Desenvolvimento de uma Ferramenta de Auxílio ao Ensino de Redes de Computadores – Roteamento *Ad Hoc*

Fernando César Faria Cabral, Luís Augusto Mattos Mendes (Orientador)

Departamento de Ciência da Computação - Faculdade de Ciência da Computação e Comunicação Social (FACICS) - Universidade Presidente Antônio Carlos - Campos Magnus - Campolide - MG

nandofariacabral@hotmail.com, luisaugustomendes@yahoo.com.br

Resumo. *O roteamento é uma questão fundamental em qualquer rede, principalmente nas redes ad hoc, pois elas não possuem infra-estrutura montadas previamente e seus nós*[1] *podem se locomover, o que torna a sua topologia quase imprecisa. Este artigo propõe o desenvolvimento de um software que demonstre o roteamento para redes ad hoc para auxílio ao ensino no meio acadêmico.*

1 – Introdução

Com o aumento do tráfego de dados nas redes de comunicação, principalmente a internet, as pessoas, cada vez mais, necessitam de suas informações em uma velocidade mais rápida. Nas redes sem-fio há necessidade de uma maior agilidade nas trocas de dados, pois sua taxa de transferência é substancialmente inferior as redes cabeadas.

Atualmente as redes sem-fio podem ser classificadas como infra-estruturadas ou *ad hoc*. [1] Nas redes infra-estruturadas, como podemos ver na Figura 1, independente se os nós estão próximos, os pacotes de dados são encaminhados para uma estação de rádio base. Esta estação verifica se o destinatário está em sua área de cobertura, estando, lhe encaminha o pacote, se não, verifica em qual estação de rádio base ele se encontra e encaminha o pacote para esta estação.

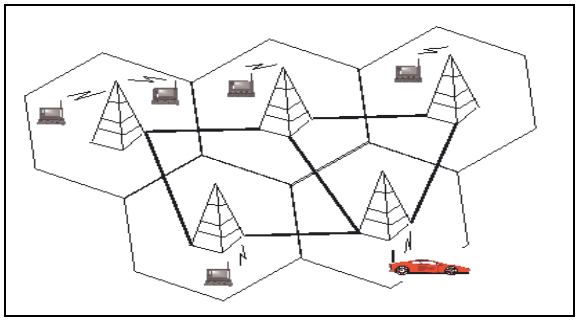

FIGURA 1. Rede sem-fio infra-estruturadas [1]

[1] Nó - Um Nó representa cada ponto de inter-conexão com rede.

Nas redes *ad hoc,* como podemos observar na Figura 2, não existe centralizador de informações, todos podem ser emissores, receptores ou atravessadores do pacote. Quando um nó recebe um pacote, ele verifica se este é para ele, não sendo, ele confere o caminho para o destinatário e envia a mensagem.

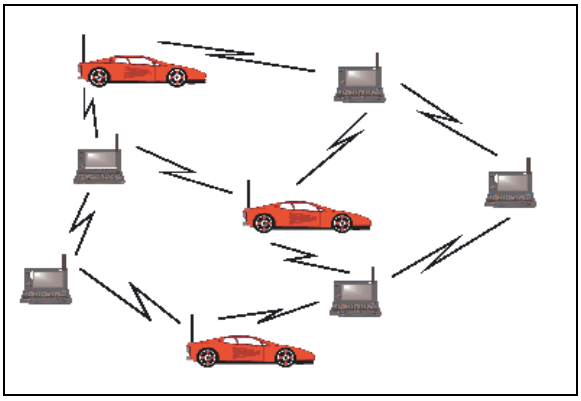

FIGURA 2. Redes *ad hoc* [1]

Outra razão para que os algoritmos de roteamento tenham uma maior importância nas redes *ad hoc* é que, como os nós são móveis tornam sua topologia dinâmica e dificultam a localização precisa do nó no momento em que se encaminha o pacote.

As redes *ad hoc* podem ser classificadas de diversas formas, no entanto, a mais utilizada é classificá-la em relação ao tipo de roteamento que ela utiliza. Quanto a esse quesito, podemos classificar uma rede *ad hoc* em duas categorias, as redes pró-ativas e as redes reativas. [1]

As redes pró-ativas ou orientada a tabela (*table-driven*) tentam manter a informação de roteamento de cada nó consistente e atualizada para todos os outros nós da rede. Estes protocolos necessitam que cada nó mantenha uma ou mais tabelas para armazenar a informação de roteamento, e se adaptam a mudanças na topologia da rede através da propagação de mensagens de atualizações que visam manter uma visão consistente de caminhos na rede. Os aspectos nos quais estes protocolos diferenciam são o número de tabelas necessárias e os métodos para difundir as mudanças na estrutura da rede. Foram propostos a implementação de dois algoritmos de roteamento pró-ativos, o DSDV *(Destination Sequenced Distance Vector)* e o GSR *(Global State Routing)* que serão discutidos nas seções 2 e 3 respectivamente.

As redes reativas ou sob-demanda (*on-demand*) só criam rotas quando estas são solicitadas pelo nó origem. Quando um nó requer uma rota para um destino, este inicia um processo de descoberta de rota dentro da rede. Este processo é completado se uma rota é encontrada ou se todas as possíveis permutações de rota tenham sido examinadas. Uma vez que uma rota tenha sido estabelecida, esta é mantida por um procedimento de manutenção de rota, até que o destino se torne inalcançável através de todos os caminhos a partir da origem, ou até que a rota não seja mais desejada. Foram propostos a implementação de dois algoritmos de roteamento para roteamento reativo, o DSR

(Dynamic Source Routing) e o AODV *(Ad Hoc On-Demand Distace Vector)* se serão discutidos nas seções 4 e 5 respectivamente.

