PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 22/09

Algoritmos de Eleição para Redes Móveis *Ad Hoc*

Pedro Nuno de Souza Moura
Markus Endler

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 22/09 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena August, 2009

Algoritmos de Eleição para Redes Móveis *Ad Hoc*

Pedro Nuno de Souza Moura and Markus Endler

{pmoura, endler}@inf.puc-rio.br

Abstract. This paper presents the leader election problem in mobile ad hoc networks and surveys several algorithms in the literature. Two of which were chosen, so that an in-depth study and a comparison, regarding not only the applicability, but also the computational complexity, were done.

Keywords: MANET, leader election, distributed algorithms.

Resumo. Este trabalho apresenta o problema da eleição de coordenador em redes móveis *ad hoc*, realizando um levantamento bibliográfico dos algoritmos presentes na literatura. Dois destes foram escolhidos para um estudo aprofundado e uma comparação quanto à aplicabilidade e à complexidade computacional.

Palavras-chave: MANET, eleição de coordenador, algoritmos distribuídos.

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

1 INTRODUÇÃO

Entre as tendências apontadas pelo avanço tecnológico nas últimas décadas, a mobilidade tem ganhado destacada notoriedade. Aparelhos como celulares, *notebooks* e *palmtops* assumem cada vez mais um papel importante no dia-a-dia das pessoas, na medida em que propiciam a realização de tarefas de maneira ágil e móvel, não exigindo a presença em um local específico de trabalho.

Mostra-se, pois, de fácil percepção a relevância do estudo de algoritmos e protocolos para esses dispositivos, que levem em consideração as características intrínsecas, bem como fatores limitantes destes. Ao se verificar a diferença existente entre as redes sem fios, ou redes móveis, e as redes tradicionais, baseadas em enlaces físicos, a importância de tal estudo se amplia e intensifica.

As redes móveis *ad hoc*, comumente denominadas apenas de *MANETs*, um acrônimo para *Mobile Ad Hoc Networks*, são redes cujos nós componentes são potencialmente móveis e que se comunicam através da troca de mensagens, seja diretamente, por meio de um enlace sem fio, seja indiretamente, por meio de uma sequência de nós intermediários. Não há a presença de entidades específicas para rotear as mensagens enviadas, cabendo a cada um dos participantes da rede esse papel.

Conforme os membros da rede se movem, partições da rede podem acontecer, assim como fusões, dependendo se um nó sai ou entra ao alcance de transmissão dos outros. Ademais, não se tem qualquer informação sobre a topologia assumida pela rede em qualquer instante.

O estudo e o desenvolvimento de algoritmos para tais redes constituem, portanto, um desafio, não só pelas mudanças constantes de topologia que podem ocorrer, como também pelas características singulares dos nós, que, tipicamente, possuem capacidade limitada de processamento e restrições de energia.

Assim sendo, este trabalho tem como objetivo apresentar os desafios impostos pela mobilidade na escolha de coordenadores em uma rede, assim como expor alguns algoritmos recentemente propostos.

2 O PROBLEMA DA ELEIÇÃO DE COORDENADOR

A presença de um nó coordenador, que execute um papel especial, em uma rede, seja esta com ou sem fio, se faz necessária em diversas situações e aplicações, como será abordado mais adiante. Entretanto, a definição de um nó especial que tenha o papel de coordenador durante toda a execução do sistema distribuído possui baixa tolerância à falha, já que, se o nó falhar, o sistema permanecerá inoperante (CHOW, 1997). Dessa forma, o sistema deve ser capaz de encontrar outro coordenador, que dê continuidade à execução do sistema distribuído.

A definição clássica presente na literatura para definição de coordenador é de eventualmente se escolher um, e somente um, nó de um conjunto pré-definido de nós participando do sistema distribuído para exercer o papel de coordenador (MASUM et al., 2006). Entretanto, dadas as características de um MANET, deve-se proceder a uma redefinição do problema, haja vista que, como partições podem ocorrer, deve-se contemplar a possibilidade de existirem componentes conexas, isto é, subgrupos de nós que não possuem comunicação. Além disso, o nó a ser escolhido deve ser aquele com o maior valor de prioridade sendo considerado, em que devem ser levados em consideração um dos seguintes critérios: tempo de bateria disponível, capacidade de processamento e distancia média em relação aos outros nós (MASUM et al., 2006).

