Daniel Câmara

Estudo de Algoritmos de Roteamento para Redes Móveis Ad Hoc

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

16 de junho de 2000

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Estudo Sobre Algoritmos de Roteamento para Redes Móveis Ad Hoc

DANIEL CÂMARA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Antônio Alfredo Ferreira Loureiro – Orientador
Departamento de Ciência da Computação – ICEx – UFMG

Prof. Claudionor José Nunes Coelho Júnior
Departamento de Ciência da Computação – ICEx – UFMG

Prof. Geraldo Robson Mateus
Departamento de Ciência da Computação – ICEx – UFMG

Prof. José Marcos Silva Nogueira
Departamento de Ciência da Computação – ICEx – UFMG

Belo Horizonte, 27 de março de 2000.

If we knew what it was we were doing,
it would not be called research, would it?

(Se nós soubéssemos o que nos estávamos fazendo,
isto não seria chamado de pesquisa, seria?)

–Albert Einstein

Resumo

Este trabalho tem como objetivo o estudo das redes de computadores móveis ad hoc. Redes ad hoc são redes de computadores móveis que têm a característica de poderem se comunicar diretamente sem a necessidade de uma infra-estrutura fixa. Será apresentado aqui uma caracterização da área de computação móvel de forma geral e de redes ad hoc. Algoritmos de roteamento de uma forma geral serão discutidos, pois são a chave para que se possa entender como os algoritmos de roteamento para redes ad hoc funcionam. Só então serão apresentados os principais algoritmos de roteamento para redes ad hoc e uma comparação entre suas características. Como resultado destes estudos foi criado um novo e eficiente algoritmo de roteamento para redes ad hoc chamado GPSAL (*GPS Ant Like Routing Algorithm*). O algoritmo proposto foi simulado para diversos cenários e comparado com outro algoritmo de roteamento da mesma classe.

Abstract

The main goal of this work is the study of routing algorithms in mobile ad hoc networks. Computers on such networks can communicate directly with each other, without the support of base stations. We will present here a characterization of mobile networks with emphasis on the ad hoc environment. General routing algorithms will also be presented in order to understand routing algorithms for ad hoc networks. The main ad hoc routing algorithms will be presented and their main characteristics will be analysed and compared. The result of this study is the specification of a new and efficient ad hoc routing algorithm called GPSAL (GPS Ant-Like Routing Algorithm), which was implemented and simulated. We simulate the algorithm for different scenarios and compare it to another algorithm also based on GPS.

Agradecimentos

Diversas pessoas colaboraram para que este trabalho pudesse ser realizado. Agradeço aos meus pais, *Pedro Izidoro Câmara* e *Maria Rosa Câmara*, que mesmo a distância me deram apoio e incentivo para a realização do mestrado. Ao meu irmão *Marcelino Câmara*, que resolveu inúmeros problemas pessoais em Curitiba, o que me garantiu a tranqüilidade que necessitava em Belo Horizonte.

Agradeço também ao meu orientador, o professor *Antonio Alfredo Ferreira Loureiro*, por além de ter dividido o seu conhecimento, de me corrigir em diversos momentos quando eu estava indo por caminhos errados, também assumiu o papel de amigo e conselheiro.

A todo o pessoal do *Grupo de Estudos de Redes Móveis Ad hoc (GEDOC)*, por me auxiliarem nas pesquisas e pelas intermináveis discussões sobre possibilidades e novos caminhos que poderiam ser seguidos.

Agradeço principalmente a *Wanessa Nascimento Matta*, minha noiva, pelo apoio e carinho nos momentos em que mais precisei, e também pelas intermináveis correções do texto, que sem dúvida melhoraram a qualidade desta dissertação.

Quero agradecer ao *CNPq* pela bolsa de pesquisa, que foi fundamental para que eu pudesse realizar meus estudos, pesquisando em redes móveis.

E por último, mas de forma alguma de menor importância, também gostaria de agradecer a *Dona Antonia* do cafezinho. Que muitas vezes é esquecida, mas que desempenhou um papel fundamental na conclusão desta dissertação, e acredito que de muitas outras também. Você pode até não acreditar mas foi necessário muito café para concluir o mestrado.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Programas

Capítulo 1

Introdução

O crescimento extraordinário que tem ocorrido nesta década nas áreas de comunicação celular, redes locais sem fio e serviços via satélite permitirá, em um futuro próximo, que informações e recursos possam ser acessados e utilizados em qualquer lugar e em qualquer momento. Dado o atual crescimento do segmento de computadores pessoais portáteis e PDAs (*Personal Digital Assistants*), estima-se que em poucos anos, dezenas de milhões de pessoas terão um *laptop*, *palmtop* ou algum tipo de PDA. Independente do tipo de dispositivo portátil, a maior parte desses equipamentos deverá ter capacidade de se comunicar com a parte fixa da rede e, possivelmente, com outros computadores móveis. A esse ambiente de computação se dá o nome de computação móvel ou computação nômade. Nesse ambiente, o dispositivo computacional[1] não precisa ter uma posição fixa na rede.

Esse novo paradigma permite que usuários desse ambiente tenham acesso a serviços independente de onde estão localizados e, o mais importante, de mudanças de localização, ou seja, proporcionando aos usuários mobilidade. Dessa forma, a computação móvel amplia o conceito tradicional de computação distribuída. Isso é possível graças à comunicação sem fio que elimina a necessidade do usuário manter-se conectado à uma estrutura fixa e, em geral, estática.

1.1 Tipos de Redes Móveis

Segundo o grupo de estudo 802.11 do IEEE (*Institute of Electrical and Electronics Engineers*), as redes sem fio podem ser classificadas como estruturadas ou independentes (*ad hoc* [3, 20]).

1.1.1 Redes estruturadas

Redes estruturadas são aquelas em que o *Host* Móvel (HM) está em contato direto com uma Estação de Suporte à Mobilidade (ESM) na rede fixa. Esta ESM normalmente está ligada a uma rede fixa de alta velocidade.

[1] Os termos computador móvel, *hop*, nodo e *host* serão usados no decorrer da dissertação para referenciar, de forma genérica, um computador portátil ou um PDA .

Neste tipo de rede sem fio, toda a comunicação é feita via ESM (figura 1.1). O funcionamento deste tipo de rede móvel é semelhante ao da telefonia celular, onde toda a comunicação deve necessariamente passar pela central, mesmo que os telefones estejam a uma distância em que poderiam, eventualmente, comunicar-se diretamente.

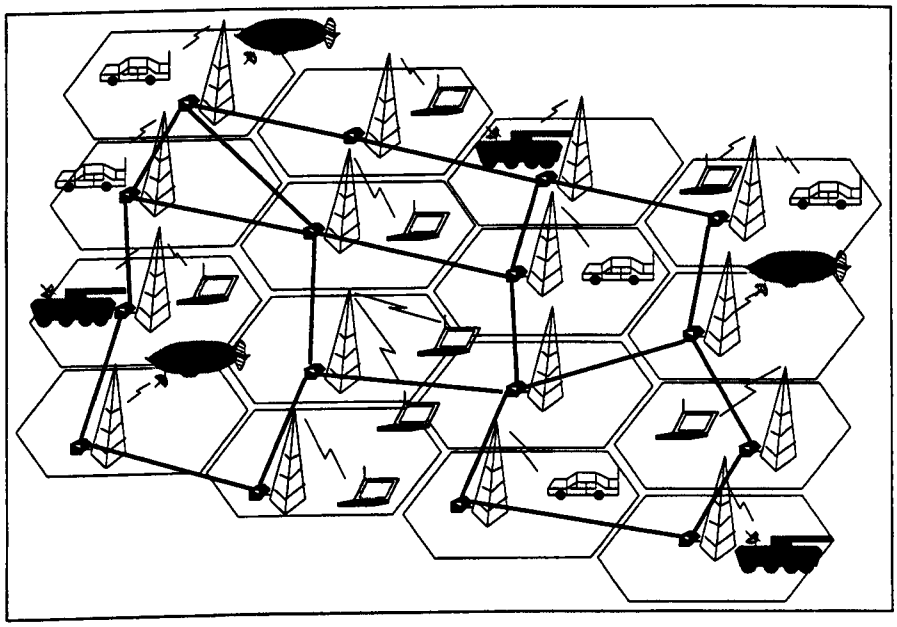


Figura 1.1: Modelo de comunicação em redes móveis estruturadas.

1.1.2 Redes Ad hoc

O outro tipo importante de redes móvel é a rede ad hoc (MANET–*Mobile Ad hoc NETwork*), onde os dispositivos computacionais são capazes de trocar informações diretamente entre si, conforme mostrado na figura 1.2. Redes ad hoc são principalmente indicadas para situações onde não se pode, ou não faz sentido, instalar uma rede fixa.

Um cenário onde uma rede móvel ad hoc pode ser usada é por exemplo, numa situação de desastre, como furacão, terremoto ou inundação, onde equipes de resgate precisam se coordenar e não se tem uma rede fixa disponível. Soldados num campo de batalha trocando informações táticas, empresários compartilhando informações numa reunião ou estudantes usando *laptops* para participar de uma aula interativa são também exemplos de situações onde redes ad hoc ser necessárias. Outra aplicação onde MANETs deverão ter no futuro um papel importante é na interconexão de *wearable computers* [5].

Numa MANET uma rota entre dois computadores pode ser formada por vários *hops* através de um ou mais computadores na rede, como visto na figura 1.3. Um dos problemas fundamentais numa rede ad hoc é determinar e manter as rotas, já que a mobilidade de um computador pode causar mudanças na topologia. Vários algoritmos de roteamento para redes ad hoc foram propostos na literatura, dentre eles destacam-se [10, 12, 14, 18, 27, 35, 37, 41, 42, 52, 54, 55, 64]. Estes algoritmos diferem na forma em que novas rotas são determinadas e como as existentes são modificadas, quando necessário.

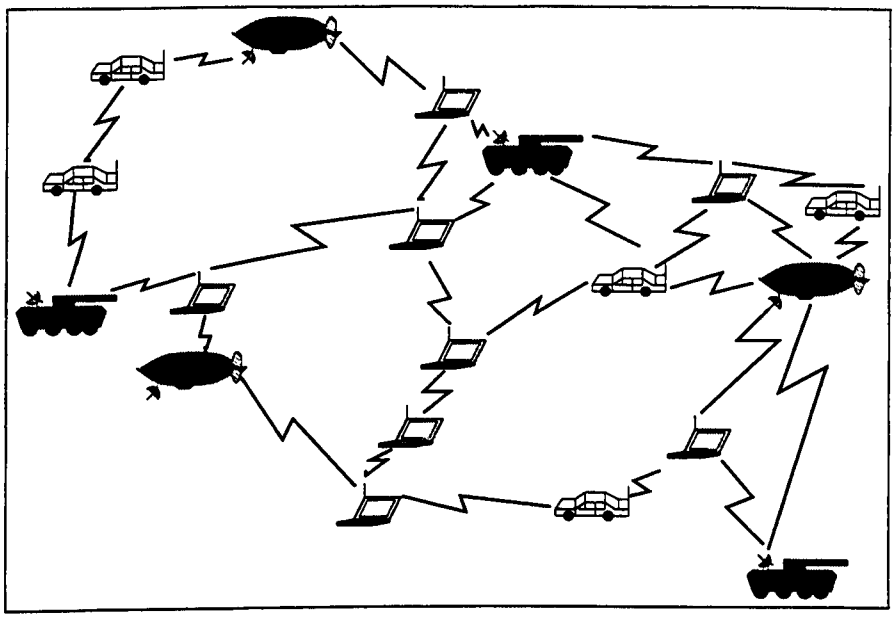


Figura 1.2: Modelo de comunicação em redes móveis ad hoc.

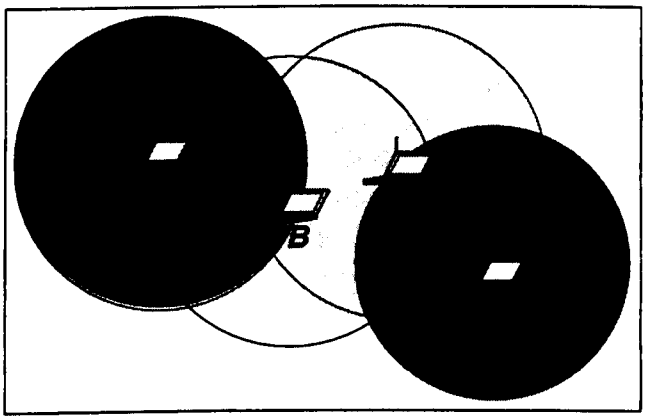


Figura 1.3: Exemplo de uma comunicação entre os computadores de uma MANET. Os círculos demonstram o alcance da comunicação das unidades móveis. Sendo assim, as mensagens de A para D por exemplo, devem passar, pelos nodos B e C para chegar em D.

1.2 Considerações Sobre Redes Ad hoc

As redes ad hoc, também chamadas de *Mobile Packet Radio Networking*, têm vantagens e desvantagens, se comparadas com redes estruturadas e fixas.

Como vantagens podem ser citadas:

- Rápida instalação: MANETs podem ser instaladas rapidamente em locais sem nenhuma infra-estrutura prévia;
- Tolerância a falhas: o mal funcionamento ou o desligamento de uma estação pode ser facilmente sanado com a reconfiguração dinâmica da rede. Em uma rede fixa, ao contrário, quando ocorre uma falha em um roteador, o redirecionamento de tráfego é uma operação complexa, quando possível. Falhas em redes estruturadas são mais

graves ainda se ocorrerem na ESM, pois todos os nodos dependentes desta ESM ficam sem comunicação;

- Conectividade: se duas estações estão dentro da área de alcance das ondas de rádio, elas têm um canal de comunicação entre elas. Em uma rede fixa, mesmo que duas estações estejam uma ao lado da outra é necessário que as estações estejam ligadas por um meio guiado para que troquem informações. Em uma rede estruturada é necessária a comunicação do HM com a ESM e da ESM para o outro HM;
- Mobilidade: em contraposição, à falta de mobilidade dos computadores fixos.

Como desvantagens podem ser citadas:

- Banda passante: canais de comunicação sem fio normalmente possuem menor banda passante que *links* em meios guiados. Em ambientes internos (*indoor*), enquanto a velocidade para redes sem fio tipicamente varia de um a dois Mbps, em redes fixas esse valor já chega a Gbps ou mais;
- Erros no enlace sem fio: a taxa de erros em um *link* sem fio é tipicamente de um bit errado a cada 10^5 ou 10^6 bits transmitidos, enquanto que em uma fibra ótica esta taxa é tipicamente de um a cada 10^{12} ou 10^{15} bits transmitidos [62];
- Localização: existe o problema de se conhecer a localização física do HM e enviar, para esse ponto, suas mensagens. Em redes fixas este problema não ocorre, pois o endereço IP indica implicitamente a localização do nodo. Para redes estruturadas esse problema foi resolvido através do protocolo IP-móvel [56], que se encarrega de localizar o nodo. Já em MANETs, localizar o usuário é um problema, pois não se tem nenhuma informação geográfica e o endereço da máquina não tem necessariamente nenhuma relação com sua posição;
- Roteamento: em uma rede fixa a topologia dificilmente se altera, os nodos ficam normalmente nas mesmas posições da rede. Em uma MANET os nodos podem se mover de um lado para outro de forma não determinística. Se em um momento o nodo A pode se comunicar com um nodo C passando-se por um nodo B, nada garante que em um próximo momento isto continue ocorrendo. Os nodos A, B e C podem ter se movido de forma a ficarem fora da área de alcance uns dos outros. Neste caso é necessário encontrar, de alguma forma, outra rota de A para C.

1.3 Colônia de Formigas

Uma das principais fontes de inspiração para este trabalho é a heurística de colônia de formigas. Esta é uma heurística relativamente recente, e é fruto da tese de doutorado de Marco Dorigo apresentada em 1992 [17]. A heurística de colônia de formigas tem este nome por basear-se no comportamento de um formigueiro.

O formigueiro tem uma organização e uma estrutura muito interessante. Quando uma formiga anda, ela deixa um trilha de um hormônio chamado feromônio. Ao encontrar um

depósito de "comida"[2] a formiga retorna para o formigueiro com a comida, voltando logo em seguida para o depósito a fim de pegar mais comida. O que deve ocorrer é que se o depósito de comida que esta formiga encontrou estiver mais próximo do formigueiro que os outros, este caminho vai se tornar mais forte, pois as formigas que se utilizam deste caminho irão ir e voltar a este deposito mais rapidamente que as formigas que buscam alimento em outros depósitos. Quando outras formigas saírem da colônia elas irão perceber este caminho e tendem a seguí-lo, já que tem um odor mais forte. Mas felizmente nem todas as formigas seguem estas trilhas. Acontece de formigas saírem em busca de novos depósitos de comida, não seguindo os caminhos já estabelecidos. E este é o mecanismo que as formigas utilizam para descobrir novos depósitos de alimento. O trabalho aqui proposto é inspirado por esta heurística, algumas alterações foram necessárias no algoritmo básico de colônia de formigas, visando adaptá-lo a dinamicidade das redes ad hoc.

1.4 Contribuições

Este trabalho apresenta contribuições para a solução do problema de roteamento em redes móveis ad hoc. As principais são a utilização da técnica heurística de colônia de formigas, para melhorar a qualidade do roteamento. A possibilidade da criação de Circuitos Virtuais Flexíveis (CVF), entre os nodos que fazem parte da comunicação em redes ad hoc. A possibilidade de se realizar o roteamento levando em consideração diferentes tipos de dispositivos e também abre um precedente, que é a discussão da possibilidade de se utilizar a rede fixa no roteamento de dados da rede móvel.

Outra contribuição do trabalho, é a forma de manipular a informação de posicionamento recebida da unidade GPS (*Global Positioning System*). Normalmente os algoritmos geográficos utilizam a informação recebida da unidade de GPS como uma forma de controlar o volume de mensagens propagadas por *floodings*(inundação) que congestionam a rede com mensagens. Em alguns casos o *flooding*, é necessário para que determinados algoritmos possam entregar a mensagem, ou encontrar o destino. O GPSAL, o algoritmo proposto neste trabalho, utiliza a informação no roteamento, como forma de melhorar a qualidade das rotas, e não apenas como forma de controlar um *flooding*. Outro ponto importante é a utilização tanto de datagrama, quanto de circuito virtual. Normalmente os algoritmos de roteamento para redes ad hoc utilizam apenas uma das duas formas de transmissão de pacotes, mas sabe-se que as duas são importantes para a transmissão de dados de forma eficiente. O GPSAL mostra que é possível que as duas formas convivam harmoniosamente na rede ad hoc, não sendo necessário optar por apenas uma das duas. O GPSAL também mostra que é possível fazer roteamento, de forma eficiente, em redes ad hoc sem a necessidade de *flooding*. Sendo que poucas são as abordagens de algoritmos para redes ad hoc que não se utilizam de *flooding* em algum momento.

Outra contribuição direta do trabalho é o estudo de alguns dos principais algoritmos existentes. Diversas observações e comparações são feitas entre os algoritmos de roteamento para redes ad hoc. Destas comparações pode-se retirar diversos pontos em comum entre os

[2]As folhas que as formigas carregam para o formigueiro não são propriamente comida. Na verdade elas levam as folhas para alimentar fungos, que crescem dentro dos formigueiros, e se alimentam destes fungos.

algoritmos analisados, e mais que isto, pode-se identificar características presentes em um algoritmo e ausentes em outros. Desta forma este trabalho pode ser utilizado como fonte de pesquisa e inspiração para a criação de algoritmos mais completos e eficientes.

1.5 Estrutura do Texto

No capítulo 2 serão discutidos os principais problemas que os algoritmos de roteamento devem abordar. No capítulo 3 será apresentada uma classificação para algoritmos de roteamento. No capítulo 4 serão discutidos alguns dos principais algoritmos de roteamento existentes. No capítulo 5 serão apresentados alguns dos principais algoritmos de roteamento para redes ad hoc. O capítulo 6 é especialmente dedicado a um estudo sobre o algoritmo LAR (*Location-Aided Routing*), já que este além de ser um algoritmo da mesma classe do GPSAL é também utilizado em comparações no capítulo 8. No capítulo 7 será apresentado o algoritmo GPSAL e suas funcionalidades. No capítulo 8 serão apresentados os resultados dos experimentos realizados com o GPSAL e o LAR. No capítulo 9 será apresentada uma comparação entre as principais características dos algoritmos para redes ad hoc discutidos nesta dissertação. No capítulo 10 serão apresentados alguns pontos que devem ser abordados na continuidade deste trabalho e finalmente no capítulo 11, são apresentadas conclusões e observações sobre a dissertação e o futuro do trabalho e das redes ad hoc.

Capítulo 2

O Problema de Roteamento

Roteamento é a principal função da camada de rede. O roteamento envolve duas atividades básicas: determinação dos caminhos e transporte dos pacotes. No decorrer deste trabalho toda a informação trafegando na rede será denominada de pacote.

2.1 Requisitos para Algoritmos de Roteamento

Serão discutidos aqui os principais requisitos que um algoritmo de roteamento, de uma forma geral, deve satisfazer. Estes requisitos são [62]:

- Funcionamento correto
- Simplicidade
- Robustez
- Imparcialidade
- Estabilidade
- Rápida convergência para a rota ótima
- Flexibilidade
- Aceitação de parâmetros de QoS (*Quality of Service*)
- Independência da tecnologia de rede

Funcionar de forma correta é provavelmente a meta mais genérica a ser observada por um algoritmo de roteamento. Ela se refere a habilidade, que o algoritmo de roteamento deve possuir, de escolher a melhor rota para o pacote. Esta melhor rota pode variar de acordo com a métrica utilizada, como por exemplo menor caminho, maior banda passante, menor atraso.

Simplicidade significa que o algoritmo de roteamento deve oferecer seus serviços com a menor sobrecarga de processamento possível. Funcionamento correto e simplicidade são

pontos muito importantes para algoritmos de roteamento, mas sem dúvida, o principal requisito para um algoritmo de roteamento é a robustez. Espera-se que um algoritmo de roteamento funcione continuamente sem falhas. O algoritmo deve ser capaz de lidar, de forma robusta e consistente, com as mudanças de topologia, falhas de equipamentos, diferentes cargas de tráfego e redes.

Estabilidade e convergência também são fatores importantes para algoritmos de roteamento. Convergência é um ponto crucial principalmente em redes ad hoc. Se os pacotes não estiverem sendo roteados pelo melhor caminho, os recursos das máquinas intermediárias estarão sendo gastos de forma indevida. Entretanto existem algoritmos que não se preocupam em convergir para a melhor rota. Estes algoritmos consideram que encontrar sempre a melhor rota não vale o custo necessário para os nodos se manterem atualizados com relação às alterações da rede, o que é necessário para encontrar a melhor rota. Esta é uma das considerações feitas pelo TORA (*Temporally-Ordered Routing Algorithm*) [51, 52].

Suporte a parâmetros de QoS (Qualidade de Serviço) é um requisito que está cada vez sendo mais exigido em algoritmos de roteamento. Isto se deve, principalmente, ao crescimento do tráfego de informação multimídia nas redes. Agora torna-se importante não só que os pacotes sejam entregues rapidamente, mas também outros fatores devam ser considerados no roteamento como atrasos, perdas de pacotes e disponibilidade dos canais. Estes, e outros parâmetros, devem ser levados em consideração pelo algoritmo de roteamento na escolha do melhor caminho.

Outro ponto que deve ser levado em consideração quando um algoritmo de roteamento é projetado é que este deve ser independente da tecnologia de rede utilizada. Isto é importante porque é desejável que o mesmo algoritmo funcione corretamente na maior variedade de computadores e meios físicos possíveis. O crescimento das redes de computadores tem incentivado o surgimento de novas plataformas e arquiteturas. Por esta razão os algoritmos de roteamento devem ser genéricos, a fim de poderem adaptar-se ao maior número de redes e plataformas.

2.2 Problemas que Devem ser Considerados

Serão apresentadas nesta seção as características do problema de roteamento em MANETs que devem ser observadas para que o roteamento seja feito de forma eficiente. Estas características são principalmente três [28]:

1. Inexistência de uma entidade central;
2. Possibilidade de rápidas alterações topológicas;
3. Todas as comunicações devem ocorrer através do meio sem fio;

A falta de uma entidade centralizadora, com a capacidade de coordenar a rede de forma completa, exige a adoção de sofisticados algoritmos de roteamento. Este fato torna não só a sua forma de operação mais complexa como o seu custo de implantação e implementação mais alto.

Devido à rápida e contínua variação topológica, a possibilidade de comunicação entre dois nodos não tem nenhum tipo de garantia; na verdade ela pode ser intermitente e esporádica. Esta é uma característica que deve receber especial atenção dos algoritmos de roteamento, pois outras rotas devem ser encontradas e validadas no menor espaço de tempo possível.

O terceiro fator, que se refere ao fato da comunicação ocorrer somente por meios não guiados, traz vários problemas relativos à conectividade, à propagação de sinais e à baixa velocidade do canal. Todos estes problemas devem ser tratados e seus impactos negativos minimizados. Além disto, tem-se um problema que é o da alocação do número reduzido de freqüências disponíveis para comunicação.

A possibilidade de rápida movimentação dos nodos e as mudanças bruscas das condições do meio de propagação podem tornar as informações de roteamento rapidamente obsoletas. Logo, uma questão importante para protocolos de roteamento em redes ad hoc é a velocidade com que este protocolo percebe as mudanças da rede. Este tempo é chamado de tempo de convergência. Independentemente do protocolo ser pró-ativo ou reativo (capítulo 3), o tempo de convergência é um fator que deve ser observado na análise do protocolo.

Outro fator que não foi listado, mas é importante e deve ser considerado é o consumo de energia. O suprimento de energia do nodo móvel normalmente é dependente de baterias e estas têm um suprimento limitado de energia. Esta limitação tem influência direta em todos os sistemas envolvidos pela computação móvel, sendo que o roteamento não é exceção. Uma das características mais desejáveis de algoritmos de roteamento em redes móveis, ao lado do tempo de convergência, é a economia de energia. Tanto enviar como receber mensagens implica em um gasto de energia, portanto, deve-se evitar ao máximo possível a realização de comunicações inúteis.