O artigo está organizado da seguinte forma: As seções 2, 3, 4 e 5 apresentam, respectivamente, o funcionamento dos algoritmos DSDV, GSR, DSR e AODV. A seção 6 apresenta as características do *software* desenvolvido.

2 – *Destination Sequenced Distance Vector* - DSDV

O *Destination Sequenced Distance Vector* é um algoritmo pró-ativo baseado na idéia do algoritmo clássico de roteamento *Bellman-Ford*. [6,10]

No DSDV, cada nó mantém uma tabela de roteamento que lista todos os destinos disponíveis, o número de nó para alcançar o destino e o número seqüencial associado a este nó. O número seqüencial é usado para distinguir rotas velhas de novas, evitando assim a formação de *loops*.

A manutenção da tabela é feita através do envio de mensagens para atualização periódicas por cada nó, informando as alterações que ocorreram em suas tabelas devido as mudanças na topologia da rede. Existem dois tipos diferentes de mensagens para atualizar as rotas, as mensagens incrementais e as mensagens completas. [9,10]

Nas mensagens incrementais, enviam-se somente aquelas entradas da tabela que tiveram as suas métricas alteradas desde a última atualização. Este tipo é adequado para redes com movimentações relativamente estáveis.

Nas mensagens completas, enviam-se a tabela de roteamento completa. É adequada para redes com mudanças freqüentes de topologia.

Cada mensagem de atualização de rota, além de conter a informação da tabela de roteamento, também inclui um número seqüencial único. A rota com o maior número seqüencial é usada. Caso duas rotas tenham o mesmo número seqüencial, então, é utilizada a rota com a menor métrica, essa métrica pode ser o menor número de saltos, a menor distância geográfica, dentre outras.

As falhas de conexão podem ocorrer quando os nós se. Elas podem ser detectadas pelo protocolo da camada de enlace ou quando não se recebe nenhuma transmissão de um nó vizinho em um espaço de tempo. Quando uma falha ocorre, a rota é automaticamente marcada como invalida e é requerida uma atualização de rota. [8]

3 - *Global State Routing* - GSR

O *Global State Routing* é um protocolo baseado no protocolo de estado de enlace (*link state* - LS). Como no LS, este protocolo mantém uma visão global da rede podendo fazer cálculos precisos de rota, contudo, utiliza técnicas para limitar o número de mensagens de controles que são enviadas, o que o leva a redução de consumo de energia e de largura de banda. [10] Uma das características mais interessantes do GSR é a

capacidade de manipular parâmetros QoS[2], segundo Câmara, esta característica é ainda pouco explorada em algoritmos de roteamento para redes *ad hoc*, mesmo sendo esta uma das mais importantes características para tráfego multimídia, podendo inclusive ser esta uma das mais importantes funções das redes *ad hoc* no futuro.[4]

Cada nó do GSR possui 4 tabelas, uma Lista de Vizinhos, uma de Topologia, uma dos Próximos Saltos e uma de Distância.[4] A Lista de Vizinho contém a relação de todos os nós que podem ser alcançados diretamente por ele, ou seja, os nós que não há necessidade de roteamento. A tabela de Topologia contém a informação sobre o estado de enlace para cada possível nó de destino. A tabela de Próximos Saltos mantém o próximo nó aos quais os pacotes deverão ser encaminhados para chegar a um destino. A tabela de Distância armazena a menor distância para cada nó destino.

As mensagens de roteamento são geradas na alteração de algum enlace, como ocorre nos protocolos LS. Ao receber uma mensagem de roteamento, o nó atualiza sua Tabela de Topologia se, somente se, o número seqüencial armazenado nesta tabela for mais novo.

Após isso, o nó reconstrói sua tabela de roteamento e difunde a informação aos seus vizinhos, mas apenas para os seus vizinhos, ao contrário do LS que envia para todos os nós da rede. No estado inicial, a Lista de Vizinhos e a Tabela de Topologia ficam vazias, sendo necessário atribuir valores de inicialização para as tabelas internas. [10]

Existem duas técnicas para reduzir o tamanho das mensagens de atualização, o *Fresh Update* e o *Fisheye* (olho de peixe), que serão discutidas nas sub-seções a seguir.

3.1 - *Fresh Update*

No *Fresh Update*, somente as informações úteis aos vizinhos são encaminhadas. A idéia é simples, se o seu vizinho contém uma informação mais recente, não faz sentido enviar esta informação para ele. [4] Para a realização desta técnica é necessário que cada nó contenha um Vetor de Seqüência de Números (VSN) onde, além de armazenar todos os destinos que estão na tabela, indica a idade de cada informação enviada.

O *Fresh Update* é feito em dois passos: [4]

1. O nó envia um VSN para seus vizinhos
2. Ao receber o VSN compara ao seu VSN. Se alguma entrada de seu VSN for mais antiga, esta entrada é removida das mensagens de atualização. Após isso, o resultado das atualizações de tabelas é disseminado.

