Algoritmos Distribuídos para Roteamento em Redes *Ad Hoc*

Tiago Rodrigues Chaves, Ricardo Augusto Rabelo Oliveira
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: tiagochaves@gmail.com, rrabelo@gmail.com

***Resumo*—Este trabalho introduz a aplicação de algoritmos distribuídos de roteamento em redes *ad hoc*. Os algoritmos *Flooding* e *Ad hoc On-demand Distance Vector* (AODV) serão apresentados em maiores detalhes introduzindo o seu funcionamento. As ações básicas relacionadas com o processo de roteamento serão estudadas e aplicadas em diferentes topologias de rede. Análise quantitativa será realizada em relação aos diferentes tipos de mensagens enviadas pelo algoritmo *Flooding*. Este trabalho tratará de algoritmos distribuídos de roteamento com atenção ao envio de mensagens entre os canais de comunicação. Os algoritmos *Flooding* e AODV serão implementados em linguagem Java e testados no *Distributed Algorithms in Java* (DAJ), simulador de algoritmos distribuídos. Também apresentaremos a análise de complexidade destes algoritmos.**

***Keywords*-Algoritmos Distribuídos, Roteamento, Redes *Ad Hoc*, Auto-Estabilização, Análise de Complexidade.**

I. Introdução

A demanda de tecnologia para comunicação sem fio está aumentando diariamente, dado a redução do custo dos equipamentos digitais aliado ao avanço da tecnologia, percebemos a popularização dos computadores portáteis e dos dispositivos móveis como *smartphones* e *tablets*. Com esse crescimento, não torna-se prático ter uma arquitetura fixa para comunicação entre os dispositivos, uma vez que estes estão em todos os lugares.

Uma rede *ad hoc*, também conhecida como MANET (*Mobile Ad hoc NETwork*), é formada por nós móveis de alcance limitado, capazes de trocar informações diretamente entre si. Ao contrário do que ocorre em redes convencionais, não existem pontos de acesso, ou seja, não há infraestrutura de conexão, os nós dependem uns dos outros para manter a rede conectada [1]. Por esse motivo, redes *ad hoc* são indicadas principalmente em situações onde não se pode, ou não faz sentido, instalar uma rede fixa. Os nós de uma rede *ad hoc* podem se mover arbitrariamente, tornando a topologia altamente mutável [2]. Assim, a conectividade entre os nós mudam constantemente, demandando permanente adaptação e reconfiguração de rotas, o que torna o roteamento em rede *ad hoc* um grande desafio.

A função de um algoritmo de roteamento é encontrar, estabelecer e manter rotas entre dois nós que desejam se comunicar. Os algoritmos de roteamento para rede sem fio *ad hoc* devem ser capaz de lidar com um número muito grande de nós com recursos limitados, tais como largura de banda e de energia. Outro desafio para os algoritmos de roteamento é que eles também precisam lidar com mobilidade dos nós, o que significa que nós podem aparecer e desaparecer em vários locais. Assim, todos os nós da rede *ad hoc* desempenham o papel de roteadores e devem participar na descoberta e manutenção das rotas para os outros nós. Mas o principal desafio para os algoritmos de roteamento em redes *ad hoc* consiste em reduzir a sobrecarga de mensagens enviadas para o roteamento, possibilitando o bom funcionamento da rede sem congestionamento dos canais de comunicação.

Existem vários algoritmos de roteamento *ad hoc*, com vantagens e desvantagens de acordo com situações específicas. A principal divisão dos algoritmos de roteamento é baseado sobre como e quando as rotas para o destino são descobertas. No contexto de redes *ad hoc*, os algoritmos de roteamento podem ser classificados como pró-ativos ou reativos.

Nos algoritmos pró-ativos, todo nó da rede possui na sua tabela de roteamento as informações para todos os possíveis destinos. A grande vantagem desses algoritmos é o fato dos pacotes serem enviados com um atraso mínimo já que os nós conhecem previamente as rotas. Porém, é preciso que as redes possuam banda suficiente para evitar congestionamento e seus recursos de energia não sejam escassos, já que a troca de mensagens de roteamento é elevada para garantir o conhecimento de rotas válidas.