Redefine-se o problema da seguinte forma: *Dada uma rede de nós móveis, cada qual com uma determinada prioridade, após um número finito de mudanças topológicas, cada componente conexa deverá eventualmente ter selecionado um único líder, possuindo a maior prioridade dentre os nós que fazem parte de sua componente* (VASUDEVAN, 2003).

Outros autores preferem citar explicitamente o fato de que uma componente conexa de nós da rede deva permanecer estática por um período suficientemente longo, para que seja possível convergir à escolha do seu coordenador. Contudo, ambas as definições vão ao encontro do fato de que deve haver períodos de estabilidade em uma componente, para que eventualmente se chegue à definição de seu líder.

Algumas aplicações do problema em questão são:

1) Distribuição de chaves em um esquema de criptografia de chave simétrica, onde se define o nó a ser escolhido como coordenador para desempenhar o papel de *Centro de Distribuição de Chaves* (*KDC – Key Distribution Center*) (MASUM et al., 2006);
2) Comunicação de Grupo para a abordagem de grupos estruturados através da presença de um coordenador, responsável por difundir as mensagens para os demais nós;
3) Coordenação da construção da tabela de roteamento em MANETs (PERKINS; ROYER, 1999);
4) Coordenação em redes de microssensores (HEINZELMAN et al., 2000); e
5) (CHEN et al., 2002) apresenta um algoritmo que se utiliza de um coordenador responsável pela administração da energia gasta por nós ociosos na rede, permitindo o desligamento da interface de rádio destes, mas sem alterar a topologia da rede, isto é, sem causar partições.

O problema de definição de coordenador já foi amplamente estudado e diversos algoritmos foram propostos para redes estáticas. No que tange a redes em anéis, (CHANG; ROBERTS, 1979) e (HIRSCHBERG; SINCLAIR, 1980) são dois algoritmos comumente utilizados e citados na literatura. Para topologias de grafos completos, (GARCIA-MOLINA, 82) propõe dois algoritmos, um para sistemas síncronos, em que falhas de nós são detectadas, e outro para sistemas assíncronos com possibilidade de ocorrerem partições na rede.

Como MANETs constituem um campo de estudo recente, poucos trabalhos foram propostos sobre o problema de eleição de coordenador. Em (MALPANI, 2000), são propostos dois algoritmos baseados no protocolo de roteamento *TORA*, o primeiro tratando de uma única mudança topológica na rede, enquanto o segundo aceita diversas mudanças simultâneas, ou seja, que ocorra uma mudança topológica antes que a rede tenha terminado de se recuperar da mudança anterior. Entretanto, em tais algoritmos qualquer nó pode ser o coordenador, não sendo necessariamente aquele com a maior prioridade.

(VASUDEVAN, 2004) apresenta um algoritmo baseado no conceito de computação difusa (*diffusing computations*) de Dijkstra e Scholten que obtém o nó cuja prioridade é máxima global na rede como coordenador, aceitando qualquer tipo de mudança topológica. Entretanto, (MASUM, 2006) afirma que a obtenção do máximo global é sempre custosa, afetando o desempenho do algoritmo.

(RAHMAN, 2008) propõe uma melhoria no algoritmo descrito acima, através do armazenamento, por cada nó, de uma lista de candidatos, de forma a diminuir o número de eleições realizadas. Os resultados das simulações apresentados demonstram que o algoritmo é, de fato, mais eficiente, tomando menos tempo para convergência e diminuindo o número de mensagens trocadas.

Foram escolhidos os trabalhos de (DAGDEVIREN; ERCIYES, 2008) e (MASUM et al., 2006) para um estudo aprofundado. O primeiro se destacou pelo fato de prover uma abordagem hierárquica, através da definição de um protocolo, para resolução do problema de escolha de um coordenador. Já o segundo, pelo fato de ser relativamente simples e ser baseado no paradigma do consenso. A seguir apresentam-se em detalhes os trabalhos supracitados.