[2] QoS – Sigla para *Quality of Service* (Qualidade de Serviço).

3.2 – *Fisheye*

O *Fisheye* tem como objetivo manter um volume alto de informações sobre os nós mais próximos e diminui o detalhamento das informações mais distantes, desta forma, reduz o tamanho da mensagem de atualização. O escopo de detalhamento é definido em termos dos nós que podem ser alcançados em certos números de saltos. Apesar de um nó não ter uma informação precisa sobre os nós mais distantes, os pacotes são roteados corretamente porque a informação de roteamento se torna mais precisa à medida que o pacote se aproxima do destino [10].

4 - *Dynamic Source Routing* - DSR

O *Dynamic Source Routing* é um algoritmo classificado como reativo, isto é, com roteamento na fonte. Assim sendo, o nó origem tem que informar por quais outros nós o pacote de dados deverá ser transmitido para chegar até o destino. Diferentemente de outros algoritmos, o DSR não encaminha mensagens periódicas para atualização de rotas, o que acarreta em uma economia de banda e de energia. [3] Foi projetado para suportar redes *ad hoc* com até aproximadamente 200 nós e altas taxas de mobilidades dos nós [2].

Cada nó da rede mantém um *cachê* de rotas, onde armazenam-se as rotas para outros nós que ele aprendeu iniciando um pedido de requisição de rotas ou por ter encaminhado pacotes ao longo de outro caminho que passasse por este nó. Além desses procedimentos, um nó pode também aprender novas rotas ouvindo a transmissão de seus vizinhos, este procedimento é opcional no DSR, e as interfaces de rede deverão funcionar em modo promíscuo para o seu funcionamento. O *cachê* de rotas permite o armazenamento de mais de uma rota para um mesmo destino. [2]

Quando um nó fonte precisa enviar dados para algum destino ele consulta seu *cachê* de rotas e verifica se possui uma rota válida para o destino. Caso possua, ele usará esta rota para fazer o envio dos dados. Caso contrário, um processo de descoberta de rotas deverá ser inicializado com o envio de pacote de uma requisição de rota (*route request)*. O quadro de requisição de rota é formado pelo endereço de destino acompanhado com o endereço do nó fonte e um número de identificação único. Cada nó que recebe este quadro faz uma verificação para saber se conhece a rota desejada. Caso não conheça, este nó acrescentará ao pacote de requisição de rota seu próprio endereço e o enviará aos seus vizinhos e assim sucessivamente. Para limitar o número de requisição de rota propagada, um nó só retransmite este pacote se ele ainda não passou por este nó, ou seja, se ele não contém o seu próprio número de endereço.

O DSR é composto por dois mecanismos, o mecanismo de descobrimento de rotas e o mecanismo de manutenção de rotas, que serão tratados nas sub-seções seguintes.

4.1 - Descobrimento de Rotas

Descobrimento de rotas é o mecanismo pelo qual um nó de origem obtém, dinamicamente, uma rota de um nó destino. Este mecanismo é usado todas as vezes que o nó de origem não contém uma rota para o destino em seu *cachê* de rotas.

Ao iniciar o descobrimento de rotas, o nó origem transmite uma requisição de rota para aproximadamente todos os nós que estão atualmente em sua área de alcance. Cada requisição de rota consta com a identificação do nó que o originou, a identificação do nó de destino, uma lista de nós pelo qual a requisição de rota passou. É também inserido na sua criação um número único de requisição, gerado pela origem. A lista de nós é inicializada com a identificação do nó que originou a requisição de rota.

Quando um nó recebe a requisição de rota, se é o objetivo do descobrimento de rotas devolve um retorno de rota (*route reply)* ao nó origem. A resposta ao nó origem pode ser encaminhado de três formas: através de uma rota presente no cachê de rotas; através da realização de um novo processo de descobrimento de rotas; ou utilizando o caminho inverso da lista de nós da requisição de rota. [3] Neste último caso, para que essa inversão ocorra, é necessário que o protocolo MAC permita um *link* de transmissão bidirecional, o que também evita uma possível segunda descoberta de rotas. [2]

Caso o nó não seja o destino, o nó intermediário verifica se contém uma rota até o destino em seu *cachê* de rotas, contendo, encaminha um retorno de rota inserindo também o caminho de seu *cachê*. Caso contrário, este nó junta o seu próprio endereço para a lista de nós e o propaga a requisição de rota.

Quando um nó recebe o retorno de rota, verifica se é o destino da resposta. Sendo o destino, adiciona a rota formada pela lista de nós no seu *cachê* de rotas e a utiliza para encaminhar o pacote de dados ao destino. Não sendo, envia o retorno de rota para o próximo nó da lista.

Para impedir uma inundação desnecessária de requisições, os nós possuem uma lista de requisição recente, contendo a identificação de todas as requisições recebidas recentemente. Assim, ao receber uma requisição, o nó verifica se já recebeu aquela mesma requisição. Tendo recebido, ignora a requisição, não a repassando a diante.

4.2 - Manutenção de Rotas

Manutenção de rotas é o mecanismo através do qual o nó origem detecta se ocorreram alterações na topologia da rede que poderão comprometer o uso das rotas. Cada nó pode monitorar os pacotes de confirmação de outros nós ou mesmo trabalhar em modo promíscuo, onde "escuta" as comunicações que passam por ele.

