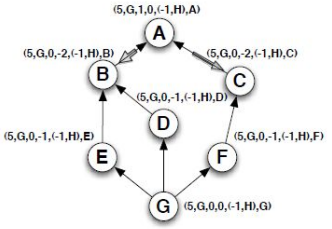


(d)

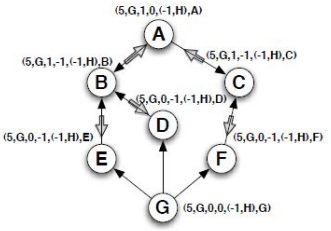


(e)

Após o nó A refletir, irá enviar a sua altura atualizada para os seus vizinhos. Ao receber a mensagem, o nó B e C irá ter o mesmo LP e RL que o nó A, terão o **r** = 1 e como não são *sink* nem foram eles que iniciaram a pesquisa irão adotar o RL do nó A e irão alterar o seu **δ** para o **δ** de A (nó que enviou a mensagem) decrementado de uma unidade. A imagem seguinte retrata essa etapa.

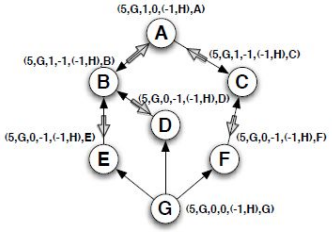


(e)

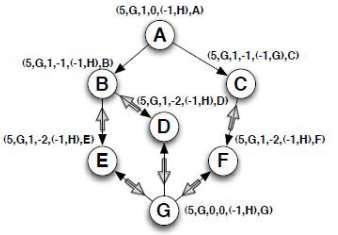


(f)

Depois dos nós B e C atualizarem a sua altura enviam o *update* para os seus vizinhos e o nó E, D e F ao receberem a mensagem irão executar um procedimento semelhante ao que estes realizaram, adotam o RL recebido e alteram o seu **δ** para o **δ** recebido decrementado em uma unidade. A imagem seguinte descreve esta etapa.

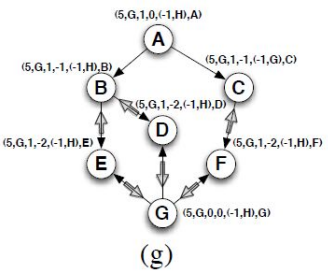


(f)

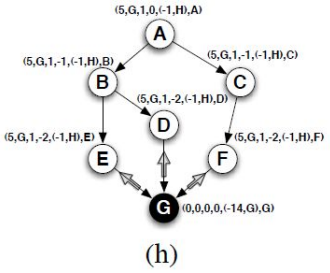


(g)

Seguidamente, o nó E, D e F irão enviar as suas alturas atualizadas para o nó G que ao receber as mensagens irá, numa primeira fase, adotar o RL recebido e alterar o seu **δ** para -3. Após receber as mensagens dos 3 nós vai-se tornar num *sink* e como foi este o nó que iniciou a pesquisa, elege-se o novo líder e envia essa nova informação para os seus vizinhos que por sua vez irão propagar a mensagem até toda a rede conhecer o novo líder. Esta etapa é exemplificada na imagem seguinte.



(g)



(h)

A. Topologia inicial

De maneira a verificar que o código implementado funcionava como o previsto, foi criada uma topologia inicial equivalente à descrita anteriormente, uma vez que permite uma comparação passo a passo entre o que é obtido e o que é esperado. Para iniciar o processo de eleição, é alterado o *link*, proveniente do líder, de *Up* para *GoingDown*. Quando cada um dos nós ligados a esse link deteta a sua mudança de estado, dá início a um evento de *LinkDown* que descarta todas as mensagens pendentes nas filas e faz cada nó apagar o outro da sua lista de vizinhos. No final do processo o estado do link mudará para *Down*, indicando que não há mais qualquer interação entre os dois nós. O nó G, após se aperceber que perdeu a sua ligação direta com o líder, inicia uma pesquisa para o tentar encontrar.

Quando a implementação se encontrava a funcionar para a topologia já mencionada, houve necessidade de confirmar o seu correto funcionamento para outro tipo de topologias iniciais. Para tal, desenvolveu-se um algoritmo capaz de criar uma rede aleatória dado o número de nós pretendidos. Nesse algoritmo é criada uma lista de listas de alturas (**hlist**), onde cada uma corresponde à lista de alturas de cada nó e cujo id está associado à posição que a lista ocupa. Cada altura é criada com um RL (0, 0, 0), um LP (-0, 1) e um id sequencial. A altura com **id** = 1 é a primeira a ser adicionada à lista de alturas, com **δ** = 0. Todas as altura seguintes escolhem aleatoriamente um nó, das alturas já pertencentes à lista, de quem serão vizinhos, assumindo o **δ** desse nó incrementado por uma unidade. De seguida ambos adicionam o outro à sua lista. Uma vez concluído, segue-se a formação de cada nó individual onde cada um irá aceder à lista e, consoante o seu id, criar as listas **heights** e **N** correspondentes. Por fim, é "cortada" uma das ligações do nó inicial, que dá início ao processo de eleição.

No entanto, este algoritmo demonstrava uma desvantagem: cada nó só teria, no máximo, 2 vizinhos.

Para contornar esta particularidade foi alterado o processo de ligação aos vizinhos. Quando um nó escolhe uma altura à qual se ligar, percorre todas as alturas com **δ** igual à do vizinho e, consoante uma probabilidade pré-definida, decide se a acrescenta à sua lista ou não.