3 ALGORITMOS ESTUDADOS

3.1 Hierarchical Leader Election Protocol

O primeiro algoritmo estudado foi proposto por (DAGDEVIREN; ERCIYES, 2008). Esse consiste em uma abordagem hierárquica, através da definição de *clusters*, para resolução do problema. O modelo de sistema assumido corresponde a:

1) Sistema é assíncrono;
2) Sem restrições sobre topologia da rede;
3) Os processos possuem prioridades distintas;
4) Comunicação é confiável e segura;
5) Partições e fusões podem ocorrer na rede.

Mais especificamente, o algoritmo está contido em um protocolo composto de quatro camadas, conforme exibido na Figura 1:

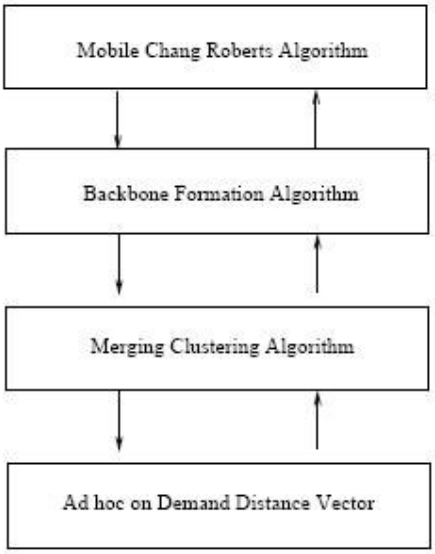

Figura 1 – As quatro camadas do protocolo. (DAGDEVIREN; ERCIYES, 2008).

A camada mais baixa corresponde ao algoritmo *AODV* de roteamento descrito em (PERKINS; ROYER, 1999), que é uma modificação do protocolo de roteamento *Distance-Vector*. Seu uso foi justificado não apenas pelo fato de ser comumente utilizado, como também por existir uma versão estável no ambiente de simulação utilizado pelos autores.

A segunda camada, descrita em mais detalhes em (DAGDEVIREN, et al., 2006), separa os nós da rede em *clusters* de forma balanceada, isto é, possuindo relativamente o mesmo número de membros. O algoritmo utilizado, denominado de *Merging Clustering Algorithm*, procede realizando fusões (*merges*) entre nós e aumentando o tamanho dos *clusters* enquanto for possível. Caso haja partições na rede, os *clusters* refletirão as componentes conexas, ou seja, os subgrupos de nós mutuamente comunicáveis da rede, sendo os representantes de cada *cluster* (*cluster head*) sempre os nós de maior prioridade.

É salutar citar que o algoritmo utilizado para *clusterização* possui uma complexidade de tempo $\Omega(\log n)$ e $\mathrm{O}(n)$, como demonstrado em (DAGDEVIREN, et al., 2006). Da mesma forma, a complexidade do número de mensagens é $O(n)$. É, pois, um algoritmo linear, mesmo no pior caso.

A terceira camada aplica um algoritmo de construção de *backbone* para comunicação entres os *clusters* (caso seja possível), de forma a obter uma topologia virtual de anel entre esses nós. Tal fato é alcançado através da aplicação de um algoritmo de árvore geradora mínima sobre os nós *cluster heads*. Note-se que, caso a rede seja desconexa, não será possível obter um *backbone* conectando todos os *clusters*.

Por fim, a quarta camada corresponde ao cerne do protocolo, em que é, de fato, obtido o coordenador da rede. Para tal, é aplicado o algoritmo de *Chang-Roberts* (CHANG; ROBERTS, 1979) sobre o anel virtual obtido no estabelecimento do *backbone*.

O algoritmo de *Chang-Roberts* procede da seguinte forma: um nó que queira se eleger envia uma mensagem com seu identificador e sua prioridade ao nó imediatamente à esquerda. Caso um dado nó receba o próprio identificador, então a sua mensagem deu a volta na rede, sendo esse o novo coordenador. Por outro lado, caso um nó de prioridade maior receba uma mensagem de menor prioridade, então retira essa mensagem da rede e envia para o seu vizinho uma mensagem com seu identificador e sua prioridade. Ademais, a complexidade do algoritmo em número de mensagens é $O(n^2)$, enquanto a de tempo corresponde a $O(n)$.