O DSR utiliza dois tipos de pacotes para manutenção de rotas: pacote de confirmação (A*cknowledgments*) e pacote de erro de rotas.

Pacotes de confirmação são utilizados para verificar o funcionamento correto dos enlaces de rota. Um pacote de confirmação é fornecido freqüentemente sem custo, como parte padrão do protocolo MAC em uso, ou por um uso passivo, no qual, por exemplo,

um nó B confirma o recebimento do nó C escutando C transmitindo o pacote para um nó D. [2]

O pacote de erro de rota é criado quando um nó encontra um problema de transmissão fatal, ou seja, quando o nó verifica que não pode encaminhar a mensagem para o próximo nó identificado no pacote e não conhece um outro caminho para o nó destino. Quando um nó recebe um pacote de erro de rota, este atualiza seu *cachê* de rota descartando, parcialmente, a rota utilizada até então. Este descarte é feito a partir do nó em que foi identificado o erro ou não conseguiu fazer a comunicação com o próximo nó. [1]

5 - *Ad Hoc On-Demand Distance Vector* - AODV

O *Ad Hoc On-Demand Distance Vector* é um algoritmo de vetor de distância e também pode ser classificado como reativo, já que uma rota só é criada se for necessária.

O AODV funciona de maneira muito similar ao DSR descrito na seção 4. Quando uma rota é requisitada, ele utiliza um processo de descoberta de rota similar ao do DSR, contudo, o AODV utiliza um mecanismo diferente para armazenar as informações de roteamento. Ao invés de armazenar toda a rota, ele utiliza uma tabela de roteamento tradicional, o que significa que ele mantém apenas uma entrada para o destino. Essa entrada armazena apenas o próximo salto para esse destino, enquanto o DSR armazena a rota completa, e até múltiplas rotas para o mesmo destino. [5,7]

Para encontrar o caminho para o destino, a origem difunde uma requisição de rota. Os vizinhos por sua vez também difundem a requisição de rota para seus vizinhos até que chegue a um nó intermediário que tenha uma informação recente de rota para o destino, ou até que chegue ao próprio destino. Um nó descarta uma requisição de rota que já tenha processado.

Quando um nó encaminha a requisição de rota para seus vizinhos, este grava em suas tabelas o nó o qual a primeira cópia da requisição veio. Esta informação é usada para construir o caminho reverso para o retorno de rota. O AODV usa somente *links* bidirecionais porque o retorno de rota seque o caminho reverso da requisição de rota. [7]

5.1 - Manutenção de Rotas

Cada nó mantém uma Tabela de Rotas de rotas ativas para cada destino como as seguintes informações:

- Destino – Nó destino da rota;
- Próximo Salto – Próximo nó para o qual tem que se enviar o pacote para chegar ao destino.
- Saltos – Número de saltos que o pacote tem que fazer para chegar ao destino
- Número seqüencial para o destino – Atributo de controle para comparar rotas antigas das novas.

- Tempo de duração para a rota (*timeout*) – Tempo de utilização da rota. Quando o *timeout* chega ao fim, o nó considera essa rota como invalida.

Toda vez que uma tabela de rotas é usada para transmitir dados da origem para o destino, o tempo de duração para a rota é reiniciado. Caso uma nova rota seja oferecida para o nó, ela será utilizada se o número seqüencial de destino for maior ou igual (com menor número de saltos) ao existente no nó. [7]

Um nó ao detectar uma falha de conexão, propaga uma mensagem de notificação de falha de conexão, que consiste em transmitir para seus vizinhos um retorno de rota com número seqüencial de destino maior do que o existente na rota e com o valor de números de saltos igual a 0.[7] Ao receber a notificação, caso os nós fonte continuem necessitando de uma rota ao destino, inicializarão o processo de descoberta de rota transmitindo uma requisição de rota com um número seqüencial de destino maior do que os anteriores.[8]

5.2 - Mensagem de *Hello*

Para manter as rotas funcionado de forma correta, o AODV exige que cada nó envie uma mensagem de *Hello* periodicamente aos seus vizinhos. Esta mensagem significa que o nó continua presente e que as rotas dependentes dele continuam válidas. Se algum nó parar de enviar mensagens de *hello*, o vizinho assume que o nó se moveu e assinala o *link* como nó perdido. O gerenciamento da conectividade local com mensagens *hello's* pode também ser usado para assegurar que somente nós com conectividade bidirecional sejam considerados como vizinhos. [4,7,9]

6 – *Software Ad Hoc*

Está seção tem como objetivo mostrar o desenvolvimento do *software* de auxílio ao ensino, Ad Hoc, um sistema computacional voltado para demonstrar o funcionamento dos algoritmos de roteamento para redes *ad hoc.*

6.1 – Ferramentas utilizadas

Para a modelagem do sistema, descrita no anexo A, foi utilizada a ferramenta CASE[3] *Rational Rose*, desenvolvido pela *Rational Software Corporation*, que modela utilizando os padrões da UML[4].

A implementação do software foi desenvolvida utilizando o Delphi 2005, um programa de desenvolvimento visual que permite a criação de sistemas em um curto espaço de tempo, quando comparado a outros ambientes de desenvolvimento.

---

[3] CASE – Sigla para *Computer Aided Software Engineering* (Tecnologia de programação assistida por computador). Uma ferramenta CASE é uma classificação que abrange toda ferramenta baseada em computadores que auxiliam atividades de engenharia de software.