2.3 Qualidades Desejáveis

A IETF (*Internet Engineering Task Force* [23]) tem um grupo de trabalho chamado MANET (*Mobile Ad hoc NETworks* [46]) que é responsável por discutir os problemas referentes às redes ad hoc. Foi publicada pelo grupo, em janeiro de 1999, a RFC 2501 *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations* [13], onde são discutidas as principais características e problemas das redes ad hoc. De acordo com o grupo MANET, as principais qualidades que algoritmos de roteamento para redes ad hoc devem apresentar são:

1. Operar de forma distribuída, sendo este um requisito fundamental para qualquer algoritmo de roteamento em redes ad hoc.

2. Não apresentar *loops* de roteamento. O algoritmo deve ser robusto, com relação a pacotes trafegando na rede por períodos arbitrários de tempo. Este problema pode degradar bastante o desempenho da rede. Segundo o grupo, uma solução que pode ser adotada para minimizar este problema é o uso de uma variável de *Time To*

Live (TTL) no pacote, mas idealmente uma abordagem melhor estruturada é mais indicada.

3. Operar de acordo com a demanda. Os algoritmos precisam ter a capacidade de se adaptar de forma automática a diferentes condições de tráfego. Se isto ocorrer de forma eficiente, tanto os recursos da rede como a carga da bateria podem ser utilizados de forma mais eficiente.

4. Modo de operação pró-ativa. Em alguns casos, a latência gerada para que o protocolo opere de acordo com a demanda pode ser inaceitável. Desta forma, se os recursos de energia e a banda passante permitirem, é desejável que o protocolo trabalhe de forma pró-ativa, que em resumo é tentar descobrir as informações antes que estas se tornem necessárias.

5. Segurança. Sem a ajuda de outros níveis da pilha de protocolos, os protocolos de roteamento para MANETs, em termos de segurança, são vulneráveis a diversas formas de ataques. Entretanto é desejável que existam mecanismos para inibir alterações na forma de funcionamento dos protocolos.

6. Operar de acordo com os períodos de sonolência (*doze mode*) do nodo. É desejável que o protocolo tenha a capacidade de se adaptar, sem maiores conseqüências, aos períodos de inatividade dos nodos móveis, sendo estes períodos avisados previamente ou não.

7. É desejável também que o algoritmo de roteamento tenha suporte a *links* unidirecionais. Tipicamente algoritmos de roteamento para redes ad hoc assumem *links* bidirecionais, onde a comunicação flui nos dois sentidos, e muitas propostas não funcionam sobre *links* unidirecionais. Entretanto, pode ocorrer em redes sem fio, de existirem *links* unidirecionais.

2.4 Análise dos Algoritmos de Roteamento

Segundo o grupo MANET, existem pontos que podem ser observados para avaliar, de forma quantitativa, um protocolo de roteamento para redes ad hoc [13]. São eles:

- Vazão e atraso *fim-a-fim* dos pacotes de dados.
- Tempo para aquisição de uma rota, principalmente para algoritmos que trabalham sob demanda de rotas.
- Porcentagem de pacotes entregues fora de ordem.
- Eficiência do protocolo, que é o número de pacotes de controle necessários para que o protocolo funcione corretamente.

É importante também, no projeto do protocolo de roteamento para MANETs, que sejam levadas em consideração questões como o tamanho da rede, conectividade, capacidade do canal, mobilidade dos nodos e fração da rede que está em *doze mode*. Estes e outros pontos precisam ser observados para que se possa incrementar a escalabilidade do protocolo. Outra característica que é desejável, como em redes fixas, é que o protocolo tenha a capacidade de manipular e escolher a melhor rota baseado em parâmetros de QoS.

O interesse em redes ad hocs vem de duas direções diferentes: por um lado os militares e por outro a comunidade da Internet [28]. Como já era de se esperar, estes dois mundos têm diferentes visões e expectativas para as redes ad hoc. Redes militares têm, relativamente, um pequeno número de nodos, mas com a necessidade de se comunicar com uma estrutura global bem maior [28]. As mensagens são, normalmente, menores que as da comunidade Internet e usadas para manipular e controlar sistemas distribuídos, como por exemplo, equipamentos de comunicação e processamento específico e sistemas de armamento. Possivelmente para as aplicações militares o mais importante é encontrar os nodos eficientemente e no menor espaço de tempo possível.

Para a comunidade Internet, o tempo não é uma restrição crítica. Mais importante que o tempo para se entregar uma mensagem é a eficiência e a economia de energia. Outra capacidade importante, na visão da comunidade Internet, é a implementação de rotas alternativas (*multipath*), pois isto pode aumentar o volume de informações enviadas, recebidas e já que normalmente aplicações Internet necessitam de uma faixa relativamente maior de banda.

Outra diferença entre redes MANET militares e Internet é o número esperado de nodos. Espera-se que as redes da comunidade militar tenham um número sensivelmente menor de nodos que as redes da comunidade Internet. Devido a isto, o gasto de recursos com mensagens de controle nas redes ad hoc da comunidade Internet torna-se um ponto fundamental em contraposição às redes militares, já que estas são compostas de poucos nodos.

Capítulo 3

Classificação dos Algoritmos de Roteamento

Os algoritmos de roteamento podem ser agrupados de acordo com sua forma de funcionamento e características. Este capítulo apresenta uma breve explicação e aponta as principais características dos principais tipos de algoritmos de roteamento. Esta classificação é apresentada para algoritmos de roteamento de uma forma geral, sendo também extensível para algoritmos de roteamento para redes ad hoc. É importante salientar que cada algoritmo pode apresentar mais de uma das características descritas abaixo, já que não são mutuamente exclusivas. Por exemplo, um algoritmo pode ser distribuído e apresentar uma abordagem pró-ativa, ou ser adaptativo e hierárquico, ou até mesmo ter estas quatro características.

Os algoritmos de roteamento podem ser classificados em :

- Estático ou Dinâmico;
- Distribuído ou Centralizado;
- Pró-ativo ou Reativo;
- *Single-Path* ou *Multiple-Path*;
- Plano ou Hierárquico;
- *Host-intelligent* ou *Router-intelligent*;
- Intra-domínio ou Inter-domínio;
- Roteamento Geográfico ou Não;
- *Link state* ou *Distance Vector*;

3.1 Estáticos ou Dinâmicos

Os algoritmos de roteamento podem ser agrupados em duas grandes classes: os algoritmos estáticos e os algoritmos dinâmicos [62]. Os algoritmos estáticos, ou não adaptativos, "aprendem" as rotas na sua inicialização e após isto estas rotas não sofrem alteração, a não ser em caso de falha. Devido à constante alteração topológica das redes ad hoc, a aplicação deste tipo de algoritmo não faz sentido, pois não é possível manter a integridade das rotas escolhidas na inicialização da rede.

Algoritmos adaptativos são os que têm a capacidade de mudar suas decisões de roteamento de acordo com o estado da rede. A forma como as informações sobre o estado da rede são obtidas e quais métricas são utilizadas para alteração das rotas variam de algoritmo para algoritmo. Pode-se perceber facilmente que este tipo de algoritmo é mais indicado para redes ad hoc, devido à sua capacidade de adaptar-se dinamicamente às mudanças de configuração da rede.

3.2 Distribuídos ou Centralizados

Algoritmos de roteamento podem também ser centralizados ou distribuídos. No modelo centralizado, todas as rotas são criadas por um único nodo da rede. Na forma distribuída, cada nodo repassa informações sobre alterações na rede para os outros nodos, sendo que a rota, neste modelo, é calculada de forma individual em cada nodo.

Algoritmos centralizados têm algumas vantagens sobre os distribuídos. Eles são mais simples e os nodos comuns da rede são liberados do cálculo das rotas, o que torna os programas dessas máquinas mais simples e leves. Infelizmente, roteamento centralizado não é confiável. No caso de uma falha no nodo que cria as rotas, toda a rede fica inoperante. É também necessário um consumo extra de banda para a requisição de rotas e a utilização deste tipo de algoritmo depende do tamanho e organização da rede. Outro problema é que se torna necessário uma rota fixa indicando o caminho para o nodo encarregado de criar as rotas. Por estes motivos, o roteamento centralizado torna-se impraticável em redes ad hoc.

Os algoritmos distribuídos, por outro lado, são menos sensíveis a falhas do que os algoritmos centralizados. O tráfego referente a mudanças na rede também não representa um grande problema, já que este é distribuído por toda a rede. Entretanto, *loops* de roteamento são muito mais comuns e difíceis de serem detectados do que em redes centralizadas. *Loops* de roteamento são situações em que um pacote fica sendo enviado sempre entre os mesmos roteadores sem nunca chegar ao destino. Por exemplo, o roteador A envia um pacote para o roteador B, que o devolve para o A e assim sucessivamente. Este é um problema comum em redes que sofrem alterações freqüentes nos estados dos *links*, como é o caso das redes ad hoc.

3.3 Pró-ativos ou Reativos

Protocolos de roteamento podem ser divididos em pró-ativos ou reativos [43]. Os protocolos pró-ativos são os que tentam obter as informações de roteamento antes que estas sejam necessárias. A idéia é avaliar continuamente as rotas. Dessa forma, quando uma das rotas for requisitada ela pode ser utilizada imediatamente. Já os protocolos reativos esperam que alguma rota seja requisitada para somente aí buscar informações e criar a rota. A primeira abordagem requer significativamente mais banda passante que a segunda, mas os pedidos de rota podem ser respondidos imediatamente. Na segunda, tem-se uma economia de banda, mas em contrapartida, o tempo para se responder a uma requisição de rotas é muito maior. Para redes ad hoc, como será mostrado no decorrer deste trabalho, existem algoritmos baseados nas duas abordagens.

3.4 *Single-Path* ou *Multiple-Path*

Outra forma de se agrupar algoritmos de roteamento é com relação à capacidade de suportar mais de uma rota para um mesmo destino. Os algoritmos que têm esta capacidade são chamados *multi-path* e os que só suportam uma rota são chamados de *single-path.*

Algoritmos *multi-path* permitem que o tráfego de mensagens seja distribuído em vários canais. Esta característica pode aumentar significativamente a taxa de utilização da rede. Outro ponto a ser observado é que algoritmos *multi-path* têm uma maior tolerância a falhas, pois no caso de uma rota ficar indisponível, tem-se outra que pode imediatamente tomar o seu lugar. Esta característica é especialmente interessante para ambientes não confiáveis ou em que a topologia se altera freqüentemente, como é o caso de redes ad hoc.

3.5 Algoritmos Planos ou Hierárquicos

Agrupar os algoritmos como planos ou hierárquicos é fazer uma referência à forma como as informações sobre alterações são repassadas entre os nodos da rede. No modelo plano todos os nodos são pares de todos os outros, não existindo nenhum tipo de organização ou estruturação na rede. No sistema hierárquico, por outro lado, a rede é dividida em regiões denominadas domínios, que podem ainda ser divididos em sub-domínios. No modelo hierárquico todos os nodos conhecem tudo sobre a estrutura da sua região, e nada sobre a estrutura interna das outras regiões. Estas regiões são interconectadas por uma estrutura chamada *backbone.* Quando um nodo precisa se comunicar com outro que está fora da sua região, ele envia as mensagens para o seu roteador do *backbone.* Este nodo conhece a estrutura das outras regiões, podendo assim enviar as mensagens para os locais corretos.

Em redes ad hoc a tarefa de organizar a rede de forma hierárquica não é simples. Os nodos alteram sua localização periodicamente e de forma aleatória. Sendo assim, não é possível implementar uma estrutura de *backbone* para dar suporte ao roteamento hierárquico. Mas mesmo assim, a idéia básica do roteamento hierárquico é usada no algoritmo ZRP (*Zone Routing Protocol*) [27], que é uma das mais promissoras propostas de algoritmos de roteamento para redes ad hoc.

3.6 *Host-Intelligent* ou *Router-Intelligent*

Alguns algoritmos de roteamento assumem que a rota deve ser totalmente criada a partir da origem. Este tipo de abordagem geralmente recebe o nome de *source routing*. Nesta abordagem os roteadores atuam como dispositivos de armazenamento e redirecionamento de mensagens, mas não tomam nenhuma decisão quanto à rota das mensagens. Outros algoritmos assumem que os computadores não conhecem nada sobre determinação de rotas, e esta responsabilidade deve ser atribuída aos roteadores. O primeiro tipo de algoritmo onde a escolha da rota é feita no *host*, é conhecido onde *host-intelligent*, e o segundo, como a escolha é feita no roteador, é conhecido como *router-intelligent*.

O compromisso entre algoritmos *host-intelligent* e *router-intelligent* define uma escolha entre um melhor caminho e economia de banda de rede [34]. Algoritmos do tipo *host-intelligent*, freqüentemente escolhem um caminho melhor, mas para isto geram um volume de tráfego adicional. Eles descobrem todos os possíveis caminhos para o destino antes de enviar o pacote. Em redes ad hoc têm-se uma grande preocupação com economia de banda e diminuição do número de pacotes trafegados. Entretanto, como veremos mais à frente, uma das abordagens mais clássicas é um algoritmo *host-intelligent* chamado DSR (*Dynamic Source Routing*) [36, 37]. Nos algoritmos *router-intelligent* as rotas são criadas à medida que o pacote vai caminhando pela rede. Esses algoritmos têm uma economia de banda, no entanto, a rota escolhida pode não ser ótima.

3.7 Intra-domínio ou Inter-domínio

A classificação de algoritmos de roteamento em intra-domínio ou inter-domínio diz respeito ao campo de atuação do algoritmo. Quando se estabelece um sistema hierárquico, com domínios diferenciados, podem existir algoritmos diferentes atuando dentro e entre os domínios.

3.8 Roteamento Geográfico ou Não

Existem algoritmos que utilizam a informação geográfica do nodo como parâmetro para escolher as rotas. Estas informações podem ser conseguidas através de GPS (*Global Positioning System*) [16]. Os algoritmos desta classe são chamados de algoritmos de roteamento geográfico.

3.9 *Linkstate* ou *Distance Vector*

Algoritmos *link state*, também chamados de "caminho mais curto" (*shortest path*), enviam as informações de roteamento para todos os nodos, ou seja, usam *flooding*. A informação enviada diz respeito apenas às informações da sua tabela de roteamento que sofreram algum tipo de alteração. Ao contrário, os algoritmos do tipo *distance vector*, também conhecidos como *Bellman-Ford* [33], enviam toda a sua tabela de roteamento,

porém apenas para seus vizinhos e não para todos os nodos da rede como o *link state*. Novamente existem algoritmos de roteamento em redes ad hoc que exploram tanto uma quanto outra abordagem.

Algoritmos *link state* têm menor propensão a gerar *loops* de roteamento que algoritmos *distance vector*, devido à sua visão mais consistente da rede. Em contrapartida, necessitam de mecanismos mais sofisticados e eficientes para controlar o tráfego que geram. Algoritmos do tipo *link state* são, em geral, computacionalmente mais intensos que algoritmos do tipo *distance vector* requerendo processadores mais potentes, mais memória e sua implementação e manutenção pode ser mais cara e difícil. Como veremos a seguir, as duas abordagens têm suas vantagens e desvantagens.

3.10 Comentários

Neste capítulo foram apresentadas as características mais desejáveis para algoritmos de roteamento, os principais problemas no roteamento de pacotes e a classificação mais comum dos algoritmos. No próximo capítulo serão apresentados alguns dos principais algoritmos de roteamento. Estes algoritmos são também a base ou a fonte de inspiração da maior parte dos algoritmos modernos de roteamento, incluindo os de redes ad hoc.

Capítulo 4

Algoritmos Clássicos de Roteamento

Neste capítulo serão apresentados alguns algoritmos clássicos de roteamento. Todos serão comentados, analisados e para todos os algoritmos serão feitas observações com relação às suas vantagens e desvantagens, quando aplicados a redes ad hoc. Os algoritmos apresentados são:

- Caminho Mais Curto;
- *Flooding*;
- *Distance Vector*;
- *Link State*;
- Roteamento Hierárquico;
- Roteamento *Multicast*;

4.1 Caminho Mais Curto

Devido à sua simplicidade e fácil entendimento, este é um dos algoritmos mais utilizados. O caminho mais curto constrói um grafo da rede onde cada nodo representa um roteador e as arestas são funções que representam a distância entre os roteadores. A função que representa a aresta pode ser baseada em número de *hops*, tráfego, atraso, banda passante, tamanho da fila, distância física, qualidade dos dados recebidos, custo do canal, ou algum outro fator relevante. Achar o caminho ótimo significa encontrar o caminho com o menor custo com relação à função utilizada.

Construído o grafo, qualquer algoritmo que encontre o menor caminho entre dois nodos de um grafo pode ser utilizado. Os algoritmos mais utilizados são o algoritmo de Dijkstra [15] e o de Floyd [22].

4.2 *Flooding*

Flooding ou inundação é um algoritmo estático. Seu modo de operação é muito simples. Cada vez que um nodo recebe um pacote, se ele próprio não for o destino da comunicação, repassa o pacote para todos os canais a que está ligado, menos para o canal por onde recebeu o pacote. Sendo assim, garante-se que se o pacote puder ser entregue ao destino, ele vai ser entregue primeiramente pelo melhor caminho.

O *flooding*, como pode-se facilmente observar, gera um grande número de cópias do pacote original, pois cada nodo multiplica o número de pacotes recebidos. A forma que é normalmente adotada para controlar o número de pacotes na rede, que tende a ser exponencial, e *loops* de roteamento é um valor de *Time-To-Leave* (TTL) que é assinalado ao pacote ao entrar na rede. Toda vez que o pacote passa por um roteador esse valor é decrementado de um. Quando o TTL chega a zero, o pacote é retirado da rede.

Uma possível variante do *flooding* é o *flooding* seletivo. Neste caso, os roteadores não enviam os pacotes para todos os vizinhos, mas apenas para os que estão, aparentemente, na direção do destino. Nesta forma o *flooding* já se torna mais interessante, mas nem sempre é simples determinar a direção do nodo destino em uma rede ad hoc. Esta variante do *flooding* é utilizada no *Location-Aided Routing* (LAR) [41], que será discutido no capítulo 6.

O objetivo do *flooding* é adaptar as rotas às mudanças contínuas da rede. Em protocolos baseados em *link state*, as trocas de tabelas são feitas por *flooding*. Existem vários algoritmos de roteamento para redes ad hoc que utilizam *flooding* para propagar mudanças nas tabelas, encontrar as rotas ou mesmo entregar os pacotes. Entretanto, na maior parte dos casos, são utilizados mecanismos para minimizar seus impactos negativos no desempenho da rede. O *flooding* na sua forma pura, devido ao número exponencial de pacotes gerados, torna-se muito ineficiente. Algumas redes ad hoc, com propósitos militares, no entanto, fazem uso de *flooding* devido à sua rápida convergência e ao número reduzido de nodos que estas redes possuem.

4.3 *Distance Vector*

Muitos algoritmos para MANETs são baseados neste algoritmo. O *Distance Vector* (Vetor de Distâncias) ou Belman-Ford Distribuído (*Distributed Bellman-Ford*) (DBF) [62], como também é chamado, trabalha requisitando periodicamente de cada um dos vizinhos suas tabelas de roteamento. Esta tabela contém todas as distâncias, a partir do *host*, até os outros roteadores da rede.

O *Distance Vector* foi o algoritmo de roteamento original da ARPANET e, ainda hoje, é usado no RIP (*Routing Information Protocol*) [45]. As vantagens do DBF são a simplicidade e eficiência computacional devido à sua característica distribuída.

Cada roteador mantém apenas uma entrada para qualquer roteador da rede, não existem caminhos redundantes. No caso de haver alguma mudança topológica na rede, o DBF tem um tempo alto de convergência e tende a criar *loops* de roteamento. Esta tendência se agrava principalmente em condições onde os *links* não sejam estáveis, como é o caso das redes ad hoc. Uma discussão mais completa sobre o problema de *loops* de roteamento pode

ser encontrada em [4]. Mesmo algumas soluções parciais adotadas em redes fixas, como *split horizon* ou *poisoned reverse*, são insuficientes para resolver o problema de *loops* em redes sem fio, como é discutido em [24].

O DBF funciona bem em teoria, mas na prática, tem um sério problema chamado *count-to-infinity*. Este problema pode ser resumido como "notícias boas andam rápido e notícias ruins demoram a chegar". Isto significa que se uma nova rota melhor for encontrada, esta informação vai se propagar rapidamente pela rede, mas no caso de uma queda de *link*, por exemplo, esta informação vai demorar muito para ser propagada. Em redes ad hoc, o problema de convergência do DBF torna-se crítico, pois os nodos se movem freqüentemente causando mudanças no estado da rede. Devido a este motivo, sua simples aplicação não é a solução mais indicada, mesmo tendo uma economia significativa em termos de banda de rede. Porém existem algoritmos para roteamento para redes ad hoc, como o ABR [64] e o AODV [55] que fazem uso das idéias básicas do DBF.

4.4 *Link State*

O OSPF (*Open Shortest Path First*) [47], algoritmo que substituiu em 1979 o DBF no roteamento da Internet, é um algoritmo do tipo *Link State* (LS). Existem diversas abordagens de roteamento para redes fixas que usam o LS. No entanto o LS na sua forma original, não é muito popular para redes sem fio. Isto porque os caminhos são calculados com base na topologia da rede, e também por ser difícil controlar o fluxo das informações de roteamento disseminadas via *flooding*. Entretanto suas idéias básicas são utilizadas em vários algoritmos de roteamento para MANETs.

A idéia geral do LS é muito simples e pode ser resumida em cinco passos [62]:

1. Descobrimento dos vizinhos e aprendizagem de seus endereços de rede;
2. Medição do atraso ou custo de comunicação para cada um de seus vizinhos;
3. Confecção de um pacote contendo tudo o que já aprendeu;
4. Envio deste pacote para todos os outros roteadores da rede;
5. Cálculo do menor caminho para todos os outros roteadores da rede;

No LS, quando um roteador detecta uma alteração no estado do seu *link* com algum dos vizinhos, ele cria um pacote contendo o novo estado e envia para todos os roteadores da rede. Os outros roteadores, ao receberem o pacote, podem atualizar a sua topologia. Devido à esta disseminação de informações o LS tem uma convergência mais rápida que o DBF. As rotas são calculadas de forma centralizada, sendo portanto fácil prevenir *loops*. Algoritmos do tipo LS têm também a capacidade de suportar mais de uma rota e métrica, pontos importantes para algoritmos de roteamento para redes ad hoc.

Por outro lado, a disseminação de informações é feita através de *flooding*. Como em redes ad hoc o estado dos *links* tem uma grande variação, os constantes *floodings* podem causar uma grande sobrecarga à rede. A sobrecarga causada pelo controle do fluxo de

pacotes faz com que o LS puro seja uma abordagem menos eficiente que o DBF para MANETs, mesmo levando em consideração a sua precisão [8].

4.5 Roteamento Hierárquico

A grande desvantagem dos algoritmos baseados em *Link State*, tanto em redes móveis como em redes fixas, é que o cálculo do menor caminho exige que se tenha um mapa de toda a rede. Esta característica pode exceder a capacidade dos nodos de menor capacidade em grandes redes [44]. Com o aumento das tabelas, não só mais memória é necessária, mas também maior poder de processamento e maior banda de rede.

A principal meta dos algoritmos hierárquicos é diminuir o tamanho das tabelas de rotas. A idéia é dividir a rede em regiões (ou domínios), onde cada roteador conhece tudo sobre a sua região, mas nada sobre a estrutura interna de outras regiões. Esta idéia, como já foi mencionado, é o princípio do *Zone Routing Protocol (ZRP) for Ad Hoc Networks* [27, 29], um dos protocolos mais escalares para MANETs.

Em todas as regiões existe pelo menos um nodo responsável por fazer o roteamento para fora da região. Este nodo é conhecido por todos os outros e se algum nodo precisar enviar pacotes para fora da região, eles são enviados para o nodo responsável pelo roteamento inter-domínios. Podem ser usados protocolos diferentes neste roteamento, já que não é necessário que seja o mesmo usado no roteamento intra-domínio.

Para redes grandes, como a Internet por exemplo, dois níveis hierárquicos podem ser insuficientes. Pode ser necessário agrupar regiões em *clusters*, os *clusters* em zonas, zonas em grupos e só então, atribuir um nome para este agrupamento [62].

O roteamento hierárquico reduz significativamente o tamanho das tabelas de roteamento e a necessidade de poder computacional dos roteadores. Mas existe um custo associado a esta vantagem, que é o aumento do tamanho dos caminhos. Felizmente como Kamoun e Kleinrock apresentam em [38], este é um custo aceitável já que o número ótimo de níveis para uma rede com N nodos é de lnN, exigindo um total de $e\ lnN$ entradas na tabela de roteamento.

Esta abordagem de roteamento, em particular, não pode ser aplicada diretamente às redes ad hoc. Não existe uma forma de separar os nodos por região, já que eles são móveis e não estão necessariamente sempre na mesma região, mas o conceito é muito útil e interessante.

4.6 Roteamento *Multicast*

O objetivo principal dos algoritmos de roteamento *multicast* é entregar pacotes a partir de um nodo da rede para vários outros, mas não todos. Este tipo de operação é utilizada por aplicações que estão separadas, mas trabalham em cooperação. Bons exemplos são algoritmos distribuídos ou clientes de um serviço de rede específico, como por exemplo vídeo sob demanda ou vídeo-conferência.

A distribuição de pacotes pode ser feita através de conexões com todos os destinos. Esta

é uma boa abordagem para um número pequeno de nodos. Quando esse número cresce esta estratégia deixa de ser viável. Outra abordagem também possível é *flooding* mas esta não é a abordagem mais eficiente, já que nem todos os nodos têm interesse na mensagem e, em redes grandes, o *flooding* pode causar problemas de congestionamento.