Nos algoritmos reativos, uma rota só é determinada quando um nó deseja enviar um pacote para outro nó, ou seja, o algoritmo atua sob demanda. Desta forma, os recursos como banda e energia podem ser utilizados de uma forma mais eficiente, pois só são gastos quando há a necessidade de descoberta de rotas. A desvantagem desses algoritmos é o maior atraso no envio dos pacotes, pois se a rota do destino do pacote não for conhecida, o procedimento de descoberta de rota deve ser realizado.

As abordagens mais clássicas ao problema de roteamento são inundamento (*flooding*) e vetor distancia (*distance vec-*

tor).

Flooding: é a abordagem mais simples, toda mensagem que chega no nó é enviado para todos os outros nós, com que este nó tem um canal de comunicação, menos para o que lhe enviou a mensagem. A abordagem é simples mas eficaz, em algum momento do tempo, o nó destino vai receber a mensagem, talvez até mais de uma vez. Porém, temos o problema de escalabilidade da rede, com o aumento desta, a largura de banda necessária ao algoritmo o torna proibitivo. Outro problema é o de *loops* de roteamento, mensagens podem ficar "vagando" indefinidamente na rede. Este problema pode ser resolvido com a atribuição de um tempo de vida da mensagem, mas cria-se o problema de determinar o tempo ideal para que a mensagem chegue ao destino, sem desperdício dos recursos da rede [3].

Distance Vector: mantém uma tabela com o menor caminho até todos os outros nós. A tabela é atualizada periodicamente, com as informações vinda dos vizinhos. A cada tabela recebida, compara com a que tem, se alguma rota for menor, atualiza sua tabela e armazena de onde veio a informação. As vantagens são sua simplicidade e eficiência computacional, devido a sua característica distribuída. O problema é que apresenta baixa convergência quando a topologia muda muito, como é o caso em redes *ad hoc* [3].

Algumas características são desejáveis para que os algoritmos de roteamento em redes *ad hoc* sejam eficazes: operação distribuída, evitar a formação de *loops*, operação sob demanda, operação pró-ativa, técnicas de segurança, adaptação a períodos de inatividade dos nós e suporte a *link* unidirecionais. [4]

Dado a natureza mutável das redes *ad hoc*, os algoritmos de roteamento normalmente são algoritmos distribuídos e apresentam técnicas de tolerância a falhas, como auto-estabilização.

Um algoritmo distribuído consiste de um paradigma no qual o sistema computacional é formado por sistemas de memória distribuída, interligados por um canal de comunicação. Os sistemas de memória distribuída são uma coleção de processadores interconectados por algum canal que permita comunicação ponto a ponto. Os processadores não compartilham fisicamente qualquer memória, de forma que a troca de mensagens torna-se necessária.

Um algoritmo auto-estabilizável é construído de forma que um determinado processo execute as mesmas operações em estados falhos ou não, visando atingir um estado correto em um número finito de passos. Tal algoritmo também não requer uma inicialização correta, pode se recuperar de qualquer falha transiente a qualquer momento o que torna a auto-estabilização uma solução para um grande número de aplicações em grafos e comunicação de protocolos [5].

Em uma rede *ad hoc* não há topologia predeterminada, nem controle centralizado, dessa forma é nos algoritmos distribuídos que se encontram todo o segredo para a otimização dessa rede, tornando assim, a chave para o sucesso ou fracasso.

Sabe-se que a troca de mensagens entre os nós computacionais é considerada crítica. Mensagens são perdidas, duplicadas, corrompida ou entregues fora de ordem. Dessa forma estudar e propor melhorias para o problema da troca de mensagens em algoritmos distribuídos, pode apresentar grandes contribuições para área.

O objetivo deste trabalho consiste em apresentar, implementar, simular e comparar dois algoritmos distribuídos, apresentando a análise de complexidade e a análise experimental do número de troca de mensagens em diferentes topologias de rede. Analisaremos um algoritmo de *Flooding* simples e ingênuo, onde uma mensagem é enviada para todos os outros nós e um algoritmo reativo, conhecido como AODV *Ad hoc On-demand Distance Vector*, este escolhido devido a sua característica adaptativa a cenários de alta mobilidade, cenário comum em redes *ad hoc*.

Este trabalho está organizado da seguinte forma. A Seção II, apresenta a definição dos algoritmos de roteamento estudados, seguidos por metodologia na seção III, experimentos na seção IV, análise de complexidade na seção V, conclusões e trabalhos futuros respectivamente nas seções VI e VII.