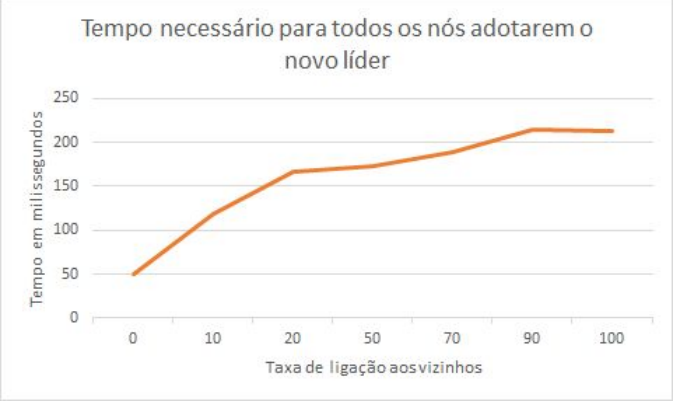
IV. Resultados e Análise Crítica

Uma vez encontrada forma de criar diferentes redes, tanto em forma como em número, tentou-se perceber no que é que a alteração destes parâmetros influencia o processo de eleição.

Para tal, foram definidos 3 indicadores: o tempo que demora a ser eleito um novo líder; o tempo que demora, desde a eleição do líder, até toda a rede ter conhecimento do novo líder; e o número de mensagens que são trocadas entre nós.

Para qualquer mudança de parâmetros foram recolhidos valores de várias execuções do programa, realizando-se no fim uma média desses valores.

Inicialmente, analisou-se o impacto da alteração da taxa de ligação aos vizinhos, e, para tal, fixou-se o número de nós a 50 enquanto se variou a taxa.



Ao analisar os resultados é de salientar o crescimento não só do tempo de eleição, como do tempo de adoção de líder, como do número de mensagens por nó à medida que a taxa aumenta.

Como o aumento da taxa implica um maior número de ligações com outros nós, é esperado o aumento de mensagens já que cada nó, para comunicar, envia a sua altura para todos os seus vizinhos.

Quanto menor a taxa, menor a probabilidade de haver um caminho alternativo para o líder, e, caso exista, maior é a probabilidade de ter de percorrer um maior caminho para chegar ao líder. Ao não haver caminho alternativo, a eleição ocorrerá numa rede menor, e portanto, com um baixo tempo de eleição. A haver caminho, o facto de ter um caminho mais rápido para o nó havia de apressar a eleição, no entanto, o aumento da taxa provoca o aumento de mensagens por nó, o que provoca um congestionamento nas ligações e consequentemente a mensagem demora mais a propagar-se.

Na análise seguinte, foi fixada a taxa de ligação aos vizinhos nos 20% e variou-se o número de nós existentes na topologia. A análise foi realizada tendo em consideração os 3 indicadores já mencionados.

Analisando os gráficos podemos concluir que à medida que aumentamos o número de nós se verifica um crescimento praticamente exponencial tanto no tempo de eleição do líder, no tempo até toda a rede adotar o novo líder e no número de mensagens. Este crescimento é explicado porque à medida que são adicionados nós a complexidade da topologia torna-se evidente, dificultando o caminho e necessitando de um maior número de mensagens para eleger um novo líder.

Durante a obtenção dos dados, ocorreram resultados que se distanciaram consideravelmente da média, quer por excesso quer por defeito. Isto deve-se à forma de como a rede é gerada, onde apesar de estar orientada (probabilisticamente) para tomar uma determinada forma, existe a possibilidade de serem formadas topologias em que ocorrem casos extremos, como por exemplo, todos os nós estarem diretamente ligados ao líder, ou todos os nós se ligarem a apenas um vizinho.

V. Conclusão

Através do estudo e implementação deste algoritmo de eleição de líder foi possível adquirir uma maior compreensão da dificuldade que advém da sua implementação em sistemas distribuídos reais que tenham por base redes dinâmicas. Ao longo do trabalho foram algumas as dificuldades encontradas desde logo porque o artigo é bastante teórico e a definição de alguns conceitos é pouco clara, o que criou algumas zonas cinzentas que tivemos de contornar de forma a que a implementação fizesse sentido.

Apesar dessas dificuldades, a implementação do algoritmo foi conseguida com sucesso e foi, ainda, possível obter a simulação de alguns casos particulares. Para além disso, existiu a preocupação de obter gráficos com resultados do tempo necessário para eleger um novo líder, do tempo necessário para que a rede tenha conhecimento que de existe um novo líder e do número de mensagens trocadas durante uma eleição.

Para concluir, dentro do tempo disponível para a execução deste projeto estamos bastante contentes com o trabalho que conseguimos realizar e acrescentamos ainda que ambos os elementos contribuíram de igual forma no desenvolvimento do trabalho.

Referências

[1] R. Ingram, P. Shields, J. Walter, and J. Welch. An Asynchronous Leader Election Algorithm for Dynamic Networks. Technical Report 2009-1-1, Department of Computer Science and Engineering, Texas A&M University, 2009.

[2] B. Awerbuch, A. Richa, and C. Scheideler. A jamming-resistant MAC protocol for single-hop wireless networks. In Proc. 27th ACM Symp. on Principles of Distributed Computing, pp. 45–54, 2008.

[3] J. Brunekreef, J.-P. Katoen, R. Koymans, and S. Mauw. Design and analysis of dynamic leader election protocols in broadcast networks. Distributed Computing, 9(4):157–171, 1996.