[4] UML – Sigla para *Unified Modeling Language* (Linguagem de modelagem unificada).

6.2 – Considerações sobre o *software*

Algumas considerações se fazem necessárias dizer a respeito sobre o software desenvolvido:

- A rede trabalha em um ambiente desprovido de qualquer tipo de interferência, seja ela elétrica ou física (obstrução de materiais sólidos).
- Todos os nós cadastrados estão dispostos a participar completamente da rede, isto é, podem ser remetentes, destinatários ou intermediários do roteamento do pacote.
- Por não haver interferência, não há pacotes corrompidos ou perdidos. Conclui-se assim que, todo pacote enviado chega a nó para o qual ele foi encaminhado.
- Não há pacote de erros, os erros de comunicação são tratados anteriormente de se começar a execução da demonstração. Para os algoritmos DSDV e GSR, descritos nas seções 2 e 3, antes de se iniciar a demonstração tabelas de rotas são recriadas. Já nos algoritmos DSR e AODV, descritos nas seções 4 e 5, as rotas que ocorreram quebras de *link* são excluídas da tabela de rotas deste nó.
- Todos os *links* são bidirecionais, ou seja, se um nó A pode encaminhar um pacote para B, consequentemente, o nó B pode encaminhar um pacote para A.
- Por haver apenas um envio de mensagem por demonstração, foi implementado somente uma variável para cada tipo de pacote que o nó possa utilizar para comunicar.

O funcionamento do *software* será apresentado na subseção a seguir.

6.3 – Funcionamento do Software Ad Hoc

6.3.1 – Tela Principal

A Figura 3 apresenta a tela principal do sistema, o usuário pode utilizar as funcionalidades do sistema através da barra de ferramentas e ou utilizando a barra de botões que se encontra abaixo da barra de ferramentas.

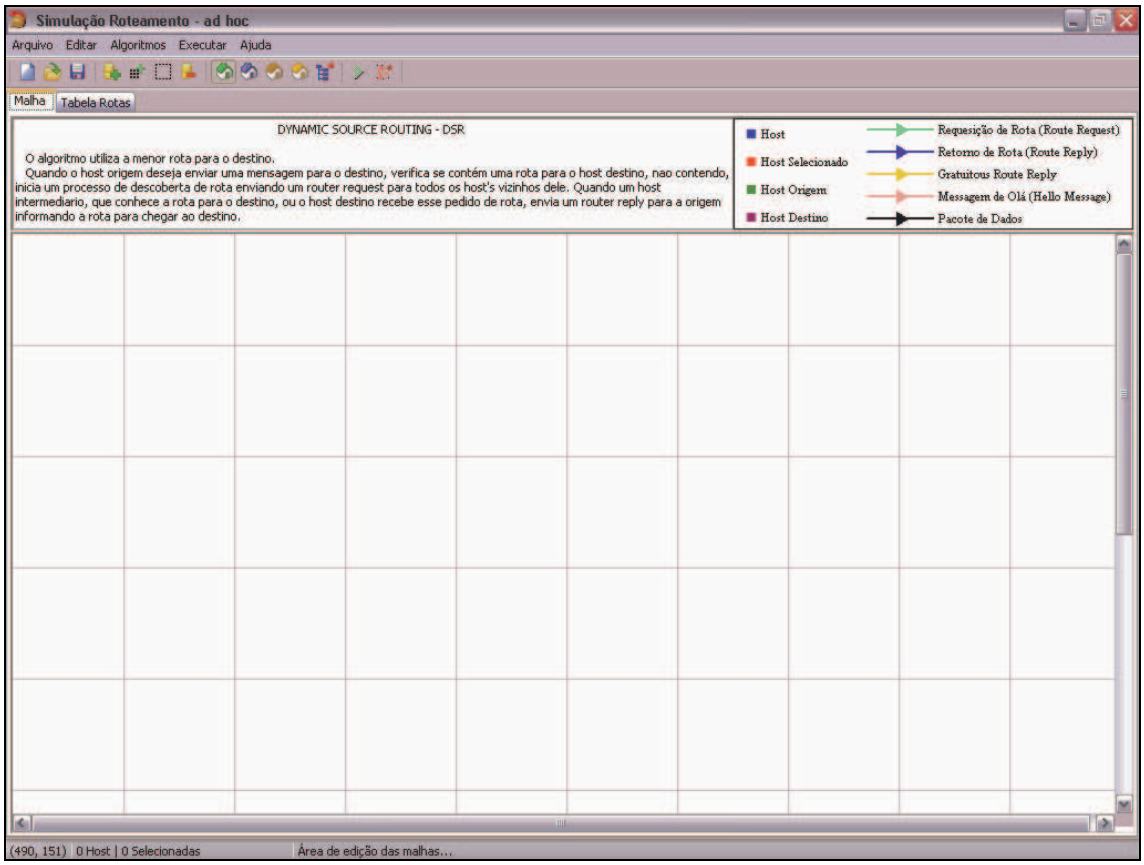

FIGURA 3. Tela principal do software ad hoc

Encontram-se também na tela principal, uma pequena descrição do funcionamento do algoritmo selecionado e a legenda do sistema. A legenda é composta por 11 símbolos: 4 quadrados de cores diferentes representando o estado que o *host*[5] se encontra e 5 setas que representa os tipos de pacote de dados que pode aparecer na demonstração de funcionamento. Logo abaixo da descrição e da legenda temos a área de edição das malha, onde é possível visualizar os *host's* cadastrados. A área de edição das malhas tem uma medida de 1.000.000 m² (1000m x 1000m).