Como o algoritmo garante que o nó mais prioritário será eleito, tem-se que o coordenador da rede corresponderá ao nó *cluster head* com maior prioridade. A Figura 2 abaixo fornece uma visão esquemática do funcionamento do protocolo:

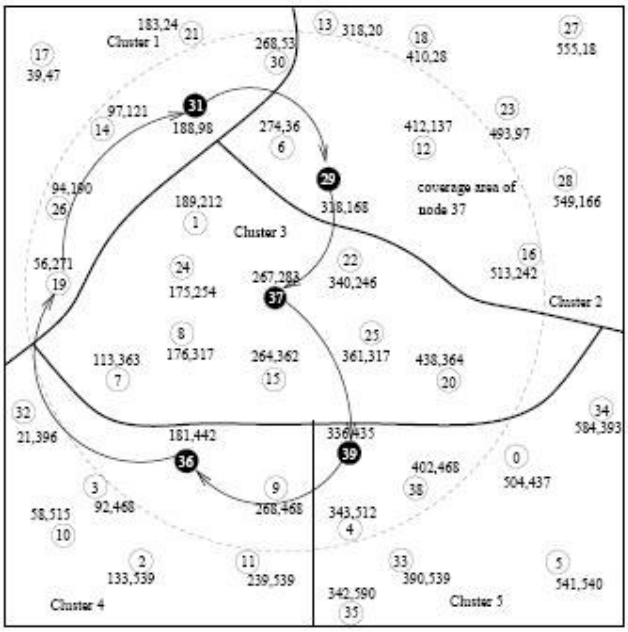

Figura 2 – Exemplo de execução do protocolo (DAGDEVIREN; ERCIYES, 2008).

Na figura acima, a área de cobertura do nó 37 corresponde à circunferência tracejada e os cinco *clusters* correspondem às regiões destacadas. Os nós *cluster heads* que compõem o anel virtual são: 31, 29, 37, 39 e 36. Após a execução do algoritmo de *Chang-Roberts*, o nó 39 é declarado coordenador.

A complexidade do número mensagens do protocolo corresponde à soma da complexidade de cada camada, sendo expressa por: $O(n) + O(kn) + O(k^2) = O(kn)$, em que k é o número de *clusters*. Por sua vez, a complexidade de tempo é expressa por: $O(n) + O(kn) + O(n) = O(kn)$. Logo, a complexidade de tempo e de mensagens do algoritmo é $O(kn)$, sendo praticamente linear no número de nós na rede.

3.2 *Asynchronous Leader Election Algorithm*

Este algoritmo, proposto por (MASUM et al., 2006), utiliza consenso para obter a máxima prioridade local (*local extrema*) de cada componente conexa de nós da rede. Quando um nó detecta a falha do líder, este tenta se eleger, enviando mensagem para um de seus vizinhos. Essa mensagem é repassada de nó em nó até que atinja a maioria dos nós na componente conexa ou atinja um determinado tempo limite. Neste caso, o nó é eleito novo coordenador e uma mensagem comunicando tal fato é disseminada entre os nós do subgrupo.

O modelo de sistema exigido pelo algoritmo corresponde a:

1) Sistema é assíncrono;
2) Cada nó possui um identificador único que é fixo durante sua existência;
3) Partições e fusões de componentes conexas da rede podem ocorrer;
4) Falhas de enlaces, assim como falhas e recuperações de nós podem ocorrer;
5) Canais de comunicação são bidirecionais, confiáveis e *FIFO*, mas uma mensagem só é garantida ser entregue se remetente e destinatário permanecerem conectados durante o período de entrega da mensagem;
6) Cada nó tem conhecimento do seu conjunto de vizinhos e percebe quando um nó entra ou sai desse conjunto; e

7) Cada nó deve permanecer na rede por um período relativamente longo.

As seguintes estruturas de dados são mantidas por cada nó:

- *ProcessID*: identificador do nó;
- *PriorityID*: prioridade do nó;
- *LeaderID*: identificador do líder da componente conexa a que este nó pertence;
- *LeaderPriorityID*: armazena a prioridade do líder;
- *LastElectionTag*: armazena o identificador da última eleição vista;
- *LastLeaderTag*: armazena o identificador do nó que originou a mensagem *Leader*; e
- *ElectionQueue*: fila que armazena as mensagens recebidas.

As seguintes mensagens são utilizadas na computação:

- *Election*: criada por um nó que queira se candidatar à eleição;
- *Leader*: mensagem que declara e informa aos demais nós da rede o vencedor da eleição; e
- *Hello*: mensagem enviada quando há fusões na rede de um nó para os seus novos vizinhos.