II. Algoritmos de Roteamento

A. Algoritmo de inundação (Flooding)

O algoritmo de roteamento *Flooding*, consiste em um algoritmo onde uma mensagem é enviada por um ou mais nós da rede, sendo retransmitida e propaganda por todo os nós da rede. Esta propagação ocasiona o efeito de inundação dos nós e canais de comunicação. Estas mensagens podem gerar *loops* infinitos, sendo retransmitida diversas vezes para o mesmo nó de diferentes vizinhos.

O algoritmo de inundação é um algoritmo de roteamento *broadcast*. Nesse algoritmo, um pacote que chega a um nó x é enviado para todos os nós conectados a x, menos para aquele de quem x recebeu o pacote, caso haja uma falha os dados podem ser entregues através de outra rota [6].

Na Figura 1, dado que uma mensagem foi transmitida pelo nó central, esta mensagem se propagará por toda a rede. Assim, os nós extremos receberam a mesma mensagem diversas vezes.

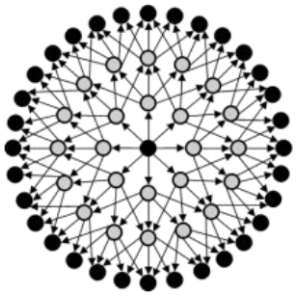

Figura 1. Flooding - Figura retirada e adaptada de [7].

B. Ad hoc On Demand Distance Vector (AODV)

AODV é um algoritmo de roteamento reativo, ou seja, as rotas são criadas e mantidas apenas quando necessário.

O algoritmo de roteamento AODV foi projetado para uso em redes móveis *ad hoc*. Como o algoritmo funciona sob demanda, só existe a necessidade de um nó A conhecer uma rota para um nó B quando a comunicação entre eles for necessária. Desta forma, evita-se o desperdício de banda e minimiza-se o uso de memória e processamento nos nós. As rotas para destinos com os quais os nós não estejam em comunicação ativa não são mantidas.

Este algoritmo utiliza as seguintes variáveis:

- Número de sequência da fonte (*SrcSeqNum*): número de sequência gerado pelo nó fonte;
- Número de sequência do destino (*DestSeqNum*): número de sequência crescente usado como forma de se verificar o quão recente é uma rota, pois o *DestSeqNum* com o maior valor é que representará a rota mais atual;
- Identificador fonte (*SrcID*): especifica o nó fonte que pretende enviar pacotes a um determinado nó da rede;
- Identificador do destino (*DestID*): especifica o nó destino que deverá receber pacotes de um determinado nó fonte;
- Identificador de *broadcast* (*BcastID*): quando vários *broadcasts* são enviados à rede, cada um recebe um identificador que é relacionado a uma determinada mensagem *RREQ*, podendo assim identificá-la. É incrementado a cada nova requisição de rota enviada;
- Tempo de vida das mensagens (*TTL*): contador que representa o tempo que se considera até que uma mensagem possa ser descartada;
- Número de *hops*: número de saltos necessários para se alcançar um determinado nó a partir de um nó origem.

O algoritmo possui as seguintes mensagens:

- Mensagens *HELLO*: enviadas periodicamente para se confirmar a conectividade local entre vizinhos;
- *RREQ*: mensagens de requisição de rota enviadas a toda a rede para encontrar o nó destino, quando a rota já não é conhecida;
- *Route Reply* (*RREP*): resposta à requisição de rota especificando como chegar até o nó destino;
- *Route Error* (*RERR*): aviso de queda de enlace. Quando um enlace deixa de existir entre dois nós é enviado uma *RERR*.

Quando um nó quer enviar um pacote a outro nó e ainda não possui uma rota determinada, o nó fonte envia um *RouteRequest* (contendo *SrcID*, *DestID*, *SrcSeqNum*, *DestSeqNum*, *BcastID* e *TTL*) a todos os seus vizinhos por difusão. Caso um vizinho não seja o destino ou não possua uma rota válida para o destino, ele repassa a *RREQ* aos seus vizinhos. Este processo se repete até a requisição atingir o destino ou um nó que conheça uma rota para este, inundando a rede.