Na Figura 4 temos a barra de botões, esta barra foi divida em 4 grupos de funcionalidades. O primeiro grupo contém os botões novo, abrir e salvar malha. O segundo grupo é de manipulação de *host's*, nele tem os botões criar um *host*, criar vários *hosts*, selecionar tudo e deletar *host's* selecionados. O terceiro grupo é dos botões de escolha do algoritmo de roteamento e deletar tabela de rotas, nele temos os botões para os algoritmos DSR, AODV, DSDV e GSR respectivamente e um quinto botão limpar tabela de rotas. O quarto grupo é composto pelo botão executar e pelo botão limpar tela que, limpa a área de edição de malha e desenha os *host's* novamente.

---

[5] *host* - O *host* é qualquer máquina ou computador conectado a uma rede. Os *host's* variam de computadores pessoais a supercomputadores, dentre outros equipamentos, como roteadores. No caso da rede *ad hoc*, host é um equipamento, com capacidade de mobilidade, que se conecta a uma rede através de um equipamento de comunicação sem-fio.

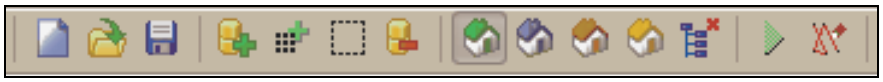

FIGURA 4. Barra de Botões

Podemos observar na Figura 5 duas abas para modificar o que será mostrado na tela. Na primeira aba, Malha, mostra a área de edição de malhas e na segunda aba, Tabela Rotas, deixa visível na tela à tabela de rotas dos *host's*.

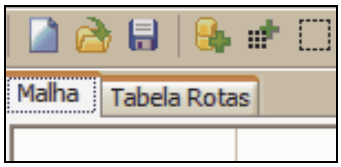

FIGURA 5. Abas

Ao se selecionar a aba Tabela Rotas, será mostrada na tela uma barra como podemos observar na Figura 6. Esta barra contém 4 botões para selecionar qual algoritmo se deseja ver a tabela de rotas, o botão atualizar para atualizar as informações pois, ao executar uma demonstração os dados mostrados na tela pode ficar desatualizados. Contém ainda, uma área de edição de texto para selecionar de qual *host* será a tabela de rotas.

FIGURA 6. Barra para configurar visualização de tabela de rotas

6.3.2 – Inserir Host

Existe duas formas para se inserir um *host*: inserir um *host* e inserir vários *host's*.

Para inserir um único *host*, um formulário como o da Figura 7 aparecerá na tela. Neste formulário o usuário insere a posição do *host*, o raio de cobertura da placa de rede e um nome para identificá-lo. A posição do *host* é composta pelas coordenadas (x,y) são utilizadas para informar a posição que o novo *host* terá área de edição de malha, os valores validos para as coordenadas correspondem entre (0,0) e (1000,1000), que são os limites da área de edição das malhas. O raio de cobertura corresponde ao alcance maximo para se ter comunicação entre esse *host* e um outro, o valor do raio é definido entre 10m e 200m. O nome é utilizado para que, ao desenhar o *host*, o usuário possa identificá-lo com mais facilidade.

Na ausência de algum desses dados, o sistema utiliza um valor *default* para o mesmo. Os valores *default* são: para a posição as coordenadas (500,500) que correspondem ao centro da área de edição; para o raio de cobertura o valor é de 100m

definido pelo padrão IEEE 802.11 b/g; para o nome é utilizado a concatenação das frases ‘host ‘ e o número de *host’s* já cadastrados.

FIGURA 7. Inserir um *host*

Para inserir vários *host’s*, um outro formulário, idêntico a Figura 8, aparecerá na tela para que, o usuário informe o número de *host’s* que se deseja criar e a forma de distribuição. Os *host’s* podem ser distribuídos de forma aleatória ou no grid. Inserindo de forma aleatória, as posições são distribuídas de forma randômica. Já na forma de grid, as posições são definidas em função a um cálculo entre o tamanho da área de edição e o número de *host’s* desejados.

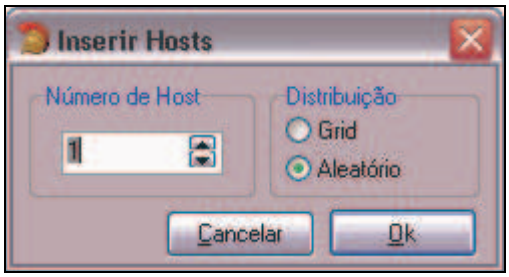

FIGURA 8 - Inserir vários *host's*

6.3.3 – Executar algoritmo

Na Figura 9 temos a tela de configuração de execução, nele consta três áreas de edição de texto, um para selecionar o *host* de origem do pacote, outro para selecionar o *host* de destino do pacote e um terceiro para indicar o tempo de vida do pacote, medido em saltos. Não informando a origem e ou o destino, o sistema seleciona um *host* aleatoriamente. O tempo de vida é utilizado somente para o pacote de requisição de rotas, seu intervalo valido é entre 0 e 255 saltos, sendo que, o valor *default* é de 255.