Uma abordagem comum para roteamento *multicast* é associar o nodo através de um endereço *multicast* a um grupo *multicast*. Este grupo *multicast* é formado pelos nodos que têm interesse em um determinado conjunto de mensagens. Um *host* pode entrar ou sair do grupo de *multicast* a qualquer momento, mas precisa comunicar a operação a um servidor. Assim, os servidores podem informar a qualquer momento os nodos que fazem parte ou não de cada grupo. O nodo deve também ser capaz de gerar pacotes para serem enviados em árvores de distribuição. Árvores de distribuição é a forma como os nodos são organizados logicamente pelo algoritmo. O principal objetivo das árvores de distribuição é ter certeza que existe apenas uma cópia de cada pacote, num determinado tempo, em cada sub-rede. Se existirem vários destinos em um mesmo ramo da árvore de distribuição, é enviado apenas uma cópia do pacote de dados para esse ramo da árvore.

Três são as técnicas atualmente utilizadas para algoritmos de roteamento *multicast*. São elas: *Reverse Path Forwarding* [65], *Multicast OSPF* [48] e *Protocol-Independent Multicast* [21]. Os conceitos de protocolos de roteamento *multicast* são muito interessantes para algoritmos de roteamento em redes móveis, pois pode-se evitar o desperdício de banda com a transmissão de pacotes repetidos. Eventualmente, em *multicast*, podem também ser gerados caminhos de reserva para um mesmo destino. Estes caminhos podem ser usados tanto como caminhos alternativos, caso exista uma perda de conexão pelo caminho original, como para aumentar a vazão de dados entre a origem e o destino, já que se pode enviar dados por mais de um caminho.

4.7 Comentários

Foram vistos neste capítulo alguns algoritmos clássicos de roteamento. Estes algoritmos são a base da maior parte dos algoritmos atuais de roteamento, inclusive os de redes ad hoc. Foram descritos suas vantagens e desvantagens quando aplicados a redes ad hoc. A seguir, no próximo capítulo, serão descritos alguns dos principais algoritmos de roteamento para redes ad hoc.

Capítulo 5

Algoritmos de Roteamento para Redes Ad hoc

Neste capítulo serão apresentados alguns dos principais algoritmos de roteamento para redes ad hoc. São eles:

- *Dynamic Source Routing in Ad Hoc Wireless Networks*
- *Ad Hoc On-Demand Distance Vector Routing*
- *Global State Routing*
- *Zone Routing Protocol*
- *Temporally-Ordered Routing Algorithm*
- *Associativity-Based Routing*

5.1 *Dynamic Source Routing in Ad Hoc Wireless Networks*

O *Dynamic Source Routing* (DSR) [36] é um algoritmo do tipo *source routing*, ou seja o nodo origem determina toda a seqüência dos nodos por onde passará o pacote. O DSR não envia mensagens periódicas para trocar informações de roteamento. Isto reflete em economia de banda de rede e, consequentemente, de energia da bateria. O DSR faz duas considerações básicas: a primeira é que a velocidade dos nodos é moderada se comparada com a velocidade de transmissão. A segunda consideração é que o diâmetro da rede é pequeno, estando entre cinco e dez nodos [37].

5.1.1 Princípios de Operação

Os príncipios de operação dizem respeito ao modo como o algoritmo trabalha, e no caso do DSR são:

- A origem cria a rota completa do pacote que é inserida no cabeçalho da mensagem e envia para o primeiro nodo indicado na lista
- Quando um *Host* Móvel (HM) recebe um pacote, se ele não for o destino da comunicação, ele simplesmente repassa o pacote para o próximo HM indicado no cabeçalho da mensagem.
- Cada HM mantém um cache com as rotas que aprende. Quando um nodo envia um pacote para outro nodo, a origem verifica se possui uma rota para o destino em seu cache. Se a rota existir, a origem usa essa rota para enviar os pacotes. Caso contrário, a origem usa o protocolo de descobrimento de rotas para encontrar uma rota para o destino.
- Cada item do cache de rotas está associado a um tempo de vida. Após a expiração deste período, a informação é retirada do cache.
- Em caso de erro no roteamento, mudança de posição ou desligamento do nodo, o protocolo de descobrimento de rotas é chamado.

5.1.2 Protocolo de Descobrimento de Rotas

Este protocolo permite que qualquer nodo da rede ad hoc encontre dinamicamente uma rota para qualquer outro nodo. O nodo requisita a rota enviando por *broadcast* um pacote de requisição de rotas, conforme mostrado na figura 5.1(a). Cada nodo ao receber este pacote, verifica se tem em seu cache uma rota para o destino requisitado. Se o nodo conhecer uma rota, envia para a origem um pacote de resposta de rota, que contém uma lista com a seqüência de todos os nodos até o destino. Caso o nodo não seja o destino nem tenha um caminho para o destino, ele insere seu endereço no registro de rotas e repassa o pacote também por *broadcast* para seus vizinhos. Este procedimento se repete até que a requisição chegue ao destino, ou até que algum nodo tenha uma rota para ele. Quando isto acontecer é enviado um pacote de resposta de conexão pelo caminho que está no registro de rotas, o que implica *links* bidirecionais (figura 5.1(b)). Para que o DSR funcione também em *links* unidirecionais, é necessário que o nodo verifique em seu cache se possui uma rota para a origem. Se tiver, essa rota é utilizada, caso contrário o mesmo protocolo de descobrimento de rotas é acionado para se encontrar um caminho diferente até o nodo origem. Na figura 5.1(a) é mostrada a requisição de rotas partindo do nodo origem (O) até o nodo destino (D). Na figura 5.1(b) é apresentada a resposta ao pedido de requisição de rotas, que ocorre pelo caminho reverso do primeiro pacote que chegou ao destino.

Um pacote de requisição de rotas contém, além dos endereços de origem e destino, um registro de rotas. Nesse registro de rotas é armazenada a seqüência de nodos por onde a requisição de rotas passou. Cada requisição de rotas tem um identificador único que é indicado pela origem. Cada nodo mantém uma lista contendo o par ⟨Origem da requisição, Identificador da requisição⟩ das requisições recentes. Com isto podem ser detectados *loops* e rotas repetidas. Um nodo ao receber uma requisição de rotas, verifica se o par está na sua lista de requisições recentes, ou se seu endereço está no registro de rotas. Se uma das duas condições for verdadeira o pacote é ignorado.

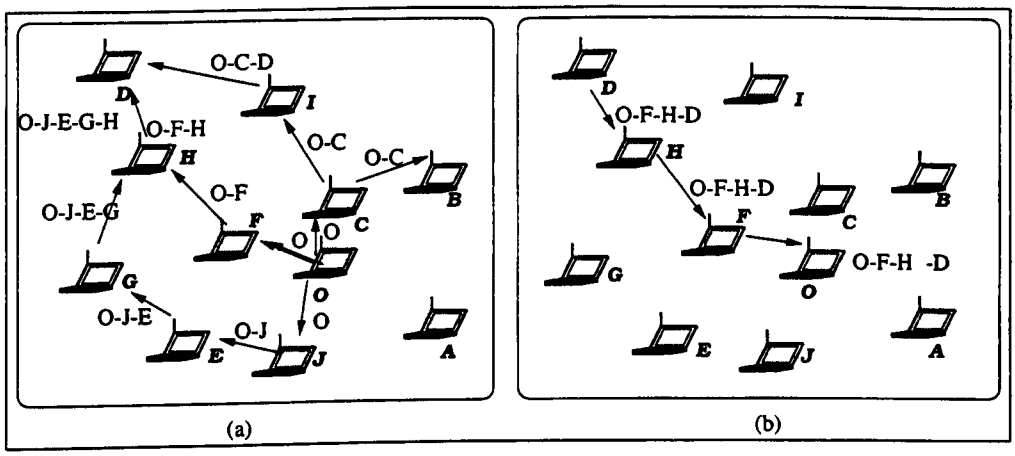


Figura 5.1: O protocolo de descobrimento de rotas do DSR.

5.1.3 Manutenção de Rotas

Manutenção de rotas é o mecanismo através do qual o nodo origem detecta se ocorreram alterações na topologia da rede que poderão comprometer o uso das rotas. Cada nodo pode monitorar os pacotes de confirmação dos outros nodos ou mesmo trabalhar em modo promíscuo, onde ouve todas as comunicações que passam por ele. Desta forma, quando um nodo percebe algum problema na comunicação com seu vizinho, envia um pacote de erro de rota para o nodo origem. Este ao receber o pacote de erro, pode usar qualquer outra rota que tenha em cache, ou mesmo iniciar outra requisição de rotas para encontrar uma nova rota.

5.1.4 Propriedades

A principal vantagem do DSR é ser um algoritmo do tipo *source routing*, não precisando que as informações do nodos sejam atualizadas continuamente, causando assim economia de banda. Infelizmente, devido a isto, o DSR precisa fazer um *flooding* de forma controlada, para descobrir a rota para o destino, o que pode ter um efeito negativo no desempenho da rede. O DSR, na forma atual, não suporta *multicast* de forma real, o que seria desejável. Se for necessário fazer *multicast*, este pode ser feito via *flooding* controlado.

Quando a resposta é enviada em um descobrimento de rotas, os nodos intermediários não aprendem a rota apenas para o nodo destino, mas aprendem também para todos os nodos do registro de rotas.

O DSR apresenta os problemas de escalabilidade típicos de algoritmos do tipo *source routing*. Quando a rede começa a crescer, o controle de pacotes, que se traduz na coleta do endereço de cada nodo por onde o pacote passou, e os pacotes de dados, que contém todas as informações do *source routing*, começam a ficar grandes. Claramente isto pode ter um impacto negativo na rede devido à limitada banda disponível [51].

Um fator que pode influenciar negativamente no DSR é o suporte a *links* unidirecionais. Trabalhar com as interfaces em modo promíscuo é um sério problema de segurança. Nesta forma de operação todas as informações são compartilhadas, mesmo as que não são para o nodo. Assim, algum intruso pode facilmente ter acesso a informações como números de

cartão de crédito ou senhas da rede, se estas não estiverem criptografadas.

5.2 *Ad Hoc On-Demand Distance Vector Routing*

O algoritmo *Ad Hoc On-Demand Distance Vector Routing* (AODV) [55] é baseado em outro algoritmo de roteamento para redes ad hoc: o *Destination-Sequenced Distance-Vector Routing Algorithm* (DSDV) [54]. De forma geral, o AODV tenta eliminar a necessidade de um *broadcast* global das informações de roteamento, o que é o maior problema com o DSDV que limita sua escalabilidade. Outro ponto importante do AODV é tentar minimizar a latência quando novas rotas são requisitadas [2].

O AODV é um algoritmo baseado em vetor de distâncias e também pode ser classificado como um algoritmo reativo, já que uma rota só é criada se for necessária. O AODV procura uma solução intermediária entre o roteamento reativo e o pró-ativo, vistos na seção 3.3. No primeiro a latência é grande, já que é necessário esperar o tempo de resposta da requisição de rotas. No segundo, o volume de informações trocadas torna a abordagem proibitiva para redes ad hoc.

5.2.1 Princípios de Operação

O nodo origem, antes de requisitar uma rota para um destino, consulta sua própria tabela de rotas. Caso não encontre nenhuma informação de rota para o destino, é feita uma requisição de rotas aos nodos vizinhos. Quando as requisições de rotas são propagadas pela rede, todos os nodos atualizam suas tabelas com relação ao nodo origem. Se em um determinado momento o nodo origem não receber uma resposta, ele pode fazer uma nova requisição ou assumir que o destino está indisponível.

Ao receber uma requisição de rotas o nodo verifica se o pedido é para ele. Se for, o nodo envia uma resposta à requisição de rotas por *unicast* pelo caminho reverso e com o mesmo número de seqüência da requisição. Caso a requisição não seja para ele, o nodo verifica se existe em sua tabela de roteamento uma rota para o destino. Se existir, ele responde à requisição de rotas por *unicast* pelo caminho reverso. Esta resposta tem o mesmo número de seqüência da requisição de rotas. Caso não tenha a informação em sua tabela, repassa o pedido aos seus vizinhos.

Cada nodo ao repassar a mensagem aos vizinhos cria uma rota reversa para que a resposta de rota possa voltar até o nodo origem. Quando a requisição de rota chega no destino, este sabe exatamente quantos nodos existem no caminho. O destino gera uma resposta com a seqüência da requisição. Cada nodo que participou do caminho da requisição repassa a resposta para o nodo anterior até que chegue à origem, quando a rota completa é criada. Cada nodo conhece apenas as informações referentes ao seu nodo e não a rota completa, como no DSR. Na figura 5.2(a) é mostrada a requisição de rotas no algoritmo AODV partindo da origem (O) para o destino (D) e na figura 5.2(b) a resposta a este pedido. Sendo que este ocorre pelo primeiro caminho pelo qual a requisição chegou ao destino, só que desta vez os nodos sabem apenas quem são os seus vizinhos.

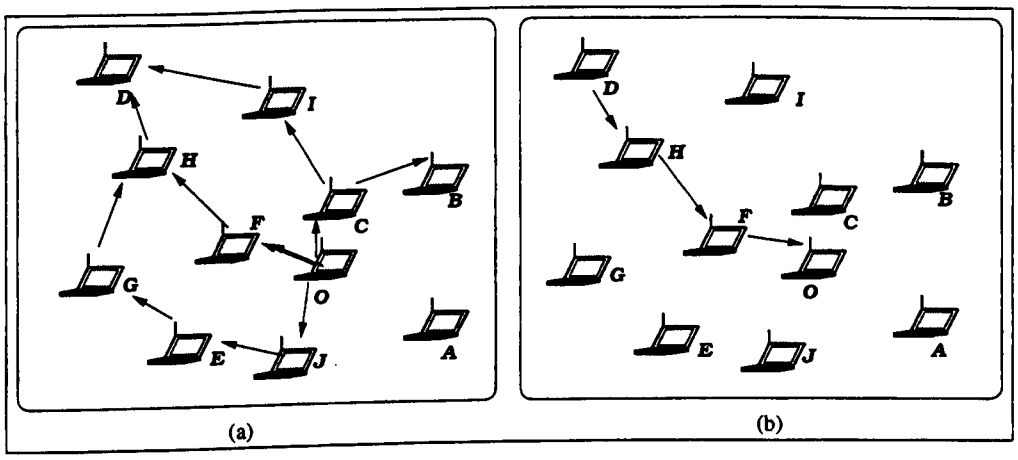


Figura 5.2: Protocolo de descobrimento de rotas do algoritmo AODV.

Para manter as rotas funcionando de forma correta, o AODV exige que cada nodo envie uma mensagem de *Hello* periodicamente aos seus vizinhos. Esta mensagem significa que o nodo continua presente e que as rotas dependentes dele continuam válidas. Se algum nodo parar de enviar mensagens de *Hello*, o vizinho assume que o nodo se moveu e assinala o *link* com o nodo como perdido. Nesse caso o nodo avisa todos os nodos que dependiam desse *link*, através de uma requisição não solicitada de rotas, que o mesmo não está mais disponível. Este aviso é propagado pela rede até a origem, que escolhe se utiliza novamente o protocolo de requisição de rotas ou simplesmente interrompe a transmissão para o destino.

5.2.2 Propriedades

O AODV, quando comparado com algoritmos clássicos de roteamento como vetor de distâncias e *link state*, apresenta uma grande redução no número de mensagens de roteamento na rede. Isto é devido à sua abordagem reativa [43]. A forma de funcionamento do AODV é semelhante a de algoritmos tradicionais, o que pode facilitar no caso de uma possível interconexão da rede ad hoc a uma rede fixa. Mesmo funcionando de forma semelhante aos algoritmos tradicionais, o AODV pode suportar tráfego *multicast* e *unicast*.

O AODV apresenta apenas uma rota para cada destino, o que pode não ser uma boa característica. Felizmente o protocolo pode ser facilmente alterado para garantir o suporte a várias rotas para um mesmo destino.

O desempenho do AODV é tão bom quanto o DSR para todas as taxas de movimentação e velocidades testadas em [6], conseguindo também alcançar a sua meta de eliminar a sobrecarga causada pelos algoritmos *source routing*. No entanto, para isto ele causa uma sobrecarga na rede com pacotes de roteamento, e gerando um custo maior que o DSR [6] para altas taxas de mobilidade.

5.3 *Global State Routing*

O *Global State Routing* (GSR) [8, 9] é um algoritmo de roteamento baseado no *link state*. Desta forma, ele mantém uma visão global da rede e pode fazer cálculos precisos de

rotas, mas ao contrário do LS usa técnicas para limitar o volume de mensagens de controle. Com isto, consegue diminuir o consumo de banda através do controle da sobrecarga que os pacotes de roteamento representam. Entretanto, as características mais interessantes do GSR são a sua capacidade de manipular parâmetros de QoS e as técnicas que utiliza para diminuir o tamanho das tabelas de roteamento.

5.3.1 Princípio de Operação

No GSR, cada nodo possui uma lista de vizinhos, uma tabela de topologia, uma tabela de próximo nodo e uma tabela de distâncias. No estado inicial, a lista de vizinhos e tabela de topologia ficam vazias. Após atribuir os valores de inicialização para as tabelas internas, o nodo deve ouvir o meio para reconhecer seus vizinhos, ou seja, os nodos de quem é capaz de receber uma comunicação.

A tabela de topologias é criada à medida que o nodo vai recebendo as mensagens de roteamento de seus vizinhos. Esta informação é preparada e disseminada periodicamente por todos os nodos. Como o algoritmo é baseado em *link state*, as mensagens de roteamento contém apenas as informações mais recentes com relação ao estado dos *links*.

Uma das diferenças entre o GSR e o LS tradicional é a forma como as informações de roteamento são disseminadas. No LS, sempre que um nodo detecta uma mudança de topologia, esta informação é enviada via *flooding* pela rede. O GSR não faz *flooding* das informações de roteamento, mas mantém a alteração em uma tabela de estado dos *links*, onde também são mantidas as informações de alterações recebidas dos vizinhos. Periodicamente o nodo envia esta tabela somente para seus vizinhos, ao contrário do LS que envia para todos os nodos da rede.

5.3.2 Redução da Atualização de Tabelas

O GSR usa duas técnicas para reduzir o tamanho das mensagens de atualização: *fresh update* e *fisheye*.

No *fresh update* somente informações úteis aos vizinhos são disseminadas. A idéia é que, se a informação sobre um nodo é mais velha que a do meu vizinho, não faz sentido enviar esta informação. Quando as informações obsoletas são eliminadas, a tabela de roteamento pode ser reduzida.

Fresh update ocorre em dois passos:

1. Primeiro passo: cada nodo envia um Vetor de Seqüências de Números (VSN) onde são indicados todos os destinos que estão na tabela. O VSN indica também a idade de cada informação.

2. Segundo passo: cada nodo compara seu VSN com os valores recebidos dos vizinhos. Se alguma entrada é mais antiga, esta entrada é removida das mensagens de atualização. Após o segundo passo o resultado das atualizações de tabelas está pronto para ser disseminado.

A outra abordagem é o *fisheye* (olho de peixe), conforme mostrado na figura 5.3. Esta técnica tem como objetivo reduzir a quantidade de informações para representar dados gráficos [40]. A idéia do *fisheye* é manter um volume alto de informações sobre os nodos mais próximos e diminuir o detalhe das informações dos nodos mais distantes. As informações de roteamento disseminadas, normalmente, contém apenas entradas relacionadas aos nodos mais próximos. As mensagens dos nodos mais distantes também podem ser enviadas, mas com freqüência muito menor. Esta abordagem minimiza o tamanho das mensagens, já que nem todas as informações são repassadas.

Na figura 5.3 tem-se um exemplo do funcionamento do *fisheye* com distância um. Os nodos 1 a 4, que estão ligados diretamente ao nodo A, têm distância um e por isto as atualizações de seus dados são mais freqüentes. Os nodos 5 a 15 têm distância dois a partir do nodo A. Eles são acessíveis para A somente se algum nodo que está diretamente conectado a A servir de ponte para a comunicação. Os nodos 15 a 34 são os que exigem mais de dois nodos intermediários para que a comunicação ocorra.

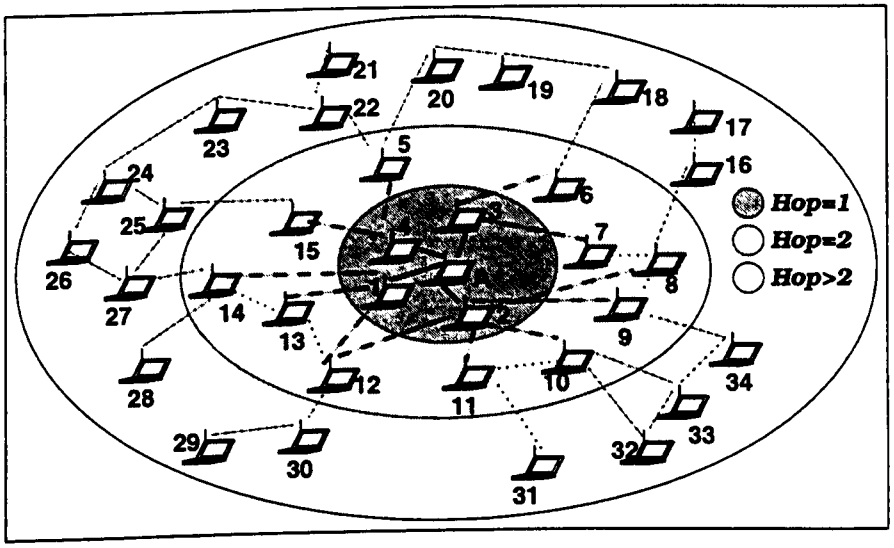


Figura 5.3: Área de atuação do *fisheye* para distância de *hop* um

5.3.3 Propriedades

O GSR consegue de forma eficiente diminuir tanto o volume quanto o tamanho dos pacotes contendo informações de roteamento. O problema é que para uma taxa de mobilidade alta o algoritmo pode causar uma imprecisão nas rotas geradas.

Uma característica do GSR, que não é ainda muito explorada por algoritmos de roteamento para MANETs, é o suporte a QoS [8], mesmo sendo esta uma das mais importantes características para o trafego multimídia, podendo inclusive ser esta uma das mais importantes funções das redes ad hoc no futuro.

5.4 *Zone Routing Protocol*

O *Zone Routing Protocol* (ZRP) [27, 29] é baseado em zonas de roteamento definidas a partir de cada nodo. Para cada *host* é exigido que ele conheça apenas as informações referentes à sua zona de roteamento. Assim as informações de roteamento podem ser propagadas apenas localmente. Para o roteamento dentro da zona do nodo, qualquer algoritmo de roteamento pode ser utilizado e no roteamento inter-zonas é utilizado o *on-demand.*

5.4.1 Princípio de Operação

A zona do nodo é definida como o conjunto de nodos que estão até uma certa distância, em número de nodos, do nodo em questão. O ZRP tem duas características distintas: trabalha de forma reativa no roteamento inter-zonas e de forma pró-ativa no roteamento intra-zona. Protocolos pró-ativos estão continuamente avaliando as alterações da rede. Assim quando um pacote necessita ser enviado, o caminho já é previamente conhecido. Nas abordagens reativas, a rota só é determinada somente quando e se ela for requisitada.

Se o destino da comunicação está dentro da zona referente ao nodo, o pacote pode ser enviado diretamente, já que o nodo sabe o caminho até o destino. Quando o destino é um nodo que está fora da zona, o ZRP pergunta, através de *multicast*, aos nodos da borda da zona se algum deles conhece o destino. Cada nodo da borda verifica se o nodo requisitado está em sua própria zona de roteamento. Se o caminho até o nodo for conhecido, este nodo envia uma resposta afirmativa para o nodo origem. Se o nodo requisitado não estiver em sua zona de roteamento, envia uma requisição para os nodos da sua borda. Isto se repete até que algum nodo responda que sabe como localizar o nodo, ou até que o número de *hops* que a requisição pode caminhar chegue ao máximo.

Cada mensagem de descobrimento de rotas tem um contador de nodos. Este contador tem como objetivo limitar a repercussão de mensagens de descobrimento de rotas pela rede. Cada vez que um nodo repassa a requisição para suas bordas o contador é decrementado. Quando este chegar a zero, a requisição é descartada.

A figura 5.4 mostra o modo de funcionamento da requisição de rotas no ZRP para um nodo externo à zona do nodo. A requisição passa da origem O até encontrar o nodo H que tem contato com o destino D. A requisição vai passando de nodo a nodo até que o nodo H recebe a mensagem e responde, já que conhece o nodo. Neste exemplo o nodo D foi encontrado por mais de um caminho. Como o ZRP não suporta mais de um caminho, apenas um é utilizado pela origem. Outro fato interessante é que o nodo C, que está dentro da área de atuação do nodo O não recebe a requisição de rota. Esta requisição é passada apenas para os nodos da borda por *multicast.* Entretanto o nodo recebe o pedido pois é borda do nodo B.