É importante citar que, para as mensagens *Election* e *Leader*, é passado um vetor de booleano *visited[ ]*, que guarda a informação se um nó já foi visitado ou não.

Quando um nó detecta a falha do coordenador, executa o seguinte procedimento, dando início à sua candidatura a coordenador:

Procedimento Initialize_Election( )
Election.PriorityID ← Current.PriorityID;
Election.LeaderID ← Current.ProcessID;
Election.ElectionTag ← Current.LastElectionTag + 1;
Election.Visited[i] ← FALSE, onde $1 \leq i \leq n$;
Election.Visited[Current.ProcessID] ← TRUE;
Election.TimeStamp ← 0;

Envia *Election* para os vizinhos usando heurísticas;
Fim-Procedimento

Os dois critérios para o término de uma eleição são:

1) Quando a mensagem visita a maioria dos nós, isto é, $(n/2)+1$ nós. Portanto, é um algoritmo baseado no consenso para obter um máximo local da rede; e
2) Quando o campo *TimeStamp* da mensagem excede um limiar pré-definido *T*.

O procedimento para tratar o recebimento de uma mensagem *Election* segue descrito abaixo. Note-se que a verificação dos critérios de término da eleição descritos acima é feita através de chamadas às funções *Valida* e *Maioria*.

Procedimento *Handle_Election*( )
se (Current.LastElectionTag ≤ Election.ElectionTag)
Election.TimeStamp++;

```
        Current.LastElectionTag ← Election.ElectionTag;

        se (Valida(Election.TimeStamp) and (not(Maioria(Election)))
            se (not(Election.Visited[Current.ProcessID]))
                Election.Visited[Current.ProcessID] ←TRUE;

                //Nó verifica se a sua prioridade é maior.
                se (Election.PriorityID < Current.PriorityID)
                    Election.LeaderID ← Current.ProcessID;
                    Election.PriorityID ← Current.PriorityID;
                fim-se

                Envia Election para os vizinhos usando heurísticas;
            fim-se
        senão
            // Caso em que a eleição se encerrou.
            Leader.ProcessID ← Current.ProcessID;
            Leader.ElectionID ← Current.LastElectionTag;
            Leader.LeaderID ← Election.LeaderID;
            Leader.PriorityID ← Election.PriorityID;
            Leader.Visited[i] ← FALSE, onde 1 ≤ i ≤ n ;
            Leader.Visited[Current.ProcessID] ← TRUE;

            Envia Leader para todos os vizinhos imediatos;
        fim-se
    senão
        //Caso em que é uma mensagem de eleição antiga. Apenas a descarta.
        Não faça nada;
    fim-se
Fim-Procedimento
```

No procedimento acima, caso um nó perceba que os critérios de parada foram satisfeitos, então inicia a disseminação de uma mensagem *Leader*.

A heurística utilizada para enviar a mensagem para os vizinhos em ambos os procedimentos descritos é:

1) Se há apenas um vizinho diretamente alcançável e ainda não visitado, então esse é o escolhido;
2) Se há mais de um vizinho, então um será escolhido e *Election.Visited[CurrentProcessID]* será definido como falso, para que se retorne ao nó posteriormente, a fim de visitar os outros vizinhos remanescentes; e
3) Se não há vizinho diretamente alcançável e que ainda não tenha sido visitado, então qualquer um é escolhido.

O seguinte procedimento trata o recebimento de mensagens *Leader*:

```
Procedimento Handle_Leader( )
    se (Current.LastElectionTag < Election.ElectionTag)
        Leader.Visited[Current.ProcessID] ← TRUE;
        Current.LastElectionTag ← Leader.ElectionID;
        Current.LastLeaderTag ← Leader.ProcessID;
```

```
            Current.LeaderID ← Leader.LeaderID;
            Current.PeaderPriorityID ← Leader.PriorityID;

            Envia Leader para todos os vizinhos v não-visitados;
        senão se ((Current.LastElectionTag = Leader.ElectionID) and
(Current.LastLeaderTag ≠ Leader.ProcessID))
            Leader.Visited[Current.ProcessID] ← TRUE;
            Current.LastLeaderTag ← Leader.ProcessID;

            se (Current.LeaderPriorityID < Leader.PriorityID)
                Current.LeaderID ← Leader.LeaderID;
                Current.LeaderPriorityID ← Leader.PriorityID;
                Envia Leader para todos os vizinhos v não-visitados;
            fim-se
        fim-se
Fim-Procedimento
```