Um nó de destino, ao responder a *RREQ*, envia pelo caminho reverso uma mensagem *RREP*, que contém o endereço da fonte e do destino, o número de sequência do destino, o contador de saltos (incrementado de uma unidade a cada salto) e seu *TTL*. Antes de enviar a *RREP*, o nó de destino atualiza o seu número de sequência para o máximo entre o valor atual e o valor que constava na *RREQ*. Em cada nó atravessado no caminho reverso pela *RREP* é armazenado, em uma entrada referente ao destino, o próximo salto para alcançar o destino, ou seja, o endereço do vizinho de quem se recebeu a resposta e o número de sequência do destino.

A movimentação de um nó ativo pode provocar a queda de um dado enlace que estava sendo utilizado. Nesta situação uma mensagem de erro (*RERR*) é enviada a todos os nós afetados, avisando sobre a queda do enlace. Dessa forma, cada nó por onde passa a *RERR* deve decidir se continua ou não a repassar esta mensagem, além de ter que atualizar a sua tabela de roteamento, registrando que os destinos especificados na mensagem estão inalcançáveis e atualizando seus números de sequência.

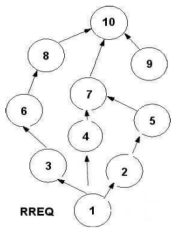

Figura 2. Mensagem RREQ no AODV.

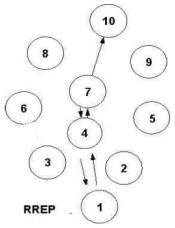

Figura 3. Mensagem RREP no AODV.

As Figuras 2 e 3 ilustram respectivamente as mensagens *RREQ* e *RREP* do AODV na rede. O *broadcast* de *RREQ* se inicia com o nó 1 enviando *RREQ* aos nós 2, 3 e 4, que por sua vez verificam se possuem a rota até o nó destino 10. Caso não possuam, eles enviam *RREQ* aos seus vizinhos diretamente conectados. Os nós 5, 6 e 7 são os vizinhos de 2, 3 e 4 respectivamente. Como o nó 7 possui uma rota até o nó destino 10, então ele envia uma *RREP* até o nó fonte. Caso outro nó possuísse também uma rota até o nó 10, este nó responderia com uma *RREP* ao nó fonte. Como, por exemplo, o nó 8.

III. METODOLOGIA

O trabalho desenvolvido foi dividido em quatro etapas:

Análise: Dentre os algoritmos tradicionais de roteamento encontrados na literatura, neste trabalho foram analisados os algoritmos de *Flooding* e AODV que apresentam diferentes abordagens. Enquanto algoritmo de *Flooding* consiste de uma abordagem simples e ingênua, onde uma mensagem é enviada para todos os outros nós, o AODV é um algoritmo clássico desenvolvido para ser utilizado por nós móveis em uma rede *ad hoc*. O protocolo propõe ser capaz de apresentar uma rápida adaptação as condições dinâmicas dos enlaces, baixo processamento e reduzida taxa de utilização da rede, determinando rotas para destinos dentro de uma rede *ad hoc* [8].

Implementação: Após análise destes algoritmos, os mesmos foram implementados em linguagem Java. A Figura 4, apresenta o código fonte da implementação do algoritmo *Flooding*.

```
1 public void main() {
2
3     out().send(new Rreq(-1, number));
4     System.out.println("RREQ: " + number + " => ?");
5     boolean loop = true;
6     while (loop) {
7         index = in().select();
8         if (index == -1) {
9             continue;
10        }
11        msg = in(index).receive();
12        if (msg instanceof Rreq) {
13            Rreq aux = (Rreq) msg;
14            out(index).send(new Rrep(aux.originador, number));
15            vizinhos[aux.originador] = true;
16            System.out.println("RREP: " + number + " => " +
17            aux.originador);
18            msg = null;
19        }
20        if (msg instanceof Rrep) {
21            Rrep aux = (Rrep) msg;
22            if (aux.destino == number) {
23                System.out.println("Eu sou: " + aux.destino +
24                " - Mensagem Recebida de: " + aux.originador);
25                vizinhos[aux.originador] = true;
26                count++;
27                if (count == nroNodos) {
28                    break;
29                }
30            } else {
31                out().send(aux);
32                System.out.println("RREP-ENC: " +
33                aux.destino + " => " + aux.originador);
34                naoVizinhos[aux.originador] = true;
35            }
36            msg = null;
37        }
38    }
39    for (int i = 0; i < nroNodos; i++) {
40        System.out.println("Posição:(Vizinho)"
41        + i + " - " + vizinhos[i]);
42    }
43    for (int i = 0; i < nroNodos; i++) {
44        System.out.println("Posição:(Nao Vizinho)"
45        + i + " - " + naoVizinhos[i]);
46    }
47}
```

Figura 4. Algoritmo de *Flooding*.