5.4.2 Propriedades

O ZRP é um algoritmo híbrido entre algoritmos reativos e pró-ativos conseguindo tirar vantagem das duas abordagens. O roteamento pode ocorrer muito rápido dentro da zona

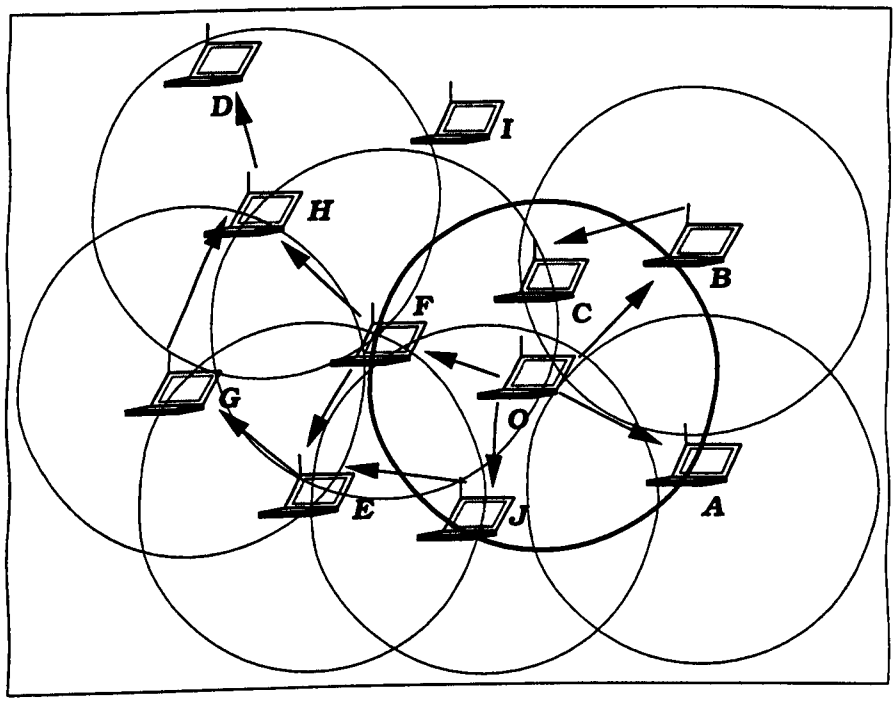


Figura 5.4: Exemplo de uma requisição de rota do nodo O para o nodo D

referente ao nodo, enquanto os nodos fora da zona podem ser encontrados de forma rápida e eficiente. Entretanto o ZRP não especifica qual protocolo de roteamento intra-zona deve ser utilizado. Pode-se portanto ter zonas que usam diferentes protocolos de roteamento para diferentes zonas. Isto não é interessante, pois neste caso os nodos devem ter suporte para diferentes algoritmos de roteamento intra-zona. O ideal é que a rede toda tenha o mesmo algoritmo de roteamento pró-ativo dentro das zonas.

A vantagem do ZRP é a sua escalabilidade. As tabelas de roteamento são pequenas, já que dizem respeito apenas aos nodos dentro da região do nodo. E caso o nodo receba uma informação sobre nodos que estão fora da sua zona, simplesmente descarta esta informação. As informações de roteamento, mesmo que enviadas via *flooding*, ocorrem apenas dentro da região do nodo, ao contrário de *flooding* puro onde as informações seriam enviadas para todos os nodos da rede.

Entretanto o ZRP também herda os problemas relativos a protocolos *on-demand*. Estes problemas são principalmente o atraso decorrente da espera pela resposta da requisição de rotas e o prazo de terminação da requisição. O problema com o contador de nodos por onde a requisição passa é complexo, pois o valor ideal deste varia de acordo com o tamanho da rede. Quanto maior a rede mais nodos são necessários para se alcancar o destino, e quanto menor, o disperdicio de recursos com pacotes inútis é proporcionalmente maior se o número de hops for grande.

5.5 *Temporally-Ordered Routing Algorithm*

O *Temporally-Ordered Routing Algorithm* (TORA) [52, 51] foi projetado para diminuir as reações do protocolo às mudanças topológicas. O objetivo é minimizar o impacto das

mudanças topológicas da rede nas atualizações das tabelas de rotas. Isso é feito restringindo as mensagens relacionadas com o roteamento a um pequeno conjunto de nodos que estejam perto de onde houve a mudança.

Para o TORA o roteamento de forma ótima não é o mais importante. A consideração feita é que manter o caminho ótimo nas tabelas de roteamento não é vantajoso em função da quantidade de mensagens necessárias. O TORA trabalha de forma independente em cada nodo, e cada nodo tem informações referentes apenas a seus vizinhos. O algoritmo pode prover múltiplos caminhos para um mesmo destino e garante roteamento livre de *loops*.

5.5.1 Princípio de Operação

O TORA foi projetado para trabalhar em conjunto com o protocolo IMAP (*Internet MANET Encapsulation Protocol*) [11]. O TORA repassa as seguintes funções ao IMAP:

- Monitorar e manter o estado da conexão
- Controlar a entrega dos pacotes
- Resolução dos endereços de rede
- Autenticação e segurança

O TORA pode ser separado em três funções básicas: criação de rotas, manutenção de rotas e remoção de rotas.

A criação de rotas basicamente requer o estabelecimento de uma seqüência de *links* iniciando na origem e indo até o destino. Para o estabelecimento da rota é construído um grafo acíclico até o destino. Quando um nodo necessita uma rota para um destino particular, transmite um pacote de requisição, que contém o endereço do destino da rota. Este pacote é propagado pela rede até que alcance o destino, ou um nodo intermediário que conheça uma rota para destino. O nodo então envia uma resposta que indica a sua "altura" com relação ao destino. À medida que este pacote vai sendo propagado pela rede, cada nodo que o recebe ajusta a sua altura a um valor maior que o do vizinho de onde a mensagem foi recebida. Isto é feito para se criar uma série de ligações dirigidas a partir da origem da requisição até o nodo que a respondeu.

A manutenção de rotas é utilizada quando um nodo descobre que uma rota até um destino já não é válida (figura 5.5). Quando isto ocorre o nodo ajusta sua altura de modo que ele seja um máximo local com relação aos seus vizinhos e transmite um pacote de resposta de rota. Se o nó não tiver nenhum vizinho que possua altura finita com relação a este destino, então ele mesmo tenta descobrir uma nova rota, da mesma forma como foi descrito acima. A remoção de rotas é usada quando um nodo detecta uma divisão na rede. Ao perceber isto o nodo gera um pacote de desobstrução que restaura o estado das rotas e remove as rotas inválidas da rede.

Na figura 5.5(a) está representada a fase inicial do protocolo de manutenção de rotas. O nodo F ajusta seu valor a um máximo local e envia uma requisição parcial de rotas para os vizinhos. Na figura 5.5(b) tem-se o estado da rede após a execução do protocolo.

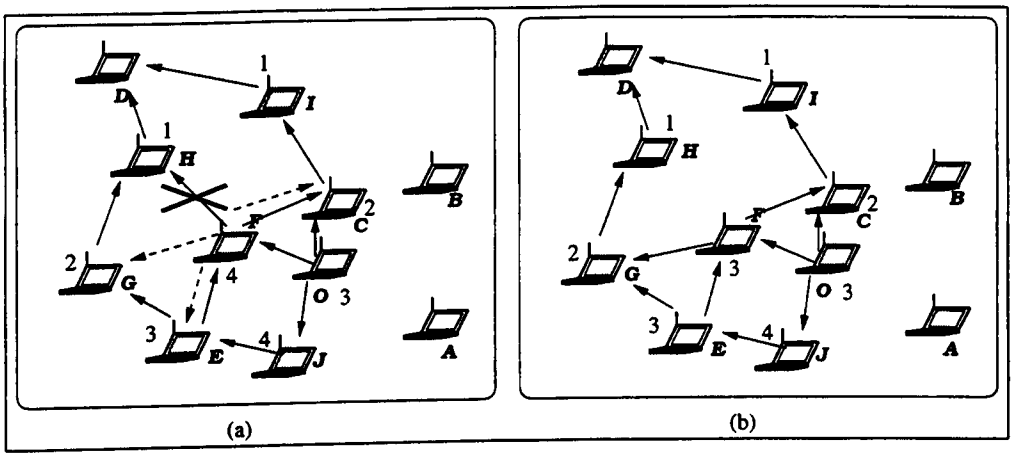


Figura 5.5: O protocolo TORA na fase inicial (a) e após a execução do protocolo (b)

5.5.2 Propriedades

O TORA é um algoritmo robusto. Em [53] pode ser encontrada uma comparação entre o TORA, um algoritmo *Link State* ideal (LSI) e *flooding* puro. Nesse trabalho é mostrado, com base em simulações, que o LSI utiliza mais banda de rede que o TORA com controle da sobrecarga dos pacotes de roteamento pela rede. Quando a taxa de mudanças topológicas aumenta, o controle com a sobrecarga causada pelas mensagens de roteamento pode causar atraso na entrega dos pacotes.

O TORA possui uma grande escalabilidade e por isto, segundo seus autores, é indicado para uso em rede móveis densas. Entretanto, em [6] o TORA teve o pior desempenho entre os algoritmos analisados, em termos de controle dos pacotes gerados. Em redes com 10 ou 20 nodos ele conseguiu entregar 90% dos pacotes gerados. Em redes com mais de 30 nodos foi incapaz de manipular o tráfego gerado pelo protocolo e uma fração significativa de pacotes de dados deixou de ser entregue.

5.6 *Associativity-Based Routing*

O *Associativity-Based Routing* (ABR) [64] tenta manter um compromisso entre *broadcasts* e comunicações ponto-a-ponto. Segundo Royer e Toh [59] o ABR é livre de *loops* de roteamento, *deadloks* e pacotes duplicados e ainda define uma nova métrica para redes ad hoc. Esta métrica é chamada de *degree of association stability*. Isto significa que as rotas devem ser calculadas de acordo com o grau de associação que uns nodos têm com os outros.

Todos os nodos de tempos em tempos geram *beacons* que são ouvidos pelos vizinhos. Cada vez que um vizinho ouvir um *beacon* aumenta o contador de associatividade do nodo que gerou o sinal. Valores altos destes contadores significam nodos com forte associação, já que podem se comunicar continuamente. Valores baixos significam baixa associação, já que poucos *beacons* foram registrados entre os nodos.

5.6.1 Princípio de Operação

O ABR pode ser dividido em três fases:

1. Descobrimento de rotas
2. Reconstrução de rotas
3. Remoção de rotas

O descobrimento de rotas é chamado quando uma rota é requisitada. O nodo faz um *broadcast* da requisição para os nodos vizinhos e espera pela sua resposta (*on-demand*). Todos os nodos que receberem a mensagem, se não forem o destino, antes de reenviarem o pacote também por *broadcast*, incluem no pacote seu endereço e o valor de sua associatividade com seus vizinhos juntamente com as informações de QoS. O nodo sucessor apaga todas as informações de associatividade do nodo anterior, deixando apenas as referentes a ele mesmo e ao caminho percorrido pelo pacote. Desta forma, quando o pacote chegar no destino ele estará apto a escolher a melhor rota, examinando associatividade dos nodos. Quando mais de uma rota tiver o mesmo valor de associatividade, a que tiver o menor número de nodos é escolhida. O nodo destino ao escolher a rota, envia um resposta pelo caminho escolhido. Os nodos desse caminho marcam a rota como válida.

A fase de reconstrução de rotas pode consistir de um descobrimento de rotas parcial, atualização de rotas válidas ou um novo descobrimento de rotas, dependendo da forma como os nodos pertencentes à rota se moveram. Movimentos da origem (figura 5.6(a)) resultam em um processo novo de descobrimento de rota. Entretanto, é enviada uma mensagem para apagar o caminho antigo de forma a liberar recursos. Quando o destino se move (figura 5.6(b)), o nodo imediatamente anterior verifica se o destino é alcançável por algum outro caminho. Caso seja, o nodo destino responde através de uma resposta parcial pelo melhor caminho ao nodo que requisitou o serviço. Se não for possível encontrar o destino, o processo recomeça no nodo imediatamente anterior. Se este processo resultar em uma volta completa à origem, novamente o processo de descobrimento de rotas é chamado. Quando uma requisição de rotas não é mais necessária, o nodo origem realiza uma remoção de rotas, que é enviada por *broadcast* para todos os nodos da rota. Desta forma os nodos podem atualizar suas tabelas retirando a rota.

5.6.2 Propriedades

O ABR apresenta alguns novos conceitos interessantes para redes ad hoc, como é o caso da associatividade e da recuperação de rotas. Outro ponto interessante desta abordagem é a capacidade de lidar com parâmetros de QoS. Por outro lado, devido a ser um protocolo do tipo *on-demand*, temos o problema do atraso causado pelo descobrimento da rota. No caso do ABR este tempo é especialmente delicado, já que o destino tenta escolher a melhor rota entre as que chegarem. Sendo assim ele deve esperar que mais de uma rota chegue para responder, ao contrário do DSR por exemplo, que responde apenas à primeira rota que chega, já que assume que é a melhor.

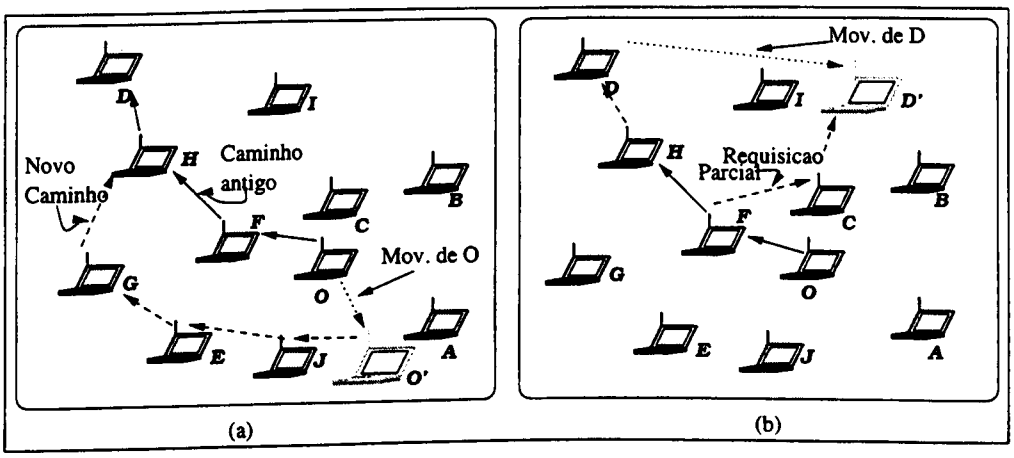


Figura 5.6: A reação do protocolo de reconstrução de rotas no ABR. Na figura (a) a origem se move de O para O′ e na figura (b) o destino se move de D para D′

5.7 Comentários

Foram vistos neste capítulo alguns dos principais algoritmos de roteamento para redes ad hoc. Foram discutidos aqui algoritmos do tipo *on-demand* como é o caso do ABR e do TORA em comparação com uma abordagem híbrida como o ZRP. Foram discutidos algoritmos como o DSR e o ABR que são do tipo *link state*, o TORA que é do tipo vetor de distâncias e algoritmos como o ABR e o GSR que podem suportar parâmetros de QoS, e o AODV que tem suporte a tráfego *multicast*.

Capítulo 6

Location-Aided Routing - LAR

Neste capítulo será apresentado um estudo sobre o algoritmo *Location-Aided Routing* (LAR). Esta atenção especial se deve ao fato de que este algoritmo é utilizado como base para diversas comparações de desempenho com o GPSAL. O LAR foi escolhido pois se enquadra na mesma categoria que o GPSAL, já que também é um algoritmo geográfico, projetado especialmente para redes ad hoc.

6.1 *Location-Aided Routing*

O *Location-Aided Routing*(LAR) [41] usa informações de localização obtidas através de GPS (*Global Positioning System*) para descobrir uma rota para um dado destino, ou seja, executa um roteamento geográfico. Usando essa informação é possível restringir a área a ser pesquisada ao se determinar uma nova rota. Essa área é definida em função da provável localização do nodo destino no momento de descobrir a rota.

6.2 Princípio de Operação

No LAR a requisição de informações sobre a localização do nodo destino é feita através de um *flooding* puro pela rede. O destino ao receber esta requisição envia de volta ao nodo origem a sua informação de localização. Em [41] é proposto duas variantes do LAR. A primeira chamada de LAR1 e uma segunda chamada de LAR2.

6.2.1 LAR1

O LAR1 define uma zona de requisição de formato retangular que pode conter o nodo destino. Esta zona de requisição deve ter um tamanho tal que inclua o nodo origem e toda a zona onde se espera que esteja o nodo destino , sendo que o tamanho dessa área é um círculo proporcional à velocidade de movimento do nodo destino e ao tempo decorrido desde o registro da última localização do destino. Cada nodo ao receber um pacote verifica se está dentro da zona de requisição. Se estiver, reenvia novamente a mensagem para seus

vizinhos, caso contrário ignora a mensagem. O pacote é repassado de nodo a nodo até que chegue ao destino ou até que seu TTL expire.

A figura 6.1 mostra o funcionamento do LAR1 e a forma como as mensagens são entregues. Os nodos que estão dentro da zona de requisição fazem *flooding* dos pacotes de dados, até que o TTL dos pacotes expire. Os nodos de fora da região simplesmente ignoram os pacotes. No programa 6.1 é apresentado, de forma algorítmica, o funcionamento interno do nodo no LAR1.

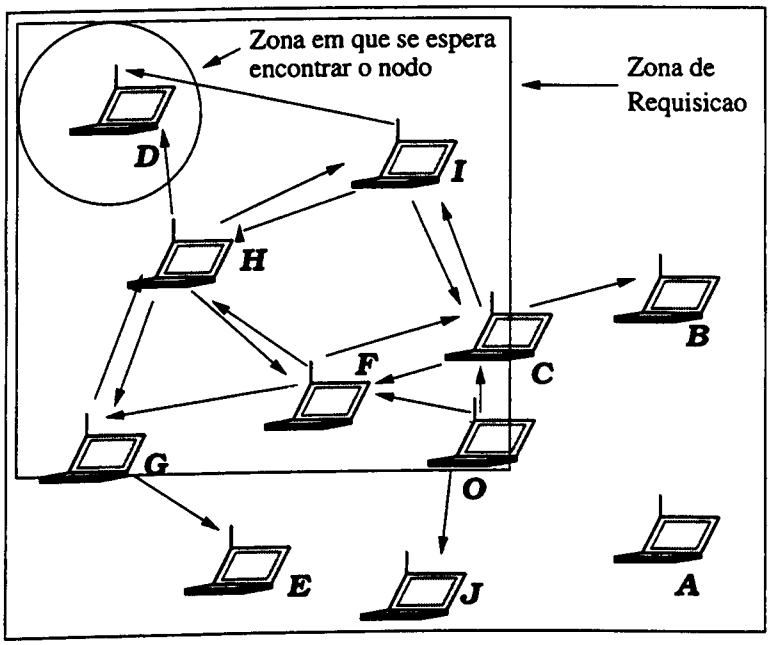


Figura 6.1: A forma de funcionamento do LAR1 na entrega de pacotes do nodo O para o nodo D

```
Início
   Espera ter mensagem no buffer;
   Testa se mensagem é para envio /* a origem do pacote */
   Se Sim
      Verifica se tem a posição do destino na tabela de roteamento
      Se Não
         Enquanto <> de Timeout ou <> de recebeu resposta faça
            Faz Flooding puro atrás da informação;
            Espera até resposta do Flooding ou Timeout;
            Se recebeu pacote de resposta
               Atualiza informações na tabela de roteamento;
         Fim_enquanto;
      Fim_Se; /* se tem a posição do destino na tabela */
      Calcula a Zona em que se Espera Encontrar o Nodo;
```

```
      Calcula a Zona de Requisição;
      Empacota Dados;
      Insere o pacote na tabela de pacotes recentemente recebidos;
      Envia Pacote;

   Se Não /* Mensagem a ser recebida */
      Verifica se Nodo está na Zona de Requisição
      Se Não
         Ignora pacote;
      Se Sim
         Testa se o pacote está na tabela de pacotes recentemente recebidos;
         Se Sim
            Ignora pacote;
         Se Não
            Testa se é o destino
            Se Sim
               Atualiza a tabela com a posição da origem;
               Testa se é Pacote de Requisição de Rotas
               Se Não
                  Armazena Dados;
               Se Sim /* é uma Requisição de Rotas */
                  Calcula a Zona em que se Espera Encontrar o Nodo;
                  Calcula a Zona de Requisição;
                  Empacota Resposta de Requisição;
                  Insere o pacote na tabela de pacotes recentemente recebidos;
                  Envia Pacote;
               Fim_Se; /* se é uma Requisição de Rotas */
            Se_Não /* Não é o destino ->  é um nodo intermediário */
               Insere o pacote na tabela de recentemente recebidos;
               Reenvia Pacote;
            Fim_Se; /* se é o destino */
         Fim_Se; /* se o pacote está na tabela de recentemente recebidos */
      Fim_Se; /* se nodo está na Zona de Requisição */
   Fim_Se; /* se mensagem é para envio */

   Finaliza o processamento do pacote;
Fim.
```

Programa 6.1: Funcionamento interno do nodo no algoritmo LAR1

6.2.2 LAR2

Na segunda variante, o LAR2, (Programa 6.2) a mensagem contém duas informações que são utilizadas na escolha de uma zona: as coordenadas (x, y) do nodo destino e uma estimativa d de quão longe esse nodo pode estar dessas coordenadas. Essas informações são usadas pelos nodos intermediários para determinar a zona mais provável de se encontrar o destino. Cada nodo ao receber uma mensagem, nesta abordagem, verifica se está mais próximo do destino que o nodo de onde recebeu a mensagem. Caso esteja, faz um *broadcast* deste pacote para seus vizinhos, caso contrário simplesmente ignora o pacote.

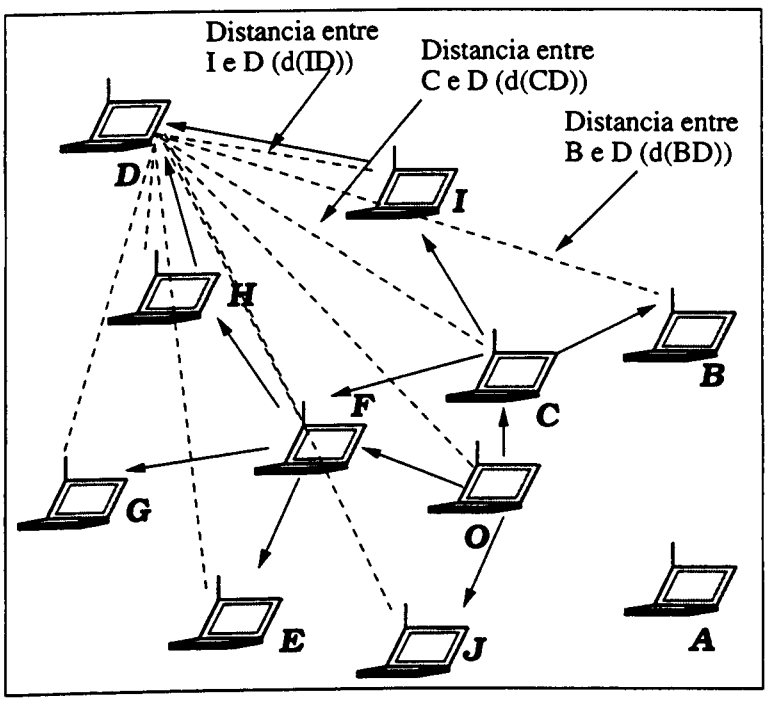


Figura 6.2: A forma de funcionamento do LAR2 na entrega de pacotes do nodo O para o nodo D.

A figura 6.2 mostra como uma comunicação se procede no LAR2. Os nodos só repassam o pacote caso estejam mais próximos do destino que o nodo que lhe enviou o pacote. No caso da comunicação entre O e D pelo caminho ⟨O, C, I, D⟩ tanto o nodo C quanto o nodo I só repassam a comunicação porque estão mais próximos do destino que o nodo anterior. A distância entre C e D (d(CD)) é menor que a distância entre O e D, e a distância entre I e D (d(ID)) por sua vez é menor que d(CD). Isso não ocorre com o nodo B, por exemplo, onde a distância entre B e D (d(BD)) é maior que d(CD). Deste modo o nodo B não repassa a informação.

```
Início
   Espera ter mensagem no buffer;
   Testa se mensagem é para envio /* é a origem do pacote */
   Se Sim
      Verifica se tem a posição do destino na tabela
```

```
   Se Não
      Enquanto <> de Timeout ou <> de recebeu resposta faça
         Faz Flooding puro atrás da informação;
         Espera até resposta do Flooding ou Timeout;
         Se recebeu pacote de resposta
            Atualiza informações na tabela;
      Fim_enquanto;
   Fim_Se;/* se tem a posição do destino na tabela */
   Calcula a Zona em que se Espera Encontrar o Nodo; /* margem de
                                                       erro*/
   Empacota Dados;
   Insere o pacote na tabela de recentemente recebidos;
   Envia Pacote;

Se Não /* Mensagem a ser recebida */
   Testa se o pacote está na tabela de recentemente recebidos
   Se Sim
      Ignora pacote;
   Se Não
      Cadastra pacote na tabela de recentemente recebidos;
      Verifica se é o destino da mensagem
      Se Não
         Verifica se é flooding puro
            Se Sim
               Faz Broadcast do Pacote;
            Se Não
               Verifica se sua distância para o destino
               é menor do que a do nodo anterior
               Se Sim
                  Substitui no pacote a distância do
                  nodo anterior pela sua;
                  Faz Broadcast desse novo Pacote;
               Se Não
                  Ignora pacote;
               Fim_se ;/* Se sua distância para o destino é menor */
            Fim_se; /* Se é flooding puro */
      Se Sim
         Verifica se o Pacote é de dados
         Se Sim
```

```
            Processa Dados;
          Se Não /*Não é o destino ->  é um nodo intermediário*/
            Cadastra endereço do destino;
          Fim_Se; /*se é o pacote de dados */
        Fim_Se; /* Se é o destino da mensagem */
      Fim_Se; /* Se o pacote está na tabela dos recentemente recebidos */
   Fim_Se; /* se mensagem é para envio */

   Finaliza processamento do pacote;
Fim.
```

Programa 6.2: Funcionamento interno do nodo no algoritmo LAR2

6.3 Propriedades

O LAR traz uma nova e importante característica que é o roteamento baseado nas informações geográficas do nodo. O maior problema com o LAR é a não exigência de que o nodo deva ter uma placa GPS, já que é um periférico relativamente barato [63], nem mesmo a falta de precisão que estes equipamentos apresentam [31], mas sim o *flooding* inicial. Para descobrir uma rota o LAR precisa de um *flooding* puro pela rede. Para redes grandes o volume de tráfego tende a crescer muito, não sendo portanto escalável.