O algoritmo permite que diversas eleições ocorram concorrentemente no sistema, aumentando a possibilidade de se achar um máximo global na rede. Mensagens de eleição antigas são descartadas pelos nós, bem como um nó sempre escolhe como líder o de maior prioridade, ainda que as eleições possuam mesmo identificador (mesmo valor do campo *ElectionID*).

Caso ocorra uma fusão na rede, um nó envia uma mensagem *Hello* para seus novos vizinhos. O seguinte procedimento trata mensagens *Hello*:

```
Procedimento Handle_Merge( )
        se (Current.LeaderPriorityID < Hello.LeaderPriorityID)
            Current.LeaderID ← Hello.LeaderID;
            Current.LeaderPriorityID ← Hello.LeaderPriorityID;
        fim-se

        Leader.ProcessID ← Current.ProcessID;
        Leader.ElectionID ← Current.LastElectionTag;
        Leader.LeaderID ← Current.LeaderID;
        Leader.PriorityID ← Current.LeaderPriorityID;
        Leader.Visited[i] ← FALSE, onde 1 ≤ i ≤ n;
        Leader.Visited[Current.ProcessID] ← TRUE;

        Envia Leader para todos os vizinhos v não-visitados;
Fim-Procedimento
```

Essencialmente, o procedimento escolhe o mais prioritário entre os líderes das componentes que se fundiram e, então, difunde essa informação por essa nova componente resultante. Ademais, cabe citar que um nó ao se recuperar de uma falha segue o procedimento acima.

A fim de provar a unicidade do coordenador por partição, efetua-se uma análise considerando dois cenários:

1) Há uma única eleição ocorrendo com o último *ElectionTag* (todas as outras serão eventualmente descartadas pelos procedimentos *HandleElection( )* e *Handle_Leader( )*): neste caso, uma vez eleito o coordenador, pelo procedimento *Handle_Leader( )*, todos os nós da partição receberão a

mensagem *Leader* com o *ID* e a prioridade do novo coordenador e, assim, terão uma única visão do nó coordenador. Caso ocorra uma partição durante ou após a eleição, eventualmente algum nó irá perceber e iniciar uma nova eleição através do procedimento *Initialize_Election( )*; e

2) Há mais de uma eleição ocorrendo com o último *ElectionTag*: neste caso, caso mais de uma eleição consiga chegar ao fim, o procedimento *Handle_Leader( )* selecionará como coordenador o nó de maior prioridade que tenha terminado sua eleição. Como todos os nós da partição são visitados pela mensagem *Leader*, segue que todos terão a mesma visão do coordenador. Por fim, caso ocorra uma partição durante ou após a eleição, eventualmente algum nó irá perceber e iniciar uma nova eleição através do procedimento *Initialize_Election( )*.

Por fim, a complexidade do algoritmo, tanto para o tempo quanto para o número de mensagens, será analisada no pior caso em que todos os nós (ou um percentual grande de n) tentam se eleger com o último *ElectionTag* relativamente ao mesmo tempo. Neste caso, cada uma das n mensagens será enviada para pelo menos $(n/2)+1$ nós, resultando em uma complexidade $O(n^2)$ para o número de mensagens. Cabe destaque ao fato de que o autor justifica essa complexidade por um ganho em tolerância a falhas, assim como maior possibilidade de obter um nó com prioridade máxima global.

A complexidade de tempo corresponde a $O(n)$, já que, para um nó que completar sua eleição, será necessário passar apenas um número constante de vezes por cada outro nó da rede, somando as etapas de eleição e de disseminação do novo coordenador (mensagens *Election* e *Leader*).

4 COMPARAÇÃO

Os algoritmos apresentados seguem vertentes diferentes na abordagem ao problema de definição de coordenador em uma MANET. O primeiro segue uma estratégia hierárquica, com a definição de *clusters* na rede. O segundo, uma estratégia baseada em consenso, em que a maioria dos nós precisa participar da eleição para que seja escolhido o novo coordenador. Ambos, por si só, constituem maneiras elegantes e interessantes de abordar o problema.