Simulação: Na etapa de simulação os algoritmos de roteamento implementados foram testados em um simulador de algoritmos distribuídos. O *Distributed Algorithms in Java* (DAJ). Para a simulação foram construídas diferentes topologias de rede, onde os algoritmos foram testados.

IV. EXPERIMENTOS

Topologia de Rede é a maneira como os nós de uma rede podem ser organizados. Na Figura 5 podemos ver cinco topologia diferentes.

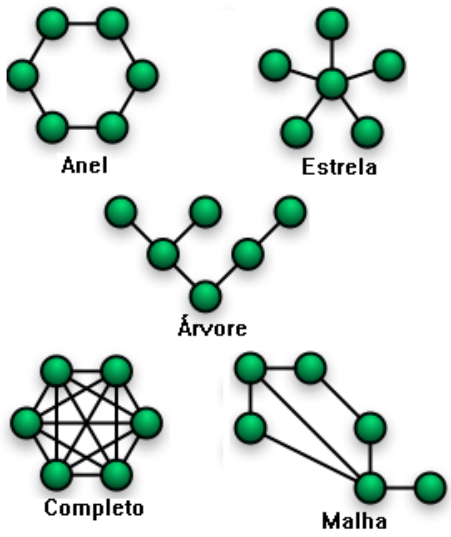

Figura 5. Topologias de Redes - Figura retirada e adaptada de [9]

Os experimentos foram realizados executando o algoritmo de *Flooding* e o AODV em cada uma das topologias de rede apresentadas na Figura 5. As redes foram definidas arbitrariamente contendo sete nós, esta definição apenas para facilitar a visualização das topologias de rede, porém a quantidade de nós pode ser expandida para um número n.

No *Flooding* implementado, cada nó envia para todos os seus nós vizinhos uma mensagem de *RREQ*, o nó que recebe esta mensagem, envia uma mensagem de *RREP* em todos os sentidos. Caso esta mensagem não alcance diretamente o responsável pelo envio do *RREQ*, uma mensagem de *RREP-ENC* é propagada até atingir o nó que enviou o RREQ. A execução termina quando todos os nós recebem a resposta de sua requisição inicial ou acontecem *deadlocks*. *Deadlock* caracteriza uma situação em que ocorre um impasse entre dois ou mais nós que ficam impedidos de continuar suas execuções, ou seja, ficam bloqueados.

A Tabela I apresenta a análise quantitativa[1] de troca de mensagens para o algoritmo de *Flooding*. A simulação foi realizada com a construção de uma rede de sete nós

[1]Problemas foram encontrados na execução dos experimentos para o algoritmo AODV, portanto sua tabela de análise quantitativa não será apresentada.

Topologia	RREQ	RREP	RREP-ENC
Ring	7	14	0
Mesh	7	18	0
Star	7	12	0
Line	7	12	0
Tree	7	10	72
Fully Conected	7	42	0

Tabela I
EXPERIMENTOS COM O ALGORITMO DE *Flooding*.

para cada uma das topologias apresentadas na Figura 5. Os parâmetros coletados mostram a quantidade de cada tipo de mensagem enviada pela rede.

V. ANÁLISE DE COMPLEXIDADE

Para algoritmos distribuídos, o principal fator a ser analisado é a ordem de complexidade das trocas de mensagens entre os nós de uma rede.

A complexidade do envio de mensagens é proporcional a troca de mensagens entre os nós. Dado o grau dos nós em determinada topologia, cada nó envia quantidade de mensagens proporcional ao seu grau, pois todos os seus vizinhos recebem a sua mensagem.

Considerando a topologia de um grafo completo, onde todos os nós estão interligados a todos os outros nós.

Suponha:

- n é o número de vértices em um grafo completo G.
- $d(n)$ é o grau do vértice, ou seja, número de arestas que chegam e saem do vértice.