6.4 Falhas do Algoritmo

O principal problema do LAR, sem dúvida, é a quantidade de pacotes gerados pelo algoritmo. Mas uma outra característica interessante do LAR é que mesmo sendo feito *flooding* ele não garante a entrega do pacote. Existem alguns casos em que, mesmo existindo uma rota para o destino, o algoritmo não consegue entregar o pacote. Os casos que foram identificados, e que serão discutidos a seguir são:

- Quando a origem e o destino estão na mesma linha;
- Concavidade da rede;
- Quando os nodos estão se movendo em direções opostas;
- Quando o tamanho do TTL é inconsistente com a dimensão da rede;

6.4.1 Origem e o destino estão na mesma linha

No LAR1 pode ocorrer de a origem e o destino da comunicação estarem na mesma linha. Neste caso a Zona de Requisição se torna estreita, como se pode ver na figura 6.3.

Desta forma, mesmo fazendo *flooding*, pode ocorrer de existirem poucos, ou nenhum, nodo na Zona de Requisição, impedindo assim a entrega do pacote (não por falta de rota e sim devido à forma que o LAR1 controla o seu *flooding*). No exemplo da figura 6.3, os nodos A, B, C e F recebem a mensagem da origem O, para o destino D, mas por estarem fora da Zona de Requisição ignoram a mensagem.

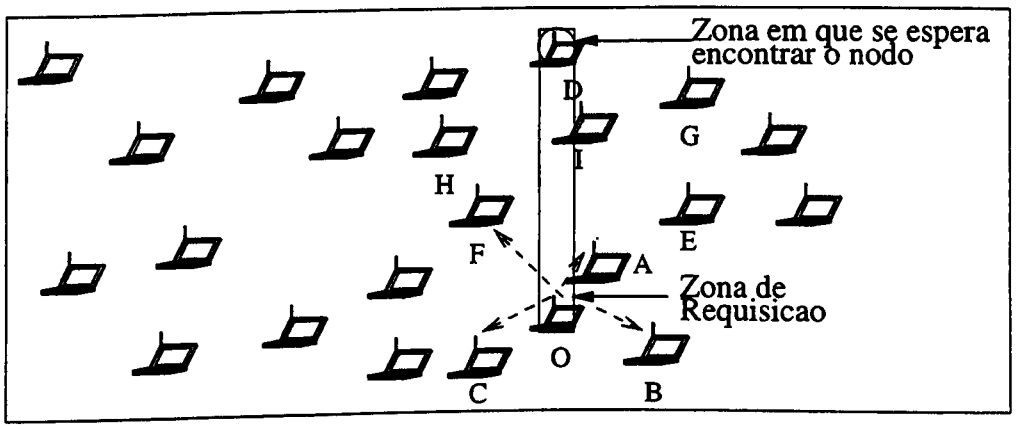


Figura 6.3: Caso os nodos estejam alinhados o LAR1 pode falhar por falta de nodos na Zona de Requisição.

6.4.2 Concavidade da rede

Este problema é típico tanto no LAR1 quanto no LAR2. Em algoritmos do tipo LAR2, o nodo que recebe o pacote sempre tenta ver se está mais próximo do destino que o anterior. Através desta observação, o algoritmo acredita que quanto mais próximo do destino melhor. Se o nodo está mais longe do destino não representa um avanço, pois o pacote não está se aproximando do destino, e sim é um retrocesso, pois o pacote está se afastando do destino e portanto deve ser descartado [61]. À primeira vista parece uma observação pertinente, mas podem haver casos de concavidade na rede. Uma concavidade na rede ocorre quando existe uma rota entre dois nodos, mas esta não passa pelos nodos de menor distância até o destino. Em algum momento é necessário que haja um retrocesso, como pode ser visto na figura 6.4. O nodo C recebe a mensagem de B e repassa para E e para F. O nodo F, mesmo sendo um caminho possível, ignora a mensagem já que está mais longe do destino que C, e E não está ao alcance de nenhum outro nodo para onde possa enviar a mensagem. Deste modo, mesmo havendo um caminho possível, que seria a rota O, A, B, C, F, G, H, D a mensagem não é entregue ao destino.

O problema de concavidade no LAR1 é um pouco diferente, mas da mesma forma pode acontecer de existir um caminho possível e a mensagem não ser entregue. Como pode ser observado na figura 6.5, pode ocorrer que, devido a mobilidade dos nodos, algum nodo responda o *flooding* inicial e saia da Zona de Requisição. No caso da figura 6.5 isto ocorre com o nodo B, que deveria enviar a mensagem para o nodo C. Mas mesmo não se utilizando diretamente o nodo C, existem outras rotas que poderiam ser utilizadas, tanto pelos nodos B e A mas isso não ocorre devido à forma de funcionamento do protocolo.

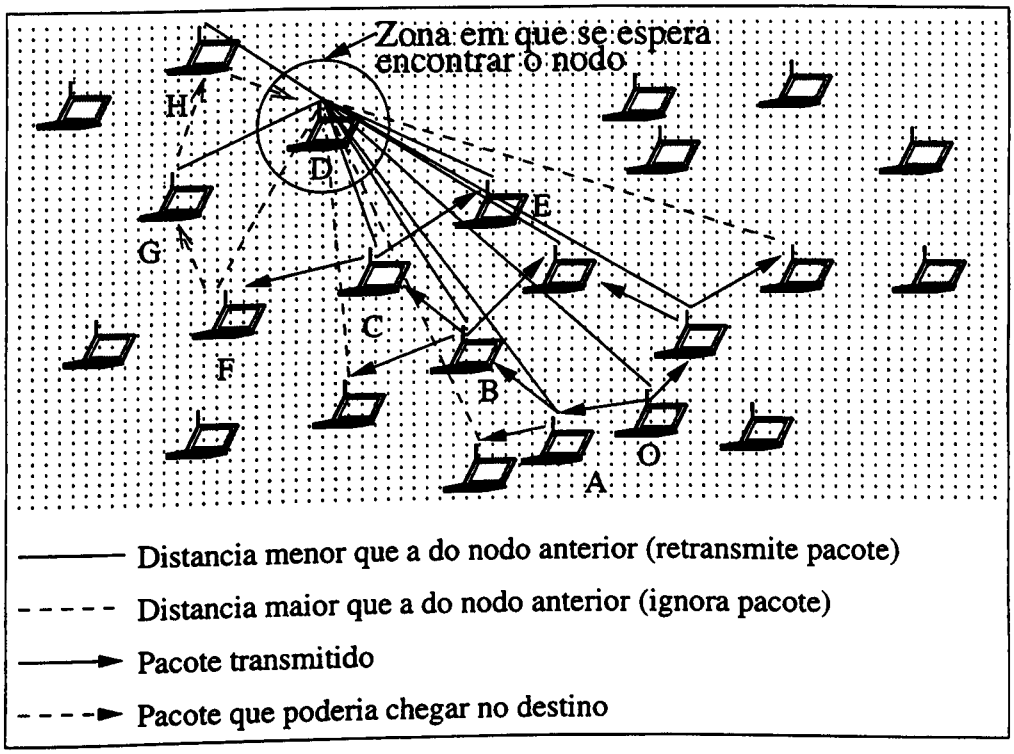


Figura 6.4: O problema de concavidade na rede, onde o LAR2 pode deixar de entregar mensagens.

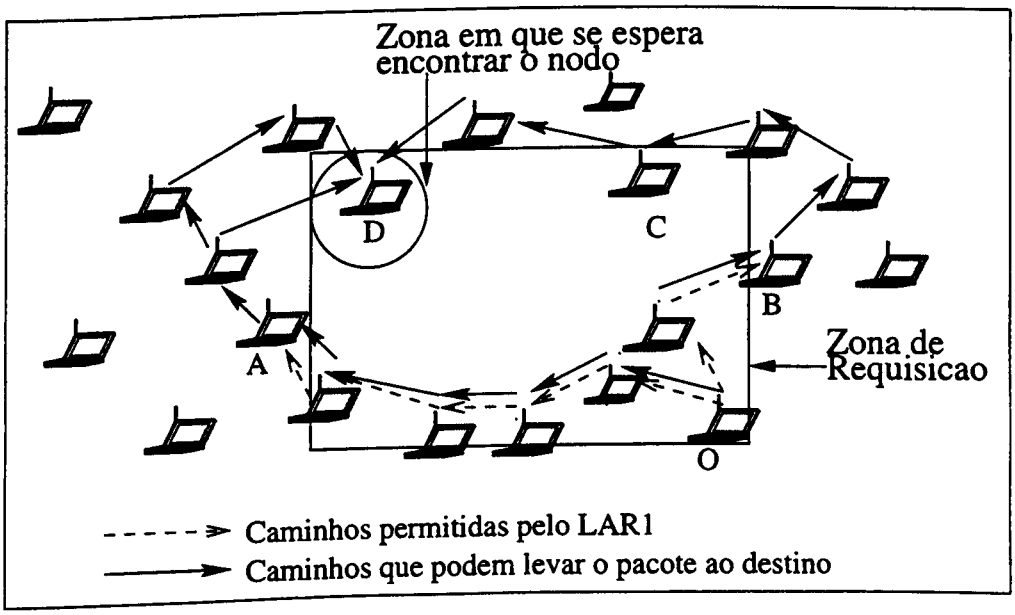


Figura 6.5: O problema de concavidade na rede, onde o LAR1 pode deixar de entregar mensagens.

6.4.3 Nodos se movendo em direções opostas

No LAR1, caso os nodos origem e destino estejam se afastando em direções opostas o algoritmo não consegue enviar o pacote. Neste caso, quando a origem envia uma mensagem para o destino, que é recebida, e este retorna a resposta, pode ocorrer que o nodo origem

esteja fora da região de requisição de rotas da mensagem de resposta. Este problema pode ser solucionado aumentando-se a Zona de Requisição dos pacotes e considerando-se a movimentação da origem, não só do destino.

6.4.4 *Time to Leave* de tamanho inconsistente

Em algoritmos que usam *flooding*, a forma mais comum de retirar as mensagens da rede é o TTL, que limita o número de vezes que uma mensagem será retransmitida.O problema de dimensionar o TTL no LAR é o mesmo enfrentado por qualquer algoritmo que use *flooding*. Caso o TTL seja maior que o necessário, há um desperdício desnecessário de recursos e, caso contrário, se o TTL for pequeno, podem haver situações em que a rota existe, só que devido ao TTL a mensagem é descartada antes de chegar ao destino. No LAR1 o nodo pode, devido à alguma configuração momentânea da rede, receber o pacote de resposta à requisição de rotas, calcular corretamente a área de requisição, enviar a mensagem, mas esta não chegar ao destino porque o número de nodos necessário é maior que o permitido pelo TTL e todos os pacotes serão descartados. Este problema pode ocorrer de forma similar tanto no LAR1 quanto no LAR2 também.

6.5 Comentários

Foi discutido neste capítulo o algoritmo *Location-Aided Routing*, que será posteriormente utilizado para comparações de desempenho com o GPSAL. Foram discutidos também o modo de funcionamento do algoritmo, suas características principais e suas principais condições de falhas.

Capítulo 7

O Algoritmo GPSAL

O algoritmo GPSAL (*GPS Ant Like Routing Algorithm*) proposto neste trabalho é baseado também em informações de localização dos nodos para fazer o roteamento. Estas informações podem ser adquiridas, a qualquer momento, através de um dispositivo de GPS. Outra característica marcante do protocolo GPSAL é a utilização de agentes modelados como formiga para coletar e disseminar as informações sobre localização dos nodos. Isso permite que os nodos da rede possuam informações mais precisas sobre os outros nodos.

7.1 O Algoritmo GPSAL

Nesta seção será descrita a forma básica de funcionamento e as principais características do GPSAL.

7.1.1 Algumas Considerações

Como considerações iniciais, necessárias ao funcionamento do algoritmo, devemos observar que cada computador móvel deve ser dotado de uma unidade de GPS capaz de fornecer a posição geográfica do nodo sempre que solicitado. O GPS fornece informações tridimensionais [31], mas por questão de simplicidade consideraremos somente as informações de latitude e longitude (X, Y), ignorando a altitude do nodo (Z). Esta simplificação tem o objetivo de simplificar a explicação e o entendimento do algoritmo, mas não há perda de generalidade, pois a consideração da altitude tende a melhorar ainda mais a acurácia do GPSAL.

A comunicação entre dois nodos vizinhos é feita pelo protocolo da camada MAC (*Medium Access Control*), que garante a entrega e a integridade dos pacotes entre os nodos intermediários.

A tabela de rotas contém o endereço de cada nodo, a posição (X,Y), a posição anterior conhecida, velocidade, o *timestamp* da informação, *timestamp* da posição anterior, se o computador é móvel ou fixo e se está apto a realizar o roteamento de mensagens ou se pode ser apenas origem ou destino da comunicação. A cada troca de tabelas, todas estas informações são trocadas entre os nodos.

Existem diferentes tipos de equipamentos que podem estar ligados a rede ad hoc sendo que estes equipamentos podem ter capacidades distintas. O GPSAL considera duas classes: o nodo pode ser um nodo apto a fazer o roteamento ou não. Os nodos que não podem fazer roteamento só podem estar nos extremos da comunicação nunca sendo um dos nodos intermediários. Sempre que se fizer referência a busca de uma rota se faz de acordo com as informações da tabela de roteamento, sendo que este parâmetro é levado em consideração. O nodo pode também mudar seu status caso a sua quantidade de energia diminua a níveis críticos.

O algoritmo de roteamento é independente dos níveis inferiores. O endereçamento pode ser tanto IP quanto IPX ou outro qualquer. A única restrição quanto à forma como é feito o endereçamento, é que cada nodo da rede deve possuir um identificador único. Nos cabeçalhos dos pacotes, apenas para efeitos de ilustração, iremos considerar os endereços como sendo de 32 bits, o mesmo tamanho do endereço no protocolo IPv4.

Cada vez que o nodo repassa informações a seu respeito, junto com estas informações é colocado um *timestamp*. Isto garante, aos outros nodos, que é sempre possível identificar se uma informação, sobre um determinado nodo, é mais nova ou mais antiga que outra.

Normalmente a velocidade da comunicação via rádio, e de processamento das mensagens, é superior à velocidade de movimentação dos hosts móveis.

7.1.2 Funcionamento Básico

Caso um computador móvel deseje enviar uma informação para outro nodo, ele verifica em sua tabela a última posição conhecida do destino. Se esta informação estiver na tabela, determina o melhor caminho baseado nas informações de posicionamento e de nodos roteadores que ele tem. O caminho é colocado na mensagem juntamente com as informações referentes à posição dos nodos intermediários e seus respectivos *timestamps*. Em seguida a mensagem é enviada para o primeiro nodo da lista. Se a tabela de rotas não tiver informações sobre o nodo destino, o computador móvel envia uma mensagem para o computador fixo mais próximo. Este, caso exista, fica encarregado de encontrar o destino, caso contrário o destino é considerado não alcançável.

As informações sobre os nodos são um retrato de um tempo anterior e podem não retratar a situação atual. Devido a isto cada nodo deve atualizar a sua tabela de acordo com as informações contidas nos pacotes, se estas forem mais atuais que as de sua tabela de roteamento. Cada nodo, ao receber uma mensagem deve sempre atualizar nesta as informações referentes a ele mesmo. Desta forma garante-se que os nodos posteriores terão informações recentes sobre os nodos por onde o pacote já passou. Feito isto, o nodo verifica se é capaz de encontrar alguma rota melhor que a atual e, se conseguir, faz o redirecionamento do pacote para esta nova rota. O nodo intermediário pode refazer a rota a partir dele, mas não deve alterar o caminho por onde o pacote já passou. Este processo é repetido em todos os nodos até que o pacote alcance o destino.

A figura 7.1 mostra um cenário onde pode ser desejável que haja um redirecionamento do fluxo de dados. O computador A deseja enviar uma mensagem para o computador G. A rota escolhida a princípio é mostrada no cenário (A). Com a mudança de localização do computador C, conforme mostrado no cenário (B), um novo, e mais econômico caminho,

pode ser utilizado na comunicação entre os nodos A e G.

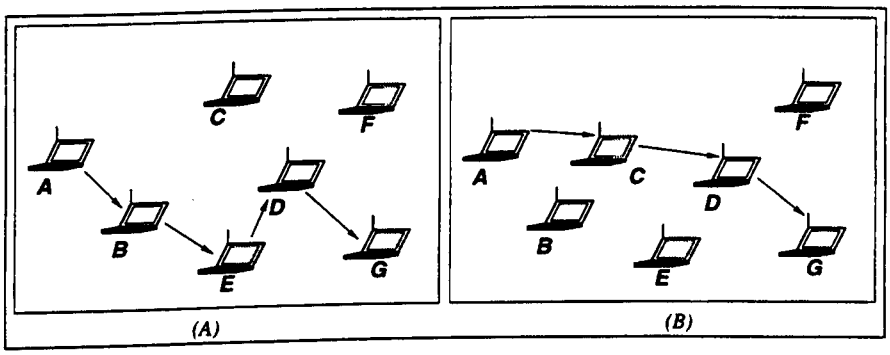


Figura 7.1: Ilustra-se aqui dois possíveis momentos de uma rede, onde se deseja enviar uma mensagem do computador A para o G

O GPSAL suporta dois tipos distintos e independentes de comunicação: datagrama e Circuito Virtual Flexível (CVF). A forma inicial de operação dos dois tipos de comunicação, como será apresentado, é semelhante.

7.1.3 Datagramas

O GPSAL provê a entrega de dados através de datagrama não confiável. Este serviço é semelhante ao oferecido pelo protocolo UDP (*User Datagram Protocol*) do modelo Internet. A forma como o datagrama e o pacote de estabelecimento de CVF são transmitidos é semelhante. No datagrama nenhum circuito é alocado, portanto não existe a manutenção do circuito. Como o serviço é não confiável, o nodo destino não envia nenhuma confirmação de recebimento do datagrama. A figura 7.2, mostra o formato do pacote de datagrama.

O nodo origem, ao enviar um pacote, sugere uma rota aplicando um algoritmo de menor caminho, baseado nas informações obtidas na sua tabela de roteamento, sempre respeitando, é claro, a condição do nodo de poder ser roteador ou nao. Havendo a rota, a mensagem é enviada para o primeiro nodo do caminho encontrado. Caso não consiga enviar o pacote, tenta uma nova rota. O nodo intermediário, ao receber um datagrama, atualiza tanto as informações da sua tabela, quanto as informações no pacote. O nodo verifica a rota para o destino e, se necessário, a altera no datagrama passando a nova sugestão de rota para o próximo nodo. Na verdade, para o datagrama existe apenas a preocupação com a informação local, enquanto no CVF deve existir uma preocupação com o circuito de forma *fim-a-fim*, pois diversos pacotes poderão ser enviados por este caminho. Nos dois casos a decisão é local, mas no datagrama não existe nenhuma grande conseqüência "global", enquanto que no CVF pode existir. Caso a rota seja ruim, por exemplo desperdiçando recursos desnecessariamente, ou não atendendo a contento os requisitos de QoS, esta rota pode afetar o desempenho não só da comunicação mas da rede como um todo. No datagrama o custo do redirecionamento só se justifica se o pacote não estiver se deslocando em direção ao destino.

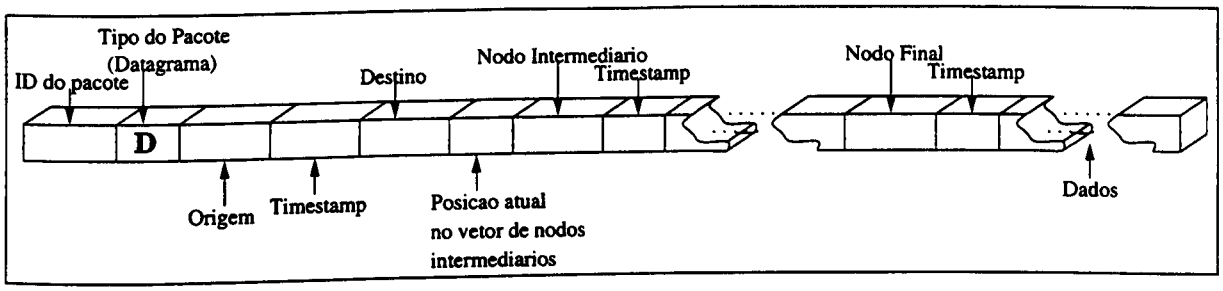


Figura 7.2: Pacote de Datagrama

7.1.4 Circuito Virtual Flexível

O outro serviço que é oferecido pelo GPSAL é o Circuito Virtual Flexível (CVF). O CVF é um circuito virtual, como definido em redes de computadores tradicionais, com a diferença de que pode ser modificado dinamicamente em função da mobilidade dos computadores.

Estabelecimento do Circuito Virtual Flexível

No processo de estabelecimento do CVF, parâmetros de Qualidade de Serviço (QoS– *Quality of Service*) podem ser negociados entre o nodo origem, os nodos roteadore intermediários e o destino. Estes requisitos devem ser inseridos, pela origem, no pacote de estabelecimento de rotas. A cada passo o nodo intermediário verifica se pode ou não atender aos parâmetros de QoS requisitados. Caso não consiga retorna o pacote para o nodo anterior, que tenta então encontrar uma nova rota. Não havendo outra rota, o pacote é passado para o nodo anterior e os recursos alocados para este circuito são liberados. Este processo se repete até que o pacote chegue à origem pelo mesmo caminho, liberando o CVF previamente reservado. Ao chegar à origem, pode-se tentar um novo caminho ou assinalar a rota como não encontrável. Serão descritos detalhadamente a seguir os passos executados pelo protocolo no estabelecimento do circuito virtual e é apresentado na figura 7.3 a forma como os pacotes são enviados no CVF.

Na Origem

O nodo origem cria o melhor caminho possível, baseado nas informações de sua tabela, e insere essa "sugestão de rota" no cabeçalho do pacote de requisição de circuito. Esta informação não é precisa, dado que o Host Móvel (HM) não pode ter um "instantâneo" do estado da rede; Por isto é desejável que haja redirecionamentos para assim encontrar um caminho melhor que o anterior. No caso do primeiro nodo não aceitar a comunicação, seja por que não está mais ao alcance, ou por que não pode suportar os parâmetros de QoS, outra rota é escolhida e o pacote enviado. O tempo de vida (*Time To Leave* - TTL) da requisição de pacotes é setado. Ao receber a Confirmação de Circuito, o nodo origem atualiza a tabela, com as informações recebidas no pacote, e inicia a comunicação pelo canal alocado. Caso ocorra um *timeout* para o circuito, ou a origem receba uma recusa de conexão, é enviada uma nova requisição, preferencialmente por outro caminho. O programa 7.1 mostra em forma algorítmica o funcionamento do nodo origem da

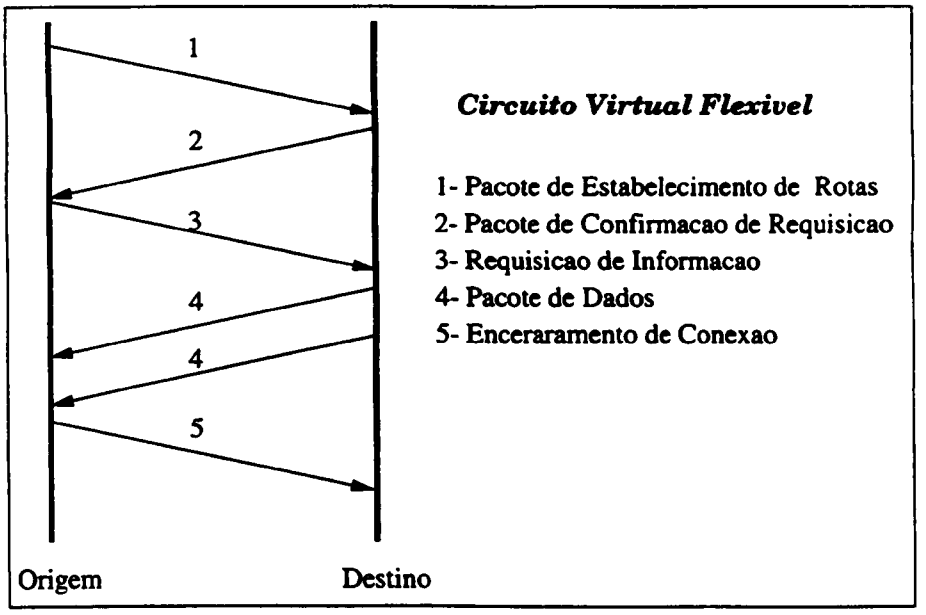


Figura 7.3: Escolha de um novo caminho pelo destino da comunicação

comunicação, e a figura 7.4 mostra o pacote de reqisição de CVF.