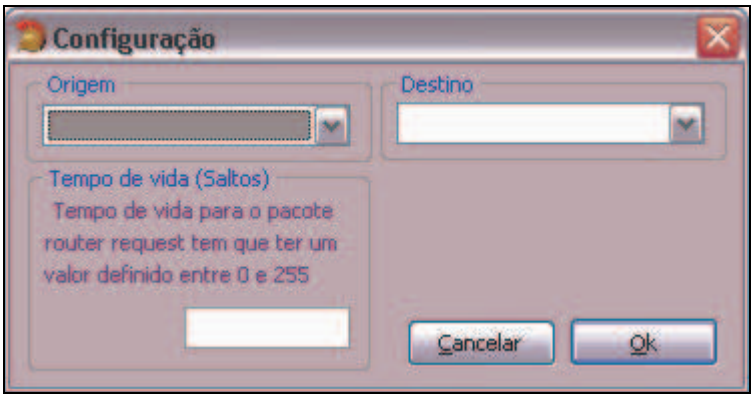

FIGURA 9. Executar algoritmo

Após confirmar a execução, o sistema verifica qual dos algoritmos disponíveis foi o selecionado e inicia a demonstração. No momento que se inicia os *host's* origem e destino são desenhados com outras cores, verde e lilás respectivamente, para facilitarem sua visualização e começa a aparecer setas entre os *host's* simbolizando o envio de algum pacote, como é mostrado na Figura 10.

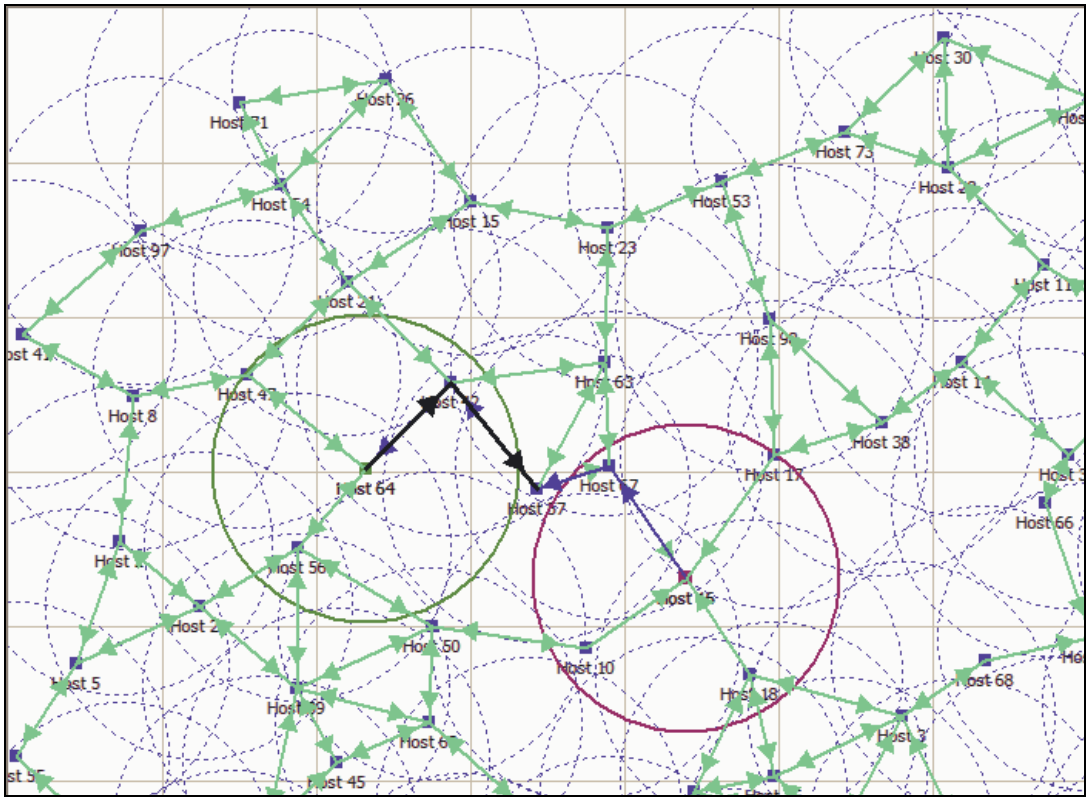

FIGURA 10. Algoritmo em execução

Na Figura 10 representa uma demosntração utilizando o algoritmo DSR tendo como origem o *host* 64 e destino o *host* 16. Como a origem não conhecia o caminho para o destino, enviou uma requisição de rota (setas de cor verde) para todos os seus

vizinhos por inundação. No momento que chegou a requisição de rota no destino, ele encaminhou origem um retorno de rota (setas de cor azul) para a origem. Ao receber o retorno de rota, a origem envia o pacote de dados (setas de cor preta) utilizando a rota do retorno de rota.

6.3.4 – Exportar Tabelas de Rotas

No menu Arquivo, da Barra de Ferramentas, existe uma opção de exportar as tabelas de rotas dos *host's,* o escolhendo, aparecerá na tela um formulário como o da Figura 11. Como se pode observar, temos a opção se selecionar os algoritmos que terão as tabelas exportadas. O arquivo que será criado é de extensão CSV, que pode ser aberto por uma planilha de texto. Foram observados nos testes realizados que existe uma incompatibilidade entre o Excel da Ferramenta Office 2000 com as Superiores, então, no formulário contém uma opção que deverá ser marcada se, no computador que será lido o arquivo, utiliza a Ferramenta Office XP ou superior.