Cada vértice do grafo contribuirá com:

$$\Theta(d(n)) = \Theta(n-1) \tag{1}$$

$$\sum d(n) = V^2 \text{ se completo.}$$

Assim, a ordem de complexidade das trocas de mensagem é representada por $(numero - de - vertices) \times (grau - do - vertice)$.

$$\Theta(n \times d(n)) = \Theta(n \times (n-1)) = \Theta(n^2 - n) = \Theta(n^2) \tag{2}$$

VI. CONCLUSÃO

Em redes *ad-hoc*, temos um vasto campo de pesquisa e vários desafios a serem vencidos. As redes sem fio quebraram com antigos padrões e possibilitaram a criação de novas aplicações impossíveis até então. Muitos problemas ainda devem ser vencidos, para que tenhamos redes sem fio funcionando de forma eficiente. Como vimos os algoritmos de roteamento estudados e apresentados tem características boas e ruins, dependendo das condições da rede. Ainda não existe nenhum algoritmo que tenha um desempenho razoável em todos os possíveis ambientes. O AODV foi apresentado como um algoritmo de roteamento dinâmico para redes *ad hoc*. Este algoritmo caracteriza-se principalmente pela realização de descoberta de rotas sob demanda, utilização de tabelas de roteamento, armazenamento de uma rota por destino, utilização de números de sequência, adoção de mecanismos que evitam *loops* e determinam as rotas mais atualizadas. Diversos trabalhos vêm sendo propostos como melhorias para resolver os problemas existentes no AODV. Desta forma, o AODV ainda pode ser considerado um grande desafio na área de roteamento em redes *ad hoc*.

VII. TRABALHOS FUTUROS

Em um próximo passo, estudaremos outro algoritmo distribuído de roteamento, o ATR *(Augmented Tree-based Routing)*, este algoritmo é apresentado por [10] propondo escalabilidade e resistência contra falha dos nós, congestionamento e instabilidade do canal de comunicação, características comuns em uma rede *ad hoc*.

Com a realização dessa etapa pretende-se propor um algoritmo distribuído de roteamento que combine as qualidades dos algoritmos AODV e ATR em busca de um algoritmo distribuído de roteamento, que apresente resultados melhores em relação ao número de troca de mensagens em redes *ad hoc*.

REFERÊNCIAS

[1] B. V. Fernandes, "Protocolos de roteamento em redes ad hoc," *UNICAMP: Programa de Pós-Graduação em Ciencia da Computação,Campinas, SP*, 2003. [Online]. Available: http://cutter.unicamp.br/document/?code=vtls000342497 - Acessado em 01 de Jun de 2011

[2] G. F. Amorim, M. C. M. Santos, A. L. A. Sobral, and P. R. L. Godin, "Roteamento em redes de comunicação sem fio ad hoc," 3º Simpósio de Pesquisa Operacional e 4º Simpósio de Logística da Marinha, 1999. [Online]. Available: http://www.gta.ufrj.br/~glauco/Spolm99_RoteamentoRedesComunicacaoSemFioAdHoc.htm - Acessado em 01 de Jul de 2011

[3] D. Câmara, "Roteamento em redes ad-hoc," Universidade Federal de Minas Gerais - Instituto de Ciências Exatas - Departamento de Ciência da Computação, 1998. [Online]. Available: http://www.eurecom.fr/camara/redes/roteamento.html - Acessado em 01 de Jul de 2011

[4] S. Corson and J. Macker, "Rfc 2501 - mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations," IETF Network Working Group, Tech. Rep., January 1999.

[5] M. Schneider, "Self-stabilization," *ACM Comput. Surv.*, vol. 25, pp. 45–67, March 1993. [Online]. Available: http://doi.acm.org/10.1145/151254.151256

[6] A. S. Tanenbaum, "Redes de computadores," *Rio de Janeiro: Editora Campus*, pp. 372–389, 2003.

[7] A. L. D. Moraes, A. F. dos Santos Xaud, and M. F. dos Santos Xaud, "Flooding." [Online]. Available: http://www.gta.ufrj.br/grad/09_1/versao-final/adhoc/olsrflash.html - Acessado em 12 de Jul de 2011

[8] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector routing," The Internet Engineering Task Force (IETF), Tech. Rep., 2003.

[9] Wikipédia, "Topologias de redes." [Online]. Available: http://pt.wikipedia.org/wiki/Ficheiro:NetworkTopologies.png - Acessado em 12 de Jul de 2011

[10] M. Caleffi, G. Ferraiuolo, and L. Paura, "Augmented tree-based routing protocol for scalable ad hoc networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, oct. 2007, pp. 1–6.