Nodo Origem

```
Início
   Espera existir pacote no buffer;

   Se Pacote a enviar
      Enquanto O pacote_enviado == Sucesso  OU
               Numero máximo de tentativas não for atingido
         Cria Caminho para o destino;
         Envia Pacote para o próximo nodo;

         Se o próximo nodo aceitou o pacote
            pacote_enviado = Sucesso;
         Se não
            pacote_enviado = Falha;
         Fim Se; /* próximo nodo aceitou o pacote */

         Se pacote tipo == CVF
            Inicializa o TTL de requisição do CVF;
         Fim Se; /* pacote tipo == CVF */

         Incrementa número de tentativas;
```

```
   Fim Enquanto;
   Se pacote é do tipo CVF
      Espera Pacote de Confirmação de CVF
      O TTL de requisição estourou
         Se Sim
            Volta no enquanto e reinicia o processo de envio;
         Se Não
            Termina processamento;
         Fim Se; /* TTL de requisição estourou */
   Fim Se; /* É pacote de CVF */
Fim Se Pacote a enviar;

Se Pacote a receber
   Verifica se é um pacote de Confirmação de CVF
      Se sim
         Envia os dados pelo caminho indicado;
      Se não
         Verifica se é um pacote de Recusa de Conexão
         Se Sim
            Verifica se a recusa veio do destino
               Se Sim
                  Termina processamento do CVF;
               Se Não
                  Encontra uma nova rota até o destino;
                  Envia o pacote;
               Fim Se; /* Verifica destino recusou conexão */
         Se Não
         Verifica se pacote de encerramento de conexão
         Se Sim
            Espera Pacote de Confirmação de rota
            Se timeout na espera
               Envia nova requisição de CVF;
            Se Não
               Envia os pacotes pelo CVF alocado;
            Fim Se; /* timeout na espera */
         Se Não
            Pacote de tipo não esperado;
         Fim Se; /* pacote de encerramento de conexão */
      Fim Se; /* pacote de Confirmação de CVF */
```

```
   Fim Se Pacote a receber;
Fim.
```

Programa 7.1: Funcionamento interno do nodo origem no GPSAL

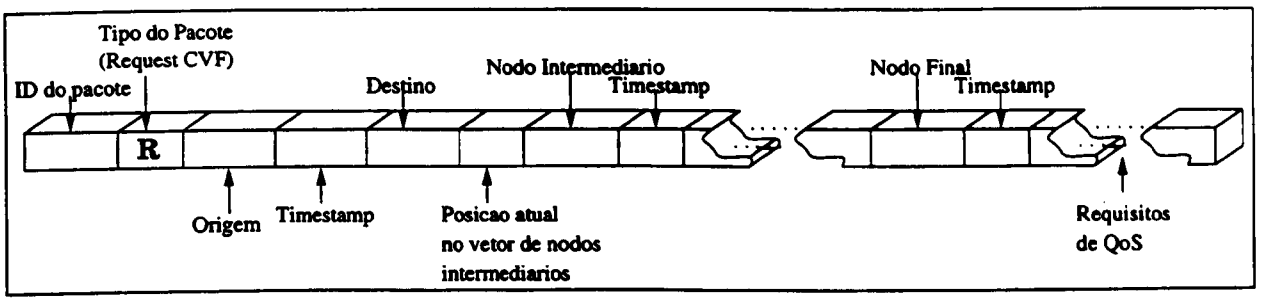


Figura 7.4: Pacote de requisição de CVF

No Nodo Intermediário

O nodo intermediário associa a requisição de circuito a um TTL, reservando os recursos pedidos. Não podendo alocar os recursos, o nodo recusa a comunicação. Mas, de qualquer forma, o nodo atualiza a sua tabela de roteamento com as informações sobre os nodos do cabeçalho do pacote, caso estas sejam mais recentes que as suas.

No caso de aceitação do CVF, se as informações de roteamento no nodo intermediário forem mais recentes, pode haver um redirecionamento do pacote se isto for interessante (o que pode ser interessante em termos de redirecionamento será discutido na seção 7.2, onde serão citadas as métricas para reavaliação de rotas). Esta possibilidade de redirecionamento se torna cada vez mais interessante, já que, a cada novo nodo, o pacote tende a estar mais próximo do destino. Dessa forma o pacote pode encontrar informações mais recentes sobre o destino ficando assim com condições de gerar uma melhor estimativa sobre sua posição. O redirecionamento, quando existir, deve sempre ocorrer a partir do nodo atual, nunca deve ser alterado o caminho já percorrido, nem nodos já percorridos devem participar do circuito. Os nodos anteriores têm informações melhores sobre suas próprias regiões, sendo assim, o nodo atual não poderia contribuir muito para a precisão da rota no caminho até ele. Com esta restrição evitam-se *loops* de pacotes. Outro motivo é que, como se trata de um circuito virtual, todos os recursos dos nodos da rota anteriores deveriam ser liberados e os da nova rota alocados, aumentando assim a complexidade do algoritmo.

Caso o nodo intermediário não consiga alocar uma rota a partir dele, libera os recursos alocados a este circuito, atualiza suas informações no pacote e devolve o pacote para o nodo anterior, com o status de recusa de conexão, como pode ser observado na figura 7.5. O nodo anterior deve então se encarregar de tentar encontrar uma nova rota. Isto pode ocorrer em cada nodo intermediário até que a origem seja alcançada, como pode ser visto na figura 7.6.

Caso o nodo não tenha uma confirmação do circuito estabelecido até que o TTL de confirmação de CVF expire, o circuito, e os recursos correspondentes a ele, são liberados

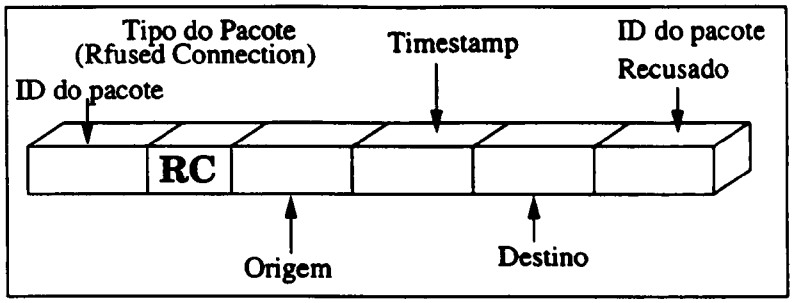


Figura 7.5: Pacote de Recusa de Conexão

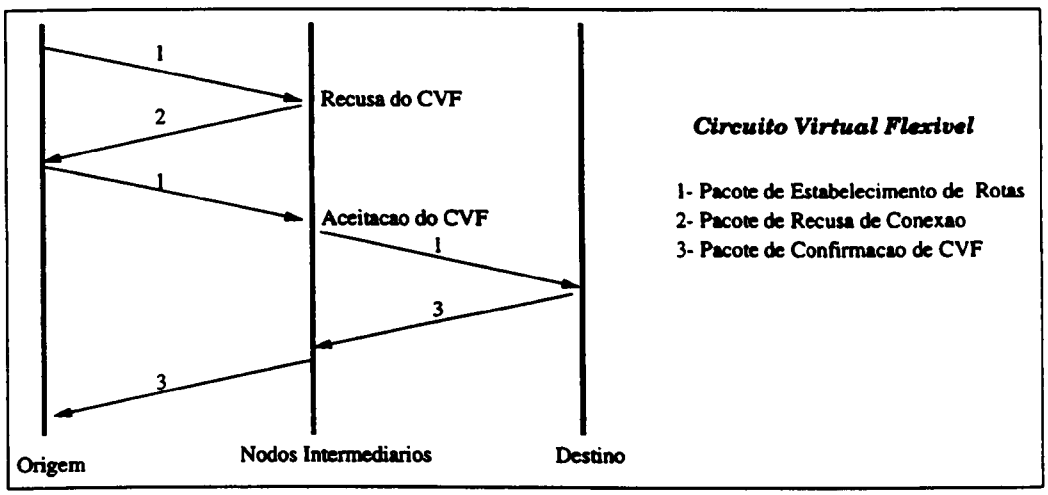


Figura 7.6: Caso o nodo intermediário não consiga alocar os recursos

para outras possíveis conexões. Um nodo, ao receber a confirmação de um CVF, atualiza as informações de sua tabela com relação às informações do pacote e às informações referentes a ele mesmo no pacote, repassando-o para o nodo seguinte. Neste momento, o nodo cria outro TTL, só que agora maior, que é de espera por utilização desse canal. Caso este TTL expire ou o nodo receba um pacote de encerramento de conexão, os recursos alocados a este CVF são liberados. O programa 7.2 mostra em forma algorítmica o funcionamento do nodo intermediário.

Nodo Intermediário

```
Início
   Espera existir pacote no buffer;
   Atualiza tabela de roteamento com as informações do pacote;
   Atualiza seus próprios dados no pacote;
   Verifica se é o destino da comunicação
      Se Sim
         Processa o pacote como sendo a Origem;
      Se Não
         Caso Pacote Seja :
```

```
Um pacote de criação de CVF
   Verifica se pode alocar os recursos exigidos
       Se sim
          Reserva os recursos;
          Envia o pacote para o próximo nodo;
          Seta o TTL de espera por confirmação para este CVF;
       Se Não
          Envia um pacote de Recusa de conexão
          para o nodo anterior;
       Fim Se; /* pode alocar os recursos exigidos */

Um pacote de Recusa de conexão
    Verifica se é procedente do destino
       Se Sim
          Desaloca recursos reservados
          para o CVF em questão;
          Repassa o pacote para o próximo nodo;
       Se Não
          Tenta encontrar novo caminho até o destino
          Se conseguir
             Envia o pacote para o próximo nodo;
          Se não
             Libera os recursos alocados;
             Envia um pacote de Recusa de conexão
             para o nodo anterior;
          Fim Se Conseguir;
       Fim Se; /* procedente do destino */

Um pacote de Confirmação de CVF
    Aloca os recursos;
    Repassa o pacote para o próximo nodo;
    Seta o TTL de utilização do circuito;

Um pacote de encerramento de conexão
    Desaloca os recursos reservados a este CVF;
    Repassa o pacote para o próximo nodo;

Um pacote de Formiga
    Verifica se deve alterar o caminho;
```

```
            Se sim
               Altera o caminho;
            Fim Se;
          Repassa  a formiga para o próximo nodo;

       Um pacote de Datagrama
           Verifica se deve alterar o caminho;
            Se sim
               Altera o caminho;
            Fim Se;
           Repassa o Datagrama para o próximo nodo;

       Um outro pacote qualquer
           Pacote de tipo não esperado;

     Fim Caso;

  Fim Se; /* destino da comunicação */
Fim.
```

Programa 7.2: Funcionamento interno do nodo intermediário no GPSAL

No nodo Destino

Chegando uma requisição de circuito virtual ao destino, este ainda pode refazer o circuito, caso tenha informações mais novas ou melhores que as do pacote de requisição. Para isto, envia um pacote de confirmação de conexão pelo novo caminho à origem, e um pacote de encerramento de conexão pelo caminho previamente reservado, liberando os recursos do caminho para outras conexões, como pode ser visto na figura 7.7. Os nodos intermediários, ao receberem uma resposta de conexão sem uma prévia alocação de circuito, agem como na requisição de circuito virtual. Porém, no caso de aceitação, aloca o circuito realmente, não fazendo apenas a reserva dos recursos.

Não sendo o caso de uma alteração de rota, o destino retorna a confirmação de circuito pelo mesmo caminho que a mensagem chegou. Sem esquecer, é claro, de atualizar sua tabela com os dados recebidos no pacote e suas informações no pacote. O programa 7.3 mostra, em forma algorítmica o funcionamento do GPSAL no nodo Destino da comunicação e as figuras 7.8 e 7.9 apresentam o formato dos pacotes utilizados.

No Nodo Destino

```
Início
```

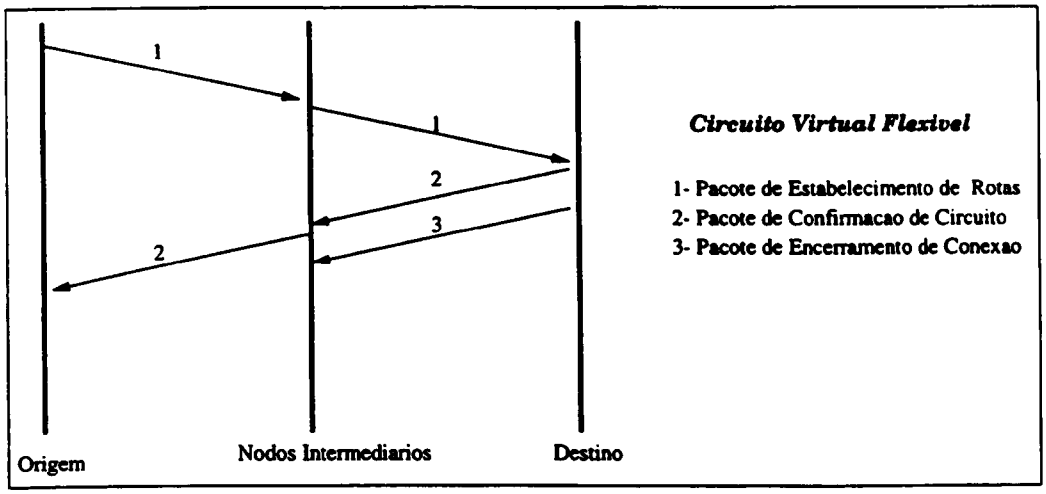


Figura 7.7: Quando, no estabelecimento de um CVF, o destino escolhe um novo caminho

```
Espera existir pacote no buffer;
Atualiza tabela de roteamento com as informações do pacote;
Atualiza seus próprios dados no pacote;
Verifica se é o destino da comunicação
   Se Sim
      Verifica se é um pacote de criação de CVF
         Se sim
            Verifica se o caminho é ótimo
               Se sim
                  Responde a rota com uma Confirmação de CVF;
               Se Não
                  Envia um pacote de Encerramento de conexão
                  pelo caminho reservado;
                  Envia uma Confirmação de CVF pelo melhor
                  caminho;
               Fim Se; /* O caminho é ótimo */
         Se Não;
            Verifica se é um pacote Formiga
               Se sim
                  Cria um caminho de retorno para a formiga;
                  Envia a formiga para a Origem;
               Se Não /* Não é uma formiga */
                  Verifica se é um pacote de Datagrama
                     Se Sim
                        Recebe o pacote de Datagrama;
                     Se Não
```

```
                    Pacote de tipo não esperado;
                Fim Se; /* Pacote de Datagrama */
            Fim Se; /* pacote Formiga */
        Fim Se; /*pacote de criação de CVF */
    Se não
       Reenvia pacote;
    Fim Se; /* destino da comunicação */
Fim.
```

Programa 7.3: Funcionamento interno do nodo destino no GPSAL

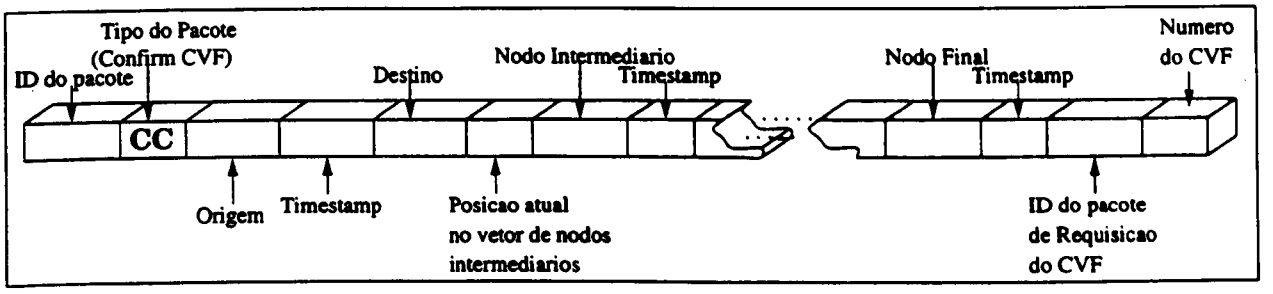


Figura 7.8: Pacote de Confirmação de CVF

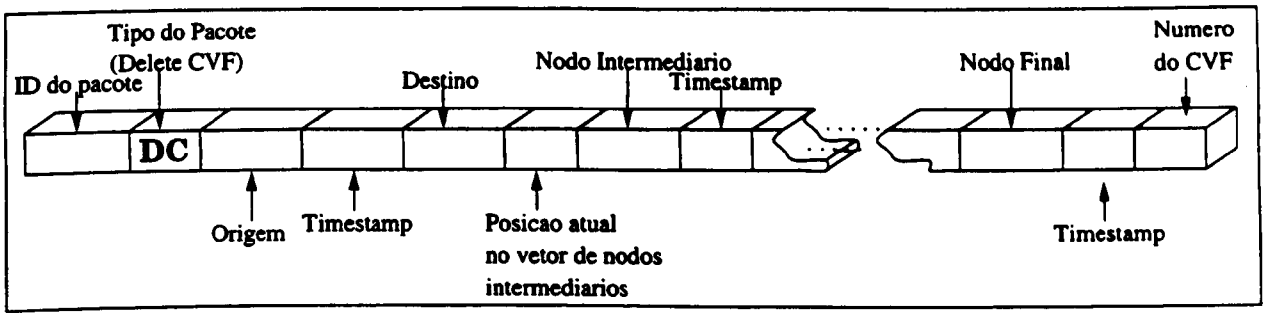


Figura 7.9: Pacote de encerramento de conexão

Manutenção do CVF

O circuito virtual no GPSAL é dinâmico, pois mesmo depois de ser estabelecido, o CVF pode ser alterado se for "vantajoso". As condições em que um CVF pode ser alterado serão discutidas na seção 7.2.

Os pacotes que passam por um CVF, não têm mais a informação sobre a rota completa, mas possuem sempre a informação de localização da origem, do destino e do nodo anterior, como é mostrado na figura 7.10. Isto é feito para não causar uma sobrecarga com a transmissão de todo o CVF, mas caso seja necessário algum redirecionamento, tem-se sempre a informação atualizada dos dois extremos da comunicação. A seguir serão

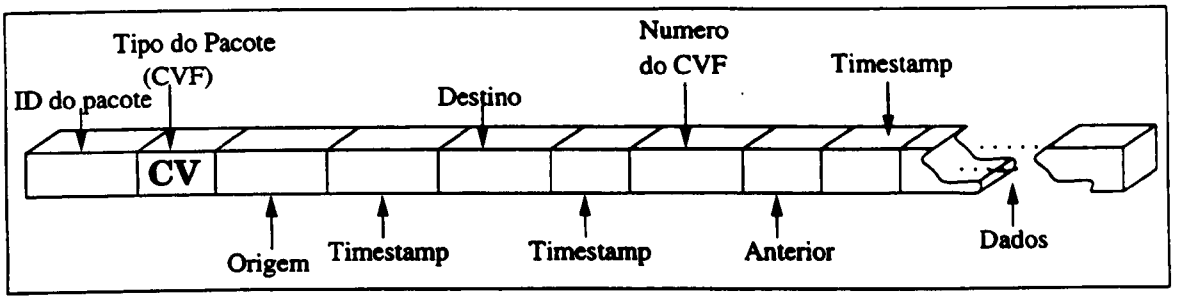


Figura 7.10: Pacote interno ao CVF

apresentadas duas propostas distintas de manipulação do CVF.

Na primeira abordagem, caso ocorra um redirecionamento, o nodo intermediário avisa ao próximo nodo que este num próximo momento esse nodonão fará mais parte do circuito e cria um pacote de confirmação de CVF. Este pacote é enviado pelo melhor caminho conhecido até a extremidade que se está tentando alcançar, que pode ser tanto a origem como o destino do CVF, dependendo de qual parte da rota será atualizada. Este pacote de alteração de rota deve ter o mesmo identificador do circuito atual. O programa 7.3 mostra, a primeira proposta de manutenção do CVF e nas figuras 7.12, 7.11 e 7.13 são apresentados os formatos dos pacotes utilizados.

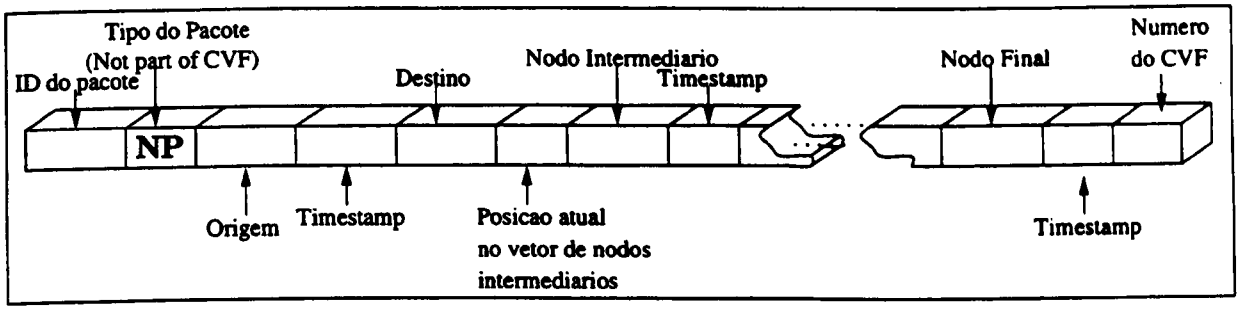


Figura 7.11: Pacote de aviso de não mais participação no CVF

```
Manutenção do CVF
   Verifica se não existe algum caminho mais vantajoso
   para o circuito em questão ou se a rota está em vias
   de ser quebrada
      Se sim
         Avisa ao próximo nodo que o CVF em questão
         vai ser desviado;
         Envia um pacote de Confirmação de CVF
         pelo novo caminho
      Fim Se; /* não existe algum caminho mais vantajoso */
Fim Manutenção do CVF;
```

Programa 7.4: Primeira abordagem de manutenção do CVF

A manipulação do pedido por outros nodos é basicamente a mesma dispensada à resposta à conexão sem prévia alocação, a não ser que algum destes nodos participe do antigo circuito. Neste caso, como o circuito é o mesmo, este nodo considera, para o redirecionamento, os recursos já utilizados pelo antigo CVF. O nodo da extremidade envia um pacote de liberação de circuito para os nodos do antigo circuito. Todos os nodos, a menos que tenham recebido o pacote de redirecionamento em um instante anterior, liberam o CVF. O objetivo desse pacote é liberar os recursos previamente alocados. Caso os nodos intermediários não recebam esse pacote, os recursos alocados serão liberados assim que o valor do TTL de utilização do circuito expirar. O nodo da extremidade, se for o destino, envia pelo novo canal uma confirmação da última mensagem recebida. Caso existam mensagens no caminho do antigo circuito, elas podem ser ignoradas, já que o nodo pode recebê-las novamente pelo novo caminho, ou eventualmente, podem ser consideradas se chegarem em tempo hábil.

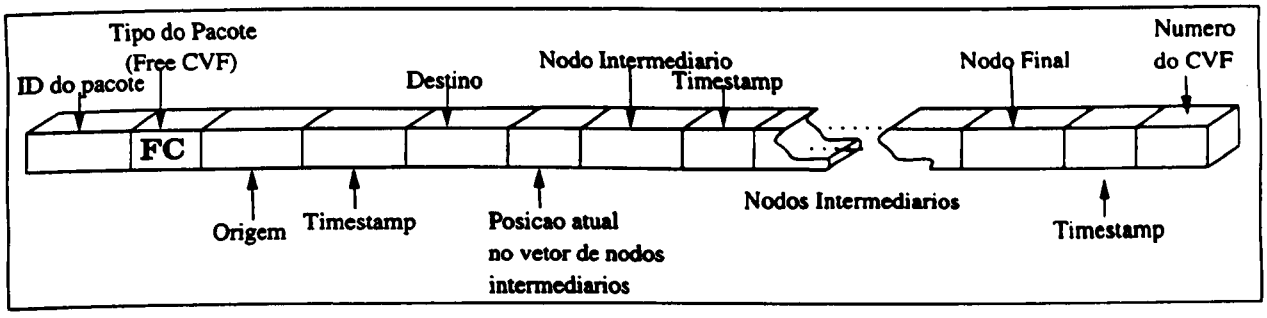


Figura 7.12: Pacote de liberação de circuito

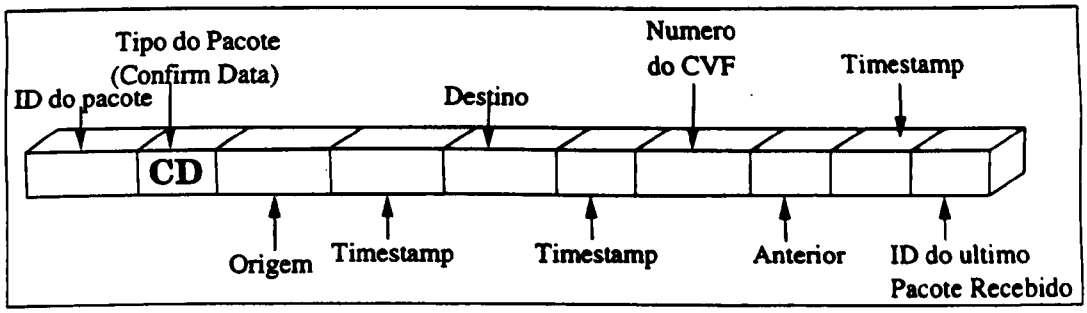


Figura 7.13: Confirmação da ultima mensagem recebida

Na segunda forma de se tratar o problema, cada nodo, ao perceber uma necessidade de fazer o redirecionamento, envia uma mensagem de redirecionamento de rota para o nodo final da parte a ser contornada, o nodo da ponta. Esta mensagem também é tratada pelos outros nodos como a resposta à conexão sem prévia alocação. A diferença entre esta abordagem e a anterior é que o nodo da ponta da parte a ser contornada apenas responde o pedido ao nodo que iniciou o processo, mas continua utilizando o circuito antigo. O novo circuito só é utilizado quando a origem o liberar para uso. O motivo disto é impedir que diversos nodos redirecionem o circuito independentemente. Com o controle centralizado

na origem, pode-se escolher sempre o melhor redirecionamento entre todos os possíveis em um determinado momento.

Na figura 7.14 tem-se um exemplo de redirecionamento de CVF. O nodo I3 percebe que I4 está se afastando e que será necessário refazer o CVF. O nodo I3 envia então um pacote de pedido de estabelecimento de nova rota para I'4, e outro de aviso que envia para para I4. A manutenção deve ser, na medida do possível, pró-ativa, procurando contornar os problemas antes que ocorram. Desta maneira, pode-se manter sempre o CVF operacional, garantindo a qualidade de serviço negociada. Outra característica importante da manutenção do CVF é que o identificador do circuito virtual permanece sempre o mesmo, tornando a alteração de rota transparente para as camadas superiores.