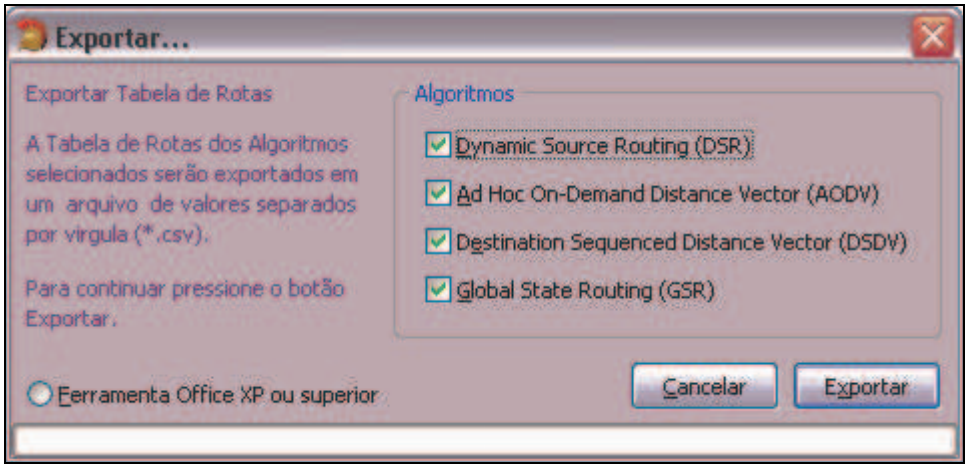

FIGURA 11 - Exportar Tabela de Rotas

7 – Considerações Finais

O roteamento, independente de qual tipo de redes está se tratando, é uma questão muito complicada para se compreender. Nas redes *ad hoc*, a situação pode ser mais agravante pela questão da mobilidade que os nós possuem.

Observa-se que, na utilização de aplicativos com interface gráfica, os alunos conseguem compreender com uma maior facilidade a disciplina ministrada. Por este motivo, o *software* ad hoc pode ser utilizado no meio acadêmico para auxiliar o professor a demonstrar o funcionamento de roteamentos para redes *ad hoc*.

Deixa-se em aberto diversos temas para trabalhos futuros, tais como: desenvolvimento de outros *softwares* que demonstrem o funcionamento de roteamento para outros tipos de rede, uma nova versão para esse *software* contendo outros

algoritmos implementados, tratamento de pacotes de erros, envio de diversos pacotes em uma mesma execução e ou tratamento de interferências.

8 – Referências Bibliográficas

[1] SILVERIO, Marcio Lopes. *Redes Ad Hoc: Protocolos de Roteamento e Segurança*, Barbacena, jul. 2005.

[2] IETF MANET WORKING GROUP. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)* Disponível em: < http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt> Acesso: 12 fev. 2006

[3] DUARTE, Lucio Mauro. *Desenvolvimento de Sistemas Distribuídos com Código Móvel a partir de Especificação Formal* Disponível em <http://www.pucrs.br /inf/pos/dissertacoes/resumos/lucio.htm> Acesso: 15 mar. 2006.

[4] CÂMARA, Daniel. *Um novo algoritmo de roteamento para redes Móveis Ad Hoc* Disponível em: <http://www2.dcc.ufmg.br/%7Edanielc/dissertacao/dissertacao _gpsal.html> Acesso: 12 fev. 2006.

[5] AMODEI JUNIOR, Aurélio; DUARTE, Otto Carlos M. B. *Segurança no Roteamento em Redes Moveis Ad Hoc* Disponível em: <www.gta.ufrj.br/seminarios/CPE825/tutoriais/aurelio/AmDu03.pdf> Acesso: 09 abr. 2006.

[6] MONTES, Luis Renato; BERNADO, Rodrigo; DE OLIVEIRA, Vinícius Garcia. *Protocolos de Roteamento e Protocolos TCP/UDP em Redes Ad Hoc* Disponível em: < http://www.teleco.com.br/tutoriais/tutorialprotocolo/default.asp > Acesso: 08 abr. 2006.

[7] IETF MANET WORKING GROUP. *Ad hoc On-Demand Distance Vector (AODV) Routing* Disponivel em: <http://moment.cs.ucsb.edu/pub/draft-perkins-manet-aodvbis-00.txt> Acesso: 01 maio 2006.

[8] FERNANDES, Bruno Vieira. *Protocolo de Roteamento em Redes Ad Hoc*, Disponível em <http://libdigi.unicamp.br/document/?code=vtls000342497> Acesso: 01 maio 2006.

[9] ALBURQUERQUE, Luciano Renovato. *Segurança em Redes Ad Hoc*, Disponível em <www.lockabit.coppe.ufrj.br/downloads/academicos/seguranca_ redes_adhoc.pdf> Acesso: 14 dez. 2005.

[10] GODIM, Paulo Roberto de Lira, *et al*. *Roteamento em redes de comunicação sem fio ad hoc* Disponível em <http://www.gta.ufrj.br/ ~glauco/Spolm99_RoteamentoRedesComunicacaoSemFioAdHoc.htm> Acesso: 27 maio 2006.