A primeira solução obtém um coordenador para cada *cluster* e, então, define um coordenador para toda a rede. Caso essa seja desconexa, não será possível nomear um coordenador geral, havendo um coordenador para cada componente conexa. O segundo algoritmo convergirá para obter um coordenador para cada componente assim que a maioria de nós desse grupo for atingida ou o *timestamp* exceder o limiar estabelecido.

Além disso, a primeira abordagem sempre determina, seja para uma componente conexa, seja para a rede toda, o coordenador com a maior prioridade, enquanto a segunda obtém um máximo local, não necessariamente global, dentre o conjunto de nós da componente ou da rede, pois nem todos os nós são consultados.

A complexidade do primeiro algoritmo corresponde a $O(nk)$, em que k é o número de *clusters* e pode ser considerado uma constante, tanto para tempo quanto para número de mensagens trocadas. Já a complexidade do segundo algoritmo corresponde a $O(n^2)$ para mensagens e $O(n)$ para tempo, conforme evidenciado no texto. Além disso, há uma sobrecarga no *payload* das mensagens transmitidas, já que é

passado um vetor de booleano de tamanho n em cada mensagem. Portanto, o primeiro algoritmo é mais eficiente sob o ponto de vista de complexidade de tempo e de mensagens.

A Figura 3 abaixo corrobora a complexidade de tempo linear obtida para o primeiro algoritmo. Nessa, o crescimento do tempo de execução com o número de nós é praticamente linear.

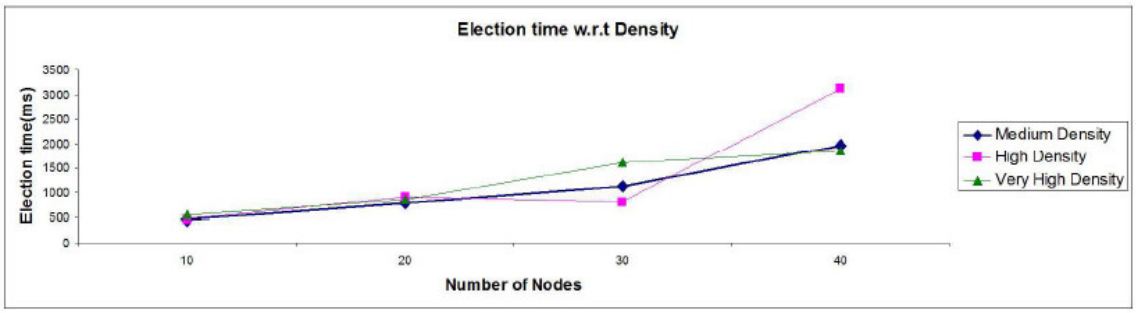

Figura 3 – Tempo de execução por densidade de nós (DAGDEVIREN; ERCIYES, 2008).

O tempo de execução pouco se altera com diferentes números de *clusters* utilizados, conforme mostrado na Figura 4 abaixo:

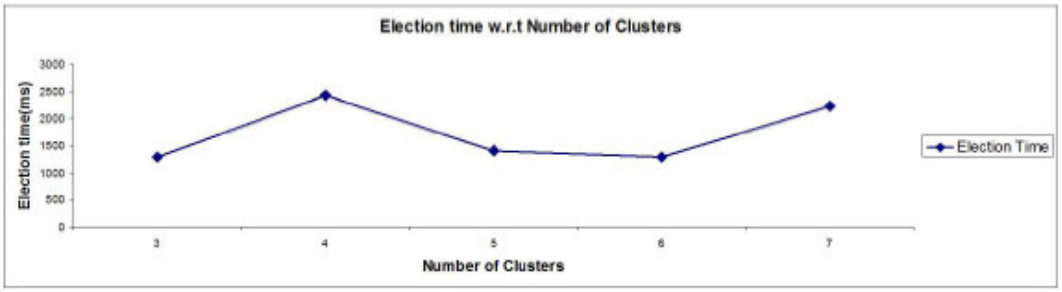

Figura 4 – Tempo de execução por número de *clusters* (DAGDEVIREN; ERCIYES, 2008).