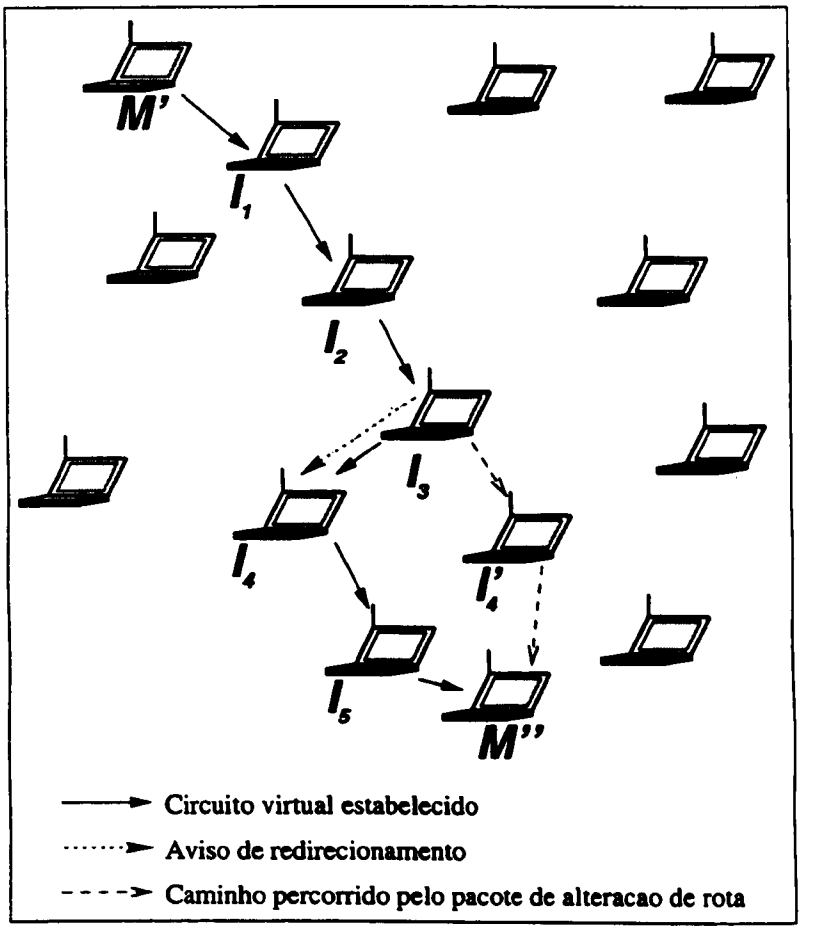


Figura 7.14: Alteração do Circuito Virtual Flexível

7.1.5 Troca de Tabelas

Um novo host móvel, ao conectar-se à rede ad hoc, ouve o meio até que encontre um nodo vizinho. Assim que encontrar, envia um "pedido de tabela" e recebe em troca a tabela completa do vizinho. Desta forma, o HM tem acesso à posição de todos os nodos da rede móvel. Esta é a única vez que toda a tabela é transmitida. A partir deste momento, o nodo só receberá, e enviará, periodicamente informações de atualização relativas à esta tabela. As trocas de tabelas são periódicas e somente são trocadas atualizações com relação à

última troca de tabelas. Este procedimento está descrito no programa 7.5. Nas figuras 7.15 e 7.16 são apresentados os formatos dos pacotes utilizados.

```
Nodo novo na rede
   Ouve a rede até que encontre algum nodo;
   Encontrando o nodo, enviar um pedido de tabela;
   Atualiza a tabela com as informações que recebeu;
   Liga um temporizador para avisar quando terá que enviar
   suas atualizações para os vizinhos;
Fim Nodo novo na rede;
```

Programa 7.5: Procedimento executado por um nodo ao entrar na rede

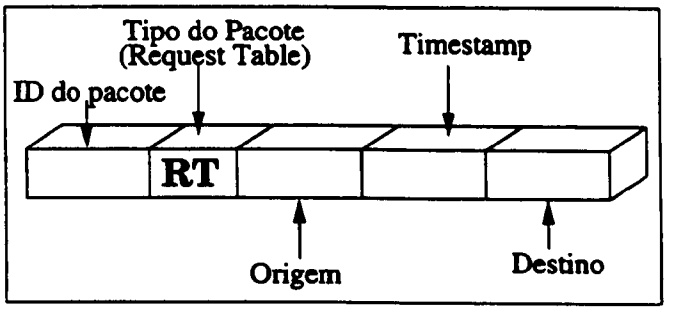


Figura 7.15: Pacote de pedido de tabela

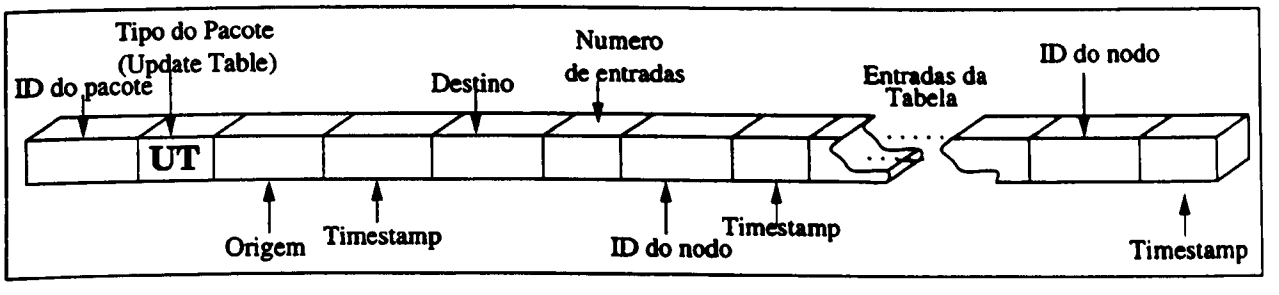


Figura 7.16: Pacote de atualização de tabela

O *timestamp* das informações garante que o nodo não vai trocar informações mais novas em sua tabela por informações mais antigas. Todas as vezes que o nodo repassa informações sobre ele, seja através de trocas de tabelas, formigas, ou mesmo atualização dentro de pacotes normais, se houver alguma alteração de posicionamento, o nodo atualiza o *timestamp* indicando que houve alteração na posição do nodo.

7.1.6 Formigas

As formigas são agentes enviados para nodos aleatórios da rede. O objetivo principal desses agentes é coletar e disseminar informações sobre localização dos nodos. As formigas

armazenam informações sobre os *hops* da rota que estão percorrendo. Isto possibilita aos nodos por onde a formiga passa obter informações novas sobre nodos distantes na rede. Esta informação é particularmente interessante no momento da troca de tabelas, pois os nodos vizinhos estarão aptos a trocar informações sobre nodos distantes, melhorando desta forma a qualidade das trocas de tabelas de roteamento. Naturalmente, existe um *overhead* associado a este processo. Este *overhead* será discutido com mais detalhes no capítulo 8.

O destino da formiga pode ser um nodo qualquer da rede, não é necessário que seja um computador com o qual o nodo mantém contato, ou mesmo com quem estabelecerá uma comunicação num curto espaço de tempo. No programa 7.6, é descrito o processo de decisão de envio ou não de formiga por cada nodo da rede e na figura 7.17 é apresentado o pacote de formiga enviado pela rede. A idéia principal dos agentes formiga é disseminar o máximo possível as informações de roteamento, sem consumir uma quantidade excessiva de recursos da rede. Mas algumas heurísticas podem ser utilizadas. Uma possibilidade é escolher o nodo da tabela de rotas que está com a informação mais antiga. Ao se escolher este destino, o nodo origem irá atualizar as informações que provavelmente estão erradas, pois quanto maior a idade da informação maior é a probabilidade do nodo ter se movimentado e de esta informação estar errada. Outra possibilidade é escolher o nodo que esteja mais distante geograficamente. Neste caso o pacote pode trafegar por mais nodos na rede, adquirindo assim mais informações sobre a rede.

```
Enviar Formiga
   Se o resto do retorno de uma função aleatória por 100 é <
   que a probabilidade do nodo gerar formiga então...
         O nodo deve enviar uma formiga para algum nodo da rede;
      Fim Se; /*  Posso enviar formiga */
Fim Enviar Formiga;
```

Programa 7.6: Decisão de se enviar ou não uma formiga no GPSAL

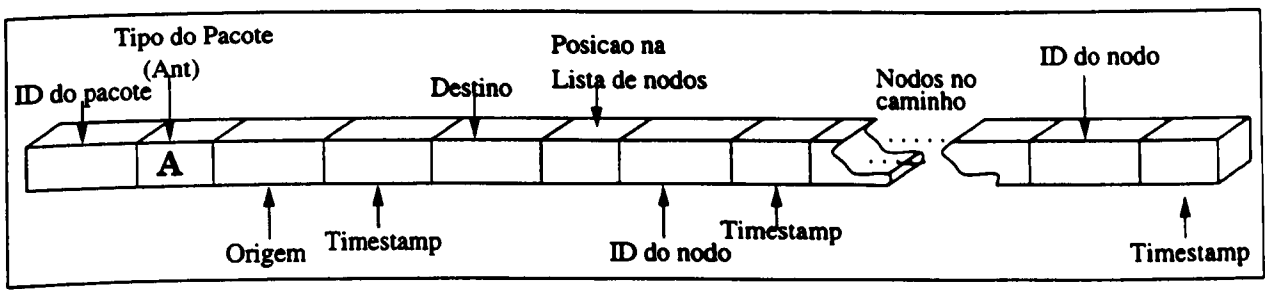


Figura 7.17: Pacote de Formiga

A rota que a formiga utiliza para chegar ao destino não precisa ser necessariamente ótima, já que a idéia é disseminar as informações de roteamento, não sendo exigido que o pacote retorne no menor tempo possível. Quando um computador móvel recebe uma

formiga, atualiza sua tabela de rotas, modifica na formiga as informações que estejam desatualizadas a seu respeito e faz um redirecionamento da rota, caso a formiga esteja indo por um caminho errado. O funcionamento de um agente formiga é semelhante ao do envio de um datagrama, a não ser pelo fato de que a formiga deve voltar para o nodo origem preferencialmente por um caminho diferente. Na figura 7.18, pode-se observar o comportamento típico de uma formiga através da rede. O principal objetivo da formiga é o de disseminar informações através da rede ad hoc. Ela normalmente escolhe um caminho qualquer e não o melhor, que no caso da figura 7.18 seria através da rede fixa, pelos nodos fixos F' e F". Ao invés disto, o caminho escolhido passa pela rede móvel.

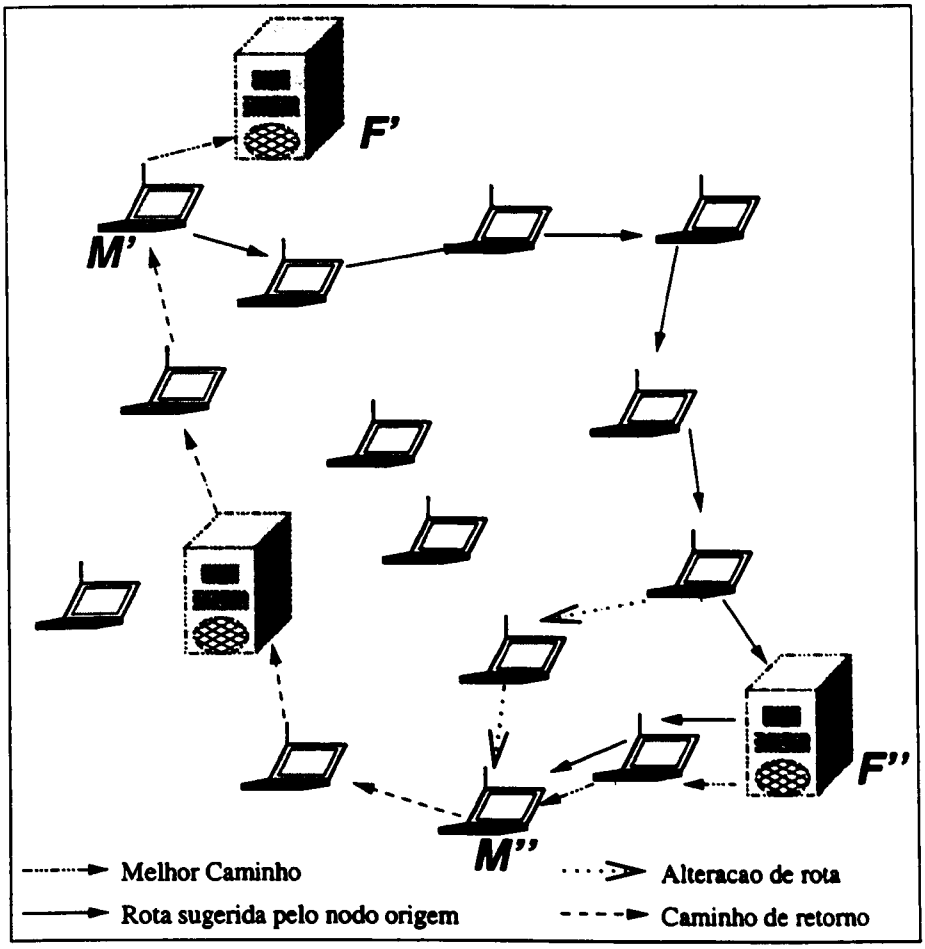


Figura 7.18: Comportamento típico de um agente formiga sendo enviado do nodo móvel M' para o nodo móvel M"

Uma das principais vantagens das formigas é que além de disseminar a informação sobre os nodos do caminho, como efeito colateral, tornam as trocas de tabelas mais atualizadas e confiáveis. Agora os nodos não têm acesso apenas a informações sobre o estado de seus vizinhos, mas possivelmente também a informações recentes de nodos relativamente distantes na rede. E estas informações, através das trocas de tabelas, são difundidas para nodos vizinhos. Desta forma, nodos que nunca receberam ou enviaram uma formiga sequer, podem se beneficiar das informações contidas nas formigas.

Um caso que deixa claro o quão importante é a rápida disseminação das informações sobre os nodos na rede, é a inclusão de um novo nodo na rede. No caso de não haver

um agente formiga, ou seja, apenas trocas de tabelas, o tempo necessário para nodos nos extremos da rede tomarem conhecimento da existência deste novo computador, é proporcional ao tempo de propagação de tabelas entre vizinhos. Mas no caso do GPSAL, o nodo pode enviar uma, ou mais, formigas pela rede, o que irá disseminar a informação tanto da existência do nodo quanto de sua localização, de forma rápida e eficiente.

7.2 Métricas para Redirecionamento de Rotas

Uma das principais características do GPSAL é o redirecionamento de rotas por nodos intermediários. Este, na verdade, é um dos maiores responsáveis pelo bom desempenho do algoritmo. Tendo em vista isto, os redirecionamentos devem ocorrer com o maior cuidado possível. Os principais pontos a serem observados e considerados no momento de se optar por uma nova rota são:

- A existência de um caminho "provavelmente" melhor que o atual
- A possibilidade de uso da rede fixa
- Informações sobre movimentação do nodo destino

7.2.1 Caminho Melhor que o Atual

Este é certamente o ponto mais genérico e abstrato. Por isso vários parâmetros podem, e devem, ser observados quando consideramos um novo caminho. Caso o host móvel tenha informações de que existe um caminho menor, ele deve alterar o atual. Se o nodo intermediário perceber que algum outro nodo está se movimentando de forma a causar uma realocação do CVF este outro nodo deve ser contornado. Pode também haver renegociação do CVF, se o nodo vizinho informar que está sobrecarregado, mesmo que este ainda possa suprir as exigências de QoS requisitadas. Desta maneira, contornando este nodo, está se tentando não só ajudar o nodo, mas também previne-se de uma possível quebra no circuito num próximo instante. Assim o desempenho da rede como um todo também é aumentado.

Pode acontecer de simplesmente não ser possível o contato com o próximo nodo. Sendo assim, outro caminho tem que ser encontrado. Um outro fator que sempre deve ser levado em consideração em qualquer tentativa de encontrar um melhor caminho, é a possibilidade do uso da rede fixa.

7.2.2 Uso da Rede Fixa

Sempre que um HM tentar encontrar um caminho melhor, deve considerar a possibilidade de usar a rede fixa, se existir alguma disponível (figura 7.19). Ao invés de rotear os pacotes pela rede móvel, pode-se desviá-los pela rede fixa, diminuindo assim o gasto de recursos nos nodos intermediários. O que é levado em consideração aqui, é que o custo para a transmissão de dados através da rede fixa é normalmente menor que na rede móvel, pois a velocidade é maior, a quantidade de erros bem menor [62] e não se tem uma preocupação

tão grande com o gasto de energia. O HM quando cria um caminho considera o custo de comunicação dentro da rede fixa com um baixo custo. Sendo assim, havendo uma rota que passa pela rede fixa que tem o mesmo custo ou é menor que a rota que passa pela rede móvel, esta é automaticamente escolhida. Sempre que possível os nodos devem tentar desviar o tráfego da rede ad hoc para a rede fixa. Assim, diminui-se o gasto de recursos dos nodos intermediários, aumentando a autonomia da rede, pois mais recursos estarão disponíveis.

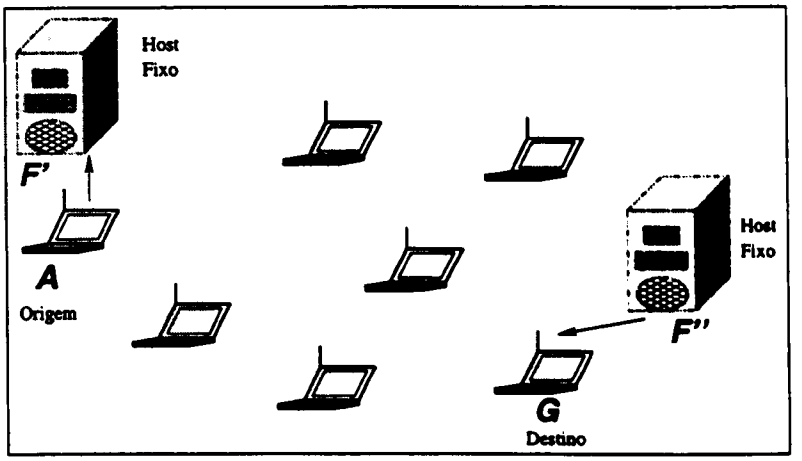


Figura 7.19: Comunicação entre o nodo A e G através da rede fixa.

A rede fixa é considerada "inteligente" o suficiente para encontrar o melhor Host Fixo (HF) em contato com o Host Móvel (HM) destino. Esta colocação não é uma restrição forte se considerarmos que estejam atuando na rede fixa protocolos como o IP-móvel [56] ou o IPv6 [32, 57]. Sempre que uma comunicação tiver como destino um host fixo, ou algum host que não esteja na tabela de roteamento, esta comunicação deve seguir para o host fixo mais próximo, devido aos motivos já discutidos. Este direcionamento acontecerá naturalmente, no caso da comunicação com um HF qualquer, pois como já foi explicado, o GPSAL considera a comunicação dentro da rede fixa com baixo custo.

7.2.3 Movimentações do Nodo Destino

O nodo intermediário pode observar a movimentação do destino e, principalmente, a sua direção e velocidade. Sendo necessário, pode redirecionar a comunicação de forma a "interceptar" o host móvel. Isto é possível se for levado em consideração a mudança de posição entre dois instantes de tempo. É claro que o destino pode mudar sua posição e direção entre o momento que informou pela última vez sua posição e o momento atual, mas uma certa "inércia" é atribuída aos hosts móveis. Isto significa que se acredita que o nodo tem a tendência de continuar o mesmo padrão de movimentação registrado. Mas mesmo que o HM tenha mudado de direção, à medida que o pacote se aproxima de sua antiga posição, as informações sobre o nodo devem melhorar de qualidade, favorecendo a convergência.

7.3 Métricas no Estabelecimento do Circuito Virtual

Os parâmetros de QoS que devem ser consideradas no estabelecimento do CVF são:

- Banda Alocada
  - Taxa constante de transmissão
  - Taxa variável de transmissão
  - Taxa disponível para transmissão
- Atraso na transferência dos pacotes
- Variação no atraso dos pacotes
- Número máximo de nodos no caminho

Banda alocada é o quanto da capacidade no nodo intermediário que deve ser reservada ao CVF e a forma como será feita esta reserva. Este é um dos fatores que aumenta a confiabilidade e a aplicabilidade do protocolo. Um circuito só é alocado se o nodo intermediário puder garantir seus quesitos de QoS, que estão armazenados juntamente com as informações do CVF, para uso no caso de um redirecionamento. A alocação da banda pode se dar em três categorias distintas: taxa constante, variável ou disponível. Taxa constante é utilizada quando um nodo necessita de um canal com disponibilidade para um fluxo constante de dados, uma vídeo conferência por exemplo. Quando é alocado um circuito com uma taxa variável, é passado o valor mínimo e máximo necessários à comunicação nesse circuito. Um exemplo deste tipo de comunicação pode ser a transmissão de vídeos MPEG, pois têm taxas variadas de transmissão. Taxa disponível significa que pode ser alocado ao CVF qualquer banda disponível.

O atraso na transferência do pacote diz respeito ao tempo total entre a saída do pacote da origem até a chegada ao destino. Neste prazo deve estar incluido também o tempo de processamento do pacote. A variação no atraso dos pacotes (*jitter*) é a variação entre o envio de dois pacotes consecutivos. Este parâmetro é especialmente importante para aplicações de vídeo e voz, onde podem ocorrer distorções se a variação máxima não for respeitada. O número máximo de *hops* em um circuito pode ser necessário para algumas aplicações como forma de tentar otimizar o caminho dos pacotes.

7.4 Complexidade do GPSAL

A complexidade do GPSAL, em termos de processamento, é o da função de cálculo do *Shortest Path*. Existem implementações para o algoritmo de Dijkstra de complexidade O(m+nC) [1], onde:

- nC : Limite superior do maior arco da rede;
- m : Número de atualizações necessárias;

7.5 Comentários

Neste capítulo vimos a definição do Algoritmo GPSAL. Estão descritas aqui as atitudes que os nodos devem tomar quando um pacote, é roteado quais os pacotes existentes e o significado de cada mensagem. Como podemos observar, o GPSAL é um algoritmo relativamente simples e fácil de ser implementado. Como veremos no capítulo 8, o algoritmo apresenta um bom desempenho, quando comparado com o LAR.

Capítulo 8

Resultados de Simulações

Neste capítulo serão descritos os diversos experimentos realizados com o intuito de caracterizar e avaliar a eficiência do GPSAL. Os dados utilizados nestas comparações foram obtidos através da simulação dos algoritmos GPSAL, LAR1 e LAR2.

8.1 Simulador

Os algoritmos LAR1, LAR2 e o GPSAL foram simulados em linguagem C padrão. Os experimentos foram executados em máquinas Sun sparc Ultra-1, com $128MB$ de memória RAM, e Sun sparc Ultra-Enterprise com $512MB$ de memória RAM, com o sistema operacional SunOS 5.5.1. O simulador foi projetado para tratar os pacotes da mesma forma que o protocolo IEEE 802.11 [3, 20], trabalhando em iterações. O tempo de uma iteração, é o tempo necessário para que cada nodo trate os pacotes enviados a ele na iteração anterior, e envie os pacotes que têm armazenado no buffer de envio. Desta forma, a camada MAC (*Medium Access Control*) também é simulada. O simulador é todo parametrizado, ao contrário dos simuladores atuais, como por exemplo o NS *Network Simulator* de Berkeley [26]. Existe uma versão deste simulador [58] para aceitar nodos móveis, mas até o lançamento da última versão, de dezembro de 1999, não era possível inserir nodos fixos e móveis, ou mesmo criar nodos com alcances variados. No simulador desenvolvido para os experimentos, desta dissertação além destes dois, diversos outros valores podem ser informados, dando uma maior liberdade e uma maior flexibilidade para a realização dos testes.

O simulador pode gerar cenários ou executar simulações sobre cenários já gerados. Nos arquivos de cenários estão a movimentação dos nodos, todos os pacotes enviados, possíveis variações de alcance e perda de pacotes. Isto garante a fidelidade das informações adquiridas, pois todas as simulações podem ser repetidas fielmente, tanto no LAR como no GPSAL.

8.2 Os Modelos Utilizados na Simulação

Diferentes modelos foram utilizados nas simulações. Isto foi necessário para que se pudesse submeter o algoritmo ao maior conjunto de situações possível. O intuito deste conjunto de variações é testar a eficiência do protocolo nas mais diversas situações. De forma geral, os parâmetros que foram variados nos testes foram a velocidade dos nodos, o alcance da comunicação, a quantidade de nodos da rede, a quantidade de formigas e a percentagem de nodos móveis e fixos. O que se está observando quando se variam estes parâmetros são: o número de pacotes trafegando na rede em um dado momento, o número total de pacotes perdidos (inclundo formigas e pacotes de dados), o número de pacotes de dados perdidos, a convergência e o tamanho dos caminhos percorridos pelos pacotes. Todos os pontos dos gráficos mostrados neste capítulo são o resultado da simulação dos protocolos em de mais de 30 cenários diferentes.

Em todas as simulações as trocas de tabelas no GPSAL ocorrem a cada três iterações. Este espaço curto entre as trocas favorece o LAR, já que as trocas de tabelas são freqüentes e têm um alto custo. Com o objetivo de normalizar as comparações, já que o LAR suporta apenas comunicação através de datagramas, tanto a comunicação do GPSAL com formigas quanto a do GPSAL sem formigas serão feitas somente através de datagramas. A parte de CVF proposta no capítulo 7 não foi implementada nesta bateria de testes do algoritmo.

Não foi utilizada nenhuma distribuição para definir a posição dos nodos ou a forma como seria o fluxo de dados entre nodos da rede. Optou-se por não se usar nenhum padrão de mobilidade como recomendado na RFC 2501 [13], onde são tratadas as características de MANETS, como por exemplo que os nodos são livres para se mover livremente pela rede. Mesmo que nas simulações tenha-se atribuído uma certa "inércia" ao nodo, que é uma tendência de continuar no mesmo sentido de movimentação, nenhum padrão de movimentação foi adotado, o que permite o nodo caminhar em qualquer direção.

As seções a seguir serão organizadas de acordo com a variação de cenários e objetivos que motivaram aquele determinado conjunto de experimentos.

8.3 Influência das Formigas na Convergência da Rede

O tempo de convergência é o tempo necessário para que um nodo tome conhecimento que houve alguma alteração no estado da rede, e tome as providências necessarias para que o problema seja contornado. Quanto menor for este tempo melhor, pois menos pacotes vão ser enviados por um caminho pior, ou mesmo inexistente. Esta métrica se torna mais importante ainda quando falamos de redes móveis, pois além da dinamicidade da rede, existe um alto custo, em termos de gasto de energia, associado a rotas erradas e à retransmissão de mensagens.

O modelo usado para simular os algoritmos nesta fase é constituído de um grid de $N \times N$ unidades contendo 70 computadores entre móveis e fixos, sendo que a probabilidade do nodo ser criado como móvel é 90%. A distribuição e movimentação dos computadores é feita de forma aleatória. O alcance máximo de comunicação de uma estação é $0,2N$ unidades a partir de sua posição. Assume-se que cada nodo tem uma velocidade máxima

que é dada pela metade do seu alcance. Esta é uma restrição que tem como objetivo apenas controlar o grau de mobilidade.

A tabela 8.1 e a figura 8.1 mostram os tempos de convergência total e do primeiro nodo para o GPSAL com formigas e para o GPSAL sem formigas. Neste experimento, criou-se um cenário aleatório de movimentação e envio de pacotes. Em um determinado momento da simulação para-se a rede e nenhum nodo se movimenta mais. A partir deste momento conta-se o tempo, em iterações, gasto para que os nodos tenham o conhecimento da posição real de todos os outros nodos da rede. O tempo de convergência do primeiro nodo é o tempo gasto até que o primeiro nodo conheça a posição real de todos os outros da rede. O tempo de convergência total, é o tempo gasto para que todos os nodos tenham o conhecimento da posição real de todos os outros nodos da rede. Para os dois casos, o GPSAL com formigas teve um tempo de convergência menor. Pode-se observar na tabela 8.1 o ganho do GPSAL com formigas em número de iterações necessárias a convergência.

Na tabela 8.1, o número de iterações para o GPSAL sem formiga não varia, pois é o resultado da execução do algoritmo nos mesmos cenários que o GPSAL com formigas, só que, como não apresenta nenhum componente aleatório, e nem utiliza formigas, o seu valor não se altera com o aumento da probabilidade de gerar formigas. A primeira linha, onde se observa que a probabilidade do nodo gerar formiga é zero para o GPSAL com formigas, significa que apenas os pacotes de dados estão trabalhando como formigas. Mesmo assim, como pode-se observar que o ganho foi considerável. Para a convergência total o ganho em número de iterações foi de 32,88% e para a convergência do primeiro nodo o ganho foi de 37,96%. Deve se observar que este ganho foi conseguido sem nenhum custo adicional, pois foram utilizados somente os pacotes de dados que de qualquer forma já teriam que ser transmitidos. A primeira linha que insere pacotes de formiga é a de probabilidade 0,1 do nodo gerar uma formiga na iteração. Com apenas as 197,4 formigas geradas, num universo de 6 a 8 mil pacotes de dados gerados em média no total por cenário, conseguiu-se um ganho em número de iterações de 38,36% para a convergência total, e de 41,67% para a primeira convergência. Pode-se observar na tabela 8.1 que quanto mais formigas se insere na rede, menor é o tempo convergência. Esta taxa de ganho em termos de convergência diminui gradualmente, mas ainda assim apresenta uma melhora. O máximo testado neste experimento foi a probabilidade de 100% do nodo gerar um pacote de formiga a cada iteração. Isto significa que cada nodo, a cada iteração, gerou uma formiga para um destino qualquer. Neste experimento, em que foram gerados 1486 formigas em média, a taxa de convergência geral do GPSAL sem formiga foi de 43,8 e do GPSAL com formiga foi de 21,0. Isto representa um ganho em número de iterações de 52,5%, enquanto que na primeira convergência o ganho foi de 54,63%.

8.4 O uso da rede fixa

Esta seção tem como objetivo verificar a validade do uso da rede fixa, visando a diminuição do tráfego dentro das redes ad hoc. Com o uso da rede fixa pode-se desviar o tráfego de dentro da rede móvel para a rede fixa, tentando desta forma aumentar a autonomia das unidades móveis e, ao mesmo tempo, melhorar a qualidade do serviço de entrega de men-

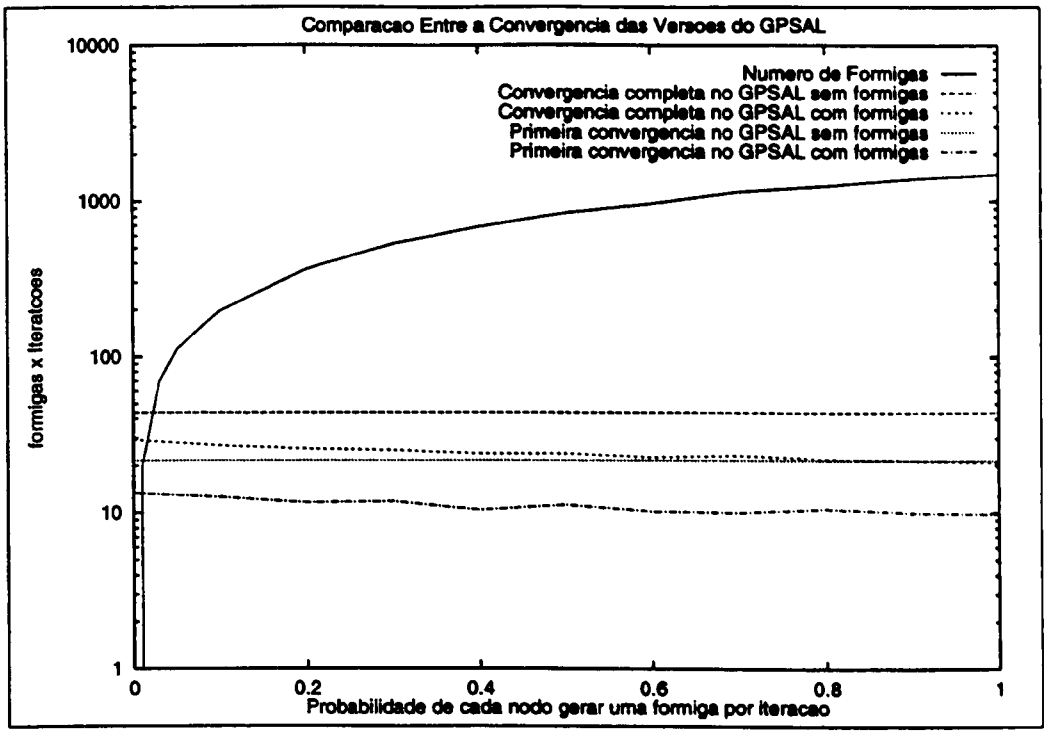


Figura 8.1: Comparação entre os tempos de convergência do GPSAL com e sem formigas

Probabilidade de cada nodo gerar uma formiga por iteração	# de formigas	# de iterações para convergência completa		# de iterações para primeira convergência	
		GPSAL sem formigas	GPSAL com formigas	GPSAL sem formigas	GPSAL com formigas
0,0	0	43,8	29,4	21,6	13,4
0,1	197,4	43,8	27,0	21,6	12,6
0,2	365,6	43,8	25,8	21,6	11,6
0,3	531,2	43,8	25,2	21,6	11,8
0,4	685,0	43,8	24,0	21,6	10,4
0,5	843,8	43,8	24,0	21,6	11,2
0,6	967,6	43,8	22,8	21,6	10,2
0,7	1151,8	43,8	23,4	21,6	10,0
0,8	1255,0	43,8	22,2	21,6	10,6
0,9	1391,0	43,8	21,6	21,6	10,0
1,0	1486,0	43,8	21,0	21,6	9,8

Tabela 8.1: Convergência da MANET usando o GPSAL com e sem formigas

sagens. Nesta simulação levamos em consideração que a rede fixa não tem preocupação com gastos de energia, sendo portanto o custo de transmissão de mensagens na rede fixa inferior ao custo na rede móvel. Além disto, assume-se que a rede fixa é mais confiável e rápida que a rede móvel. Sendo assim, consideramos que todos os nodos fixos podem se comunicar diretamente uns com os outros, e que o custo desta comunicação é quase nulo, se comparado com o custo da comunicação dentro da MANET. Assim, quando o GPSAL escolhe um destino, ele tende a escolher um caminho que passe pela rede fixa. A figura 8.2 mostra o impacto do uso de nodos fixos no tráfego de mensagens referentes à rede ad hoc.

O modelo simulado para o caso da rede fixa possui as seguintes características: 150

nodos em uma região quadrada de $N \times N$, alcance de transmissão de $0,3N$ unidades, velocidade média de $0,04N$/iteração e máxima de $0,08N$/iteração. Em cada iteração o nodo tem a probabilidade de $0,5$ de gerar e enviar um pacote a outro nodo qualquer da rede, sendo analizadas 100 iterações. Podemos observar na figura 8.2 que no período inicial, antes de a rede atingir uma certa estabilidade, os nodos da rede fixa não representam uma grande vantagem, mas à medida que o número de pacotes trafegando na rede aumenta, há uma sensível diminuição no número de pacotes trafegando na rede, quando a rede fixa é usada. Sendo assim há uma economia de recursos na rede, de uma forma geral. Outro ponto que pode ser avaliado neste experimento é o número de nodos intermediários. No experimento da figura 8.2, o tamanho médio dos caminhos é de $3,7275$ para 20% de nodos fixos de $4,6801$ para 10% de nodos fixos e de $7,4481$ para somente unidades móveis na rede. Esta diminuição no tamanho dos caminhos dos pacotes representa, além de uma economia de recursos da rede, uma melhora significativa na qualidade dos serviços oferecida. Quanto menos *links* móveis houverem na comunicação, menor é a chance de perda de pacotes, de erros de transmissão e mesmo de atraso dos pacotes.

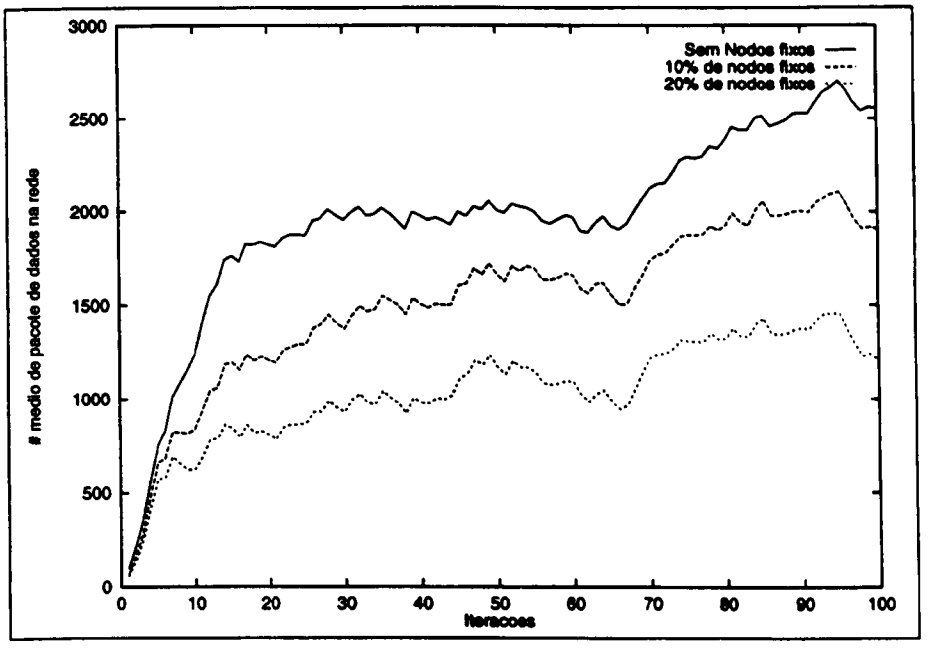


Figura 8.2: Comparação do *overhead* de pacotes gerados na rede ad hoc com o uso ou não da rede fixa

8.5 Caracterização do GPSAL

Od testes descritos nesta seção têm como objetivo avaliar o comportamento do GPSAL em diferentes situações. Foram variados aqui o número de formigas na rede, o número de nodos, o alcance e a velocidade. A comparação foi feita com relação ao número médio de pacotes na rede, de pacotes totais perdidos (formigas inclusive), de pacotes de dados perdidos, de *hops* no caminho e convergência da rede. O modelo de simulação para os testes seguintes, caso não seja explicitamente dito o contrário, são:

- Alcance: 300 *unidades*
- Número de nodos: 30
- Tamanho do cenário: 1000 × 1000 *unidades*
- Pacotes de dados gerados: 100, sempre de várias origens para vários destinos
- Nodos fixos: 20%

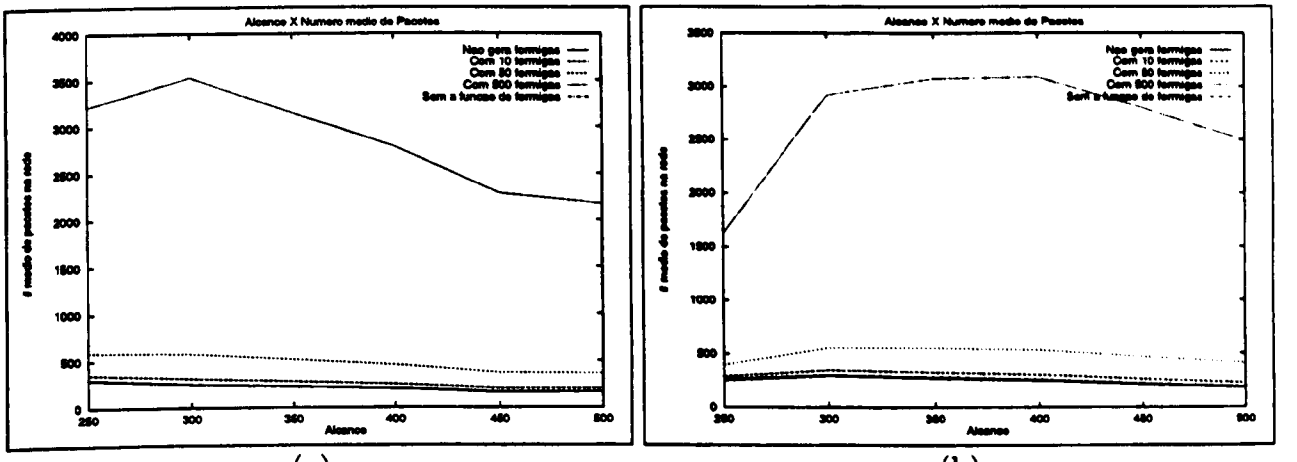
8.5.1 Variação do Alcance

O objetivo destes testes é avaliar o comportamento do GPSAL com relação à variação do alcance dos nodos que participam da rede. Serão analisados o número de pacotes na rede, o número de pacotes perdidos, o número de pacotes de dados perdidos, a convergência da rede e o tamanho dos caminhos. Além disto serão consideradas duas velocidades médias diferentes, de 5 e 25 unidades/iteração com ou sem nodos fixos na rede. Nos gráficos serão apresentadas cinco curvas :

- *Não gera formigas*: não cria pacotes de formiga, se utiliza apenas dos cabeçalhos dos pacotes de dados, que estes agem como formigas armazenando informações. Deste modo não é gerado *overhead* algum, pois os pacotes de dados devem ser enviados de qualquer maneira.
- *Com 10 formigas*: são gerados, durante toda a simulação, 10 pacotes de formigas em pontos aleatórios da rede
- *Com 50 formigas*: são gerados, durante toda a simulação, 50 pacotes de formigas em pontos aleatórios da rede
- *Com 500 formigas*: são gerados, durante a simulação, 500 pacotes de formigas em pontos aleatórios da rede
- *Sem a função de formiga*: além do GPSAL, neste experimento, não gera formigas, nem as informações dos cabeçalhos dos pacotes de dados é colhida. Ou seja, nem os pacotes de dados agem como formiga.

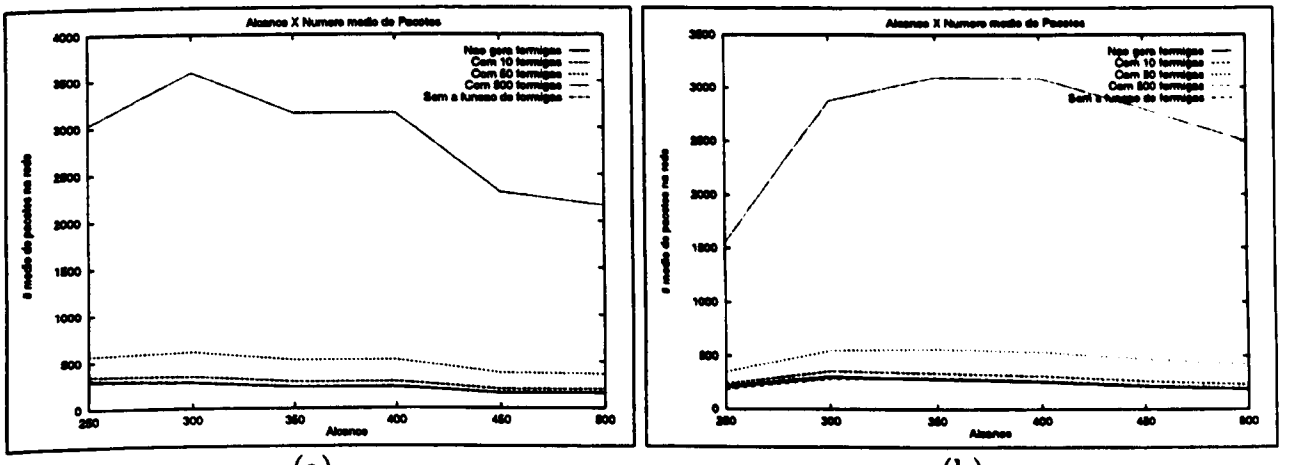
Os gráficos das figuras 8.3 apresentam-se (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, e 8.4 (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração. Mostram o número médio de pacotes na rede durante os experimentos de variação do alcance da comunicação. O comportamento dos experimentos é o esperado, na medida em que os experimentos que geram mais formigas têm um tráfego maior de pacotes. As curvas das figuras 8.3 (a) e 8.4 (a) têm um comportamento semelhante e previsível, pois quanto maior o alcance, mais rápido o pacote vai ser entregue e menos pacotes vão coexistir na rede. O comportamento da curva do GPSAL sem formigas e sem os pacotes de dados agindo como formigas, nas figura 8.3 (b) 8.4 (b), pode ser explicado pela perda de pacotes e pela entrega mais rápida de pacotes. Quando

o alcance é pequeno, muitos pacotes de formiga são perdidos, diminuindo assim o número de pacotes na rede, até atingir um ponto de equilíbrio. O número de pacotes começa a diminuir porque, com o aumento do alcance, os pacotes são entregues mais rapidamente, diminuindo o volume de pacotes médio na rede.



(a) (b)

Figura 8.3: Comparação entre a relação do número médio de pacotes trafegando na rede e o alcance dos nodos

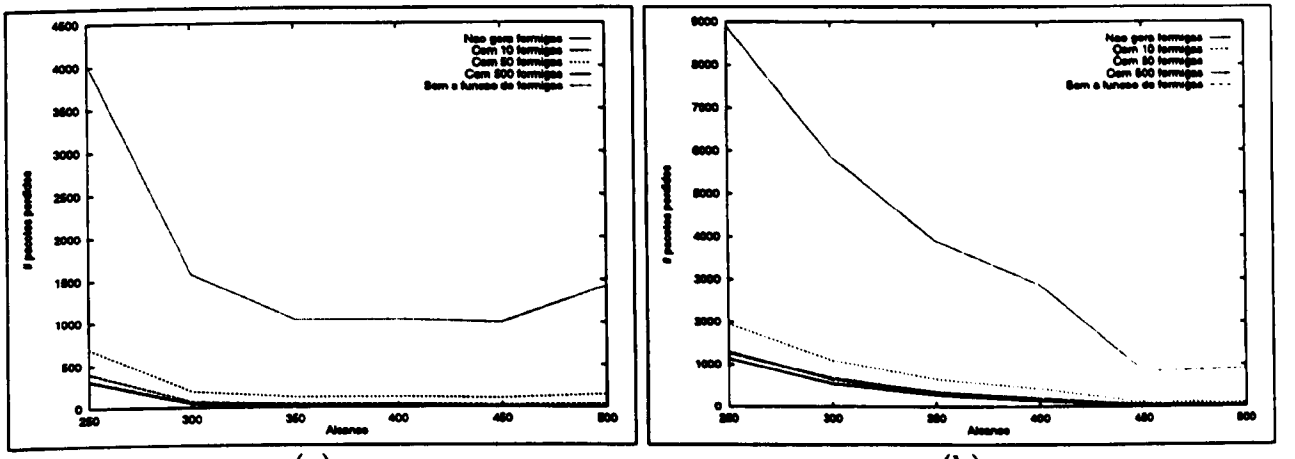


(a) (b)

Figura 8.4: Comparação entre a relação do número médio de pacotes trafegando na rede e o alcance dos nodos

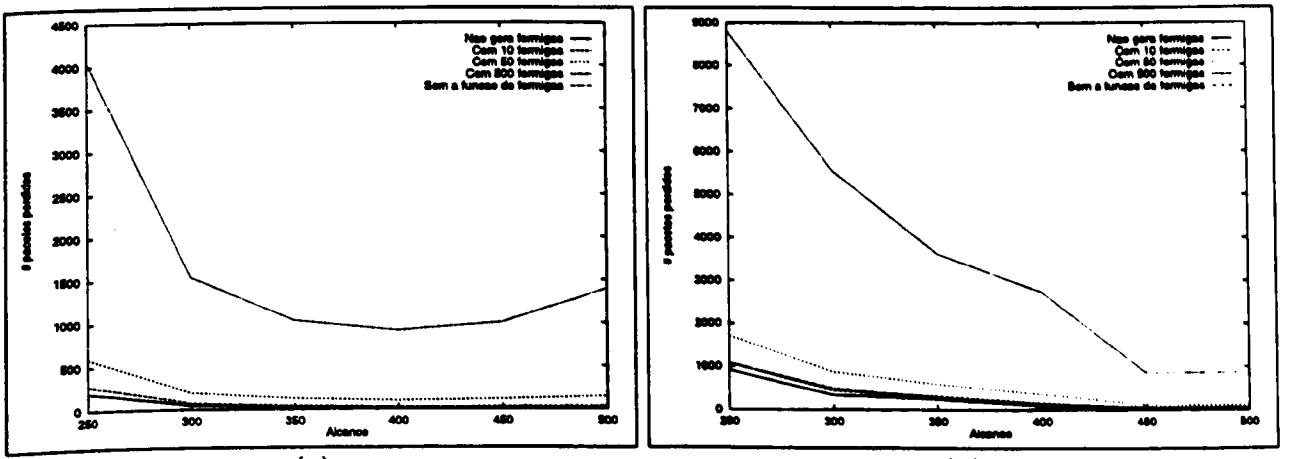
Os gráficos da figura 8.6 apresentam em (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, mostram o número total de pacotes perdidos pelo algoritmo com relação à variação do alcance dos nodos. Como pode-se observar, quanto maior o alcance menor é o número médio de pacotes perdidos. Neste experimento está se levando em consideração a perda de todos os tipos de pacotes. Deve-se observar que o comportamento do algoritmo que tem a menor velocidade, figura 8.6 (a), é melhor que o que tem maior velocidade, figura 8.6 (b). Isto se deve ao fato de que quanto menor a velocidade dos nodos, melhor é a qualidade das informações de roteamento como um todo, que trafegam na rede. O comportamento

dos dois conjuntos de testes, os da figura 8.6 (a) e (b) e da figura 8.5, onde tem-se em (a) cenários com nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, são idênticos. O uso da rede fixa diminuiu a perda de pacotes e o fez de forma uniforme em todas as variações de alcance testadas. O resultado do experimento é o esperado: quanto maior a quantidade de formigas, maior é a perda de pacotes. Este resultado é esperado por que o GPSAL considera as formigas como *overhead*, sendo assim nem todas as possibilidades são tentadas para enviar o pacote até o destino. Se um pacote de dados for perdido, não é um problema tão grande, ao contrário dos pacotes de dados enviados por datagrama que têm um tratamento do tipo *Best Effort*, onde se tenta entregar o pacote de todas as formas possíveis.



(a) (b)

Figura 8.5: Comparação entre a relação do número total de pacotes perdidos e o alcance dos nodos



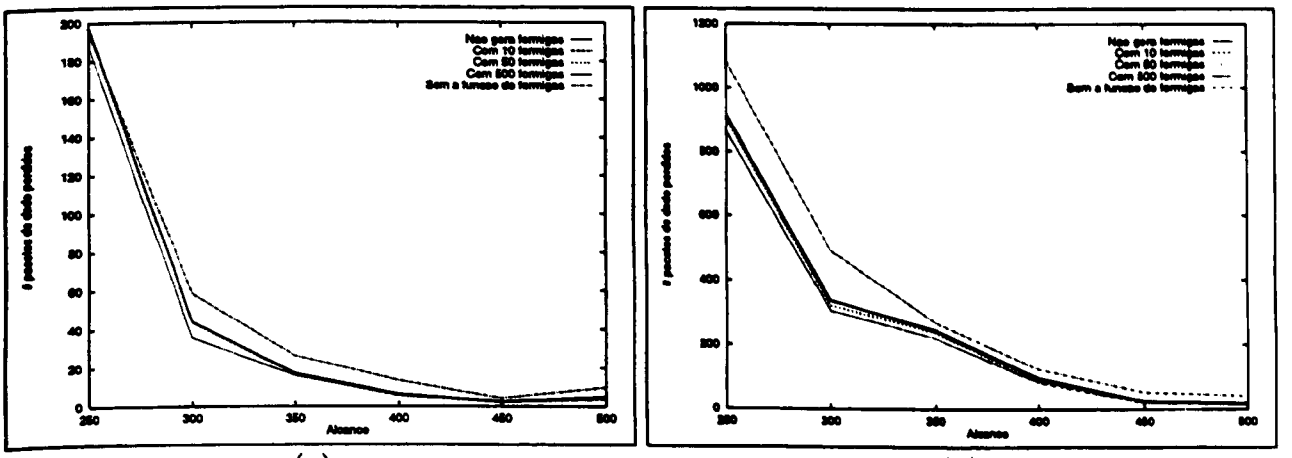
(a) (b)

Figura 8