Apesar de mais eficiente, a primeira solução consiste, de fato, em um protocolo, isto é, uma composição de diversos algoritmos em camadas, o que, de certa forma, pode restringir sua aplicação, já que seu uso implica no uso de suas camadas inferiores. Por outro lado, o segundo algoritmo, apesar de menos eficiente sob o ponto de vista de número de mensagens, não impõe nenhuma restrição sobre as suas camadas subjacentes.

5 CONCLUSÃO

Pela análise realizada, pôde-se perceber que o estudo de algoritmos para redes móveis *ad hoc* representa um grande desafio, pelas próprias características da rede. Não há nós especiais responsáveis pelo roteamento, mudanças constantes de topologia podem acontecer e os nós têm propriedades particulares, como bateria limitada e baixa capacidade de processamento.

A presença de um coordenador se faz importante em diversas aplicações e sistemas distribuídos. Assim sendo, justifica-se o estudo de tal problema no âmbito das redes móveis. Contudo, pouco ainda foi estudado e desenvolvido, restando muita coisa a ser explorada.

Foram apresentadas duas soluções para o problema supracitado: uma mais robusta, que fornece um protocolo hierárquico de definição de líder, e um algoritmo baseado em consenso que garante eventualmente obter um máximo local para cada componente conexa da rede. A primeira abordagem se mostrou mais eficiente, por possuir baixa complexidade de execução e de mensagens. Por outro lado, apesar de sua complexidade quadrática, dependendo-se da aplicação que se queira realizar, pode-se optar pela segunda abordagem, por não fazer nenhuma restrição quanto aos algoritmos a serem utilizados nas camadas mais baixas da rede.

REFERÊNCIAS

1) BOUKERCHE, Azzedine; ABROUGHI, Kaouther. *An Efficient Leader Election Protocol for Mobile Networks*. In: Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing, 2006, 1129 – 1134 p.

2) CHANG, E. J.; ROBERTS, R. *An Improved Algorithm for Decentralized Extrema Finding in Circular Arrangements of Processes*. In: Communications of the ACM, 1979, 281 – 283 p.

3) CHEN, Benjie et al. *Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks*. In: Wireless Networks, 2002, vol. 8, 481 – 494 p.

4) CHOW, R.; JOHNSON, T. *Distributed Operating System and Algorithms*. Nova Iorque: Addison Wesley, 1997.

5) DAGDEVIREN, Orhan; ERCIYES, Kayhan. *A Hierarchical Leader Election Protocol for Mobile Ad Hoc Networks*. In: Proceedings of the 8th International Conference on Computational Science, 2008, 509 – 518 p.

6) DAGDEVIREN, Orhan et al. *Merging Clustering Algorithms in Mobile Ad Hoc Networks*. In: International Conference on Computational Science and Its Applications, 2006, 681 – 690 p.

7) GARCIA-MOLINA, Hector. *Elections in a Distributed Computing System*. In: IEEE Transactions on Computers, 1982, C-31, 48 – 59 p.

8) HEINZELMAN, Wendi et al. *Energy-Efficient Communication Protocol for Wireless Microsensor Networks*. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

10) HIRSCHBERG, D.S.; SINCLAIR, J. B. *Decentralized Extrema-Finding in Circular Configurations of Processors*. In: Communications of the ACM, 1980, 627 – 628 p.

11) MALPANI, Navneet et al. *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, 2000, 96 – 103 p.

12) MASUM, Salahuddin et al. *Asynchronous Leader Election in Mobile Ad Hoc Networks*. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications, 2006, 827 – 831 p.

13) PERKINS, Charles; ROYER, Elizabeth. *Ad Hoc On-Demand Distance Vector Routing*. In: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999, 90 – 100 p.

14) RAHMAN, Muhammad et al. *Performance Analysis of Leader Election Algorithms in Mobile Ad Hoc Networks*. In: International Journal of Computer Science and Network Security, vol. 2, 2008.

15) VASUDEVAN, Sudarshan et al. *Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Network*. In: Proceedings of the 12th IEEE International Conference on Network Protocols, 2004, 350 – 360 p.

16) VASUDEVAN, Sudarshan et al. *Leader Election Algorithms for Wireless Ad Hoc Networks*. In: Proceedings of the DARPA Information Survivability Conference and Exposition, 2003, 261 – 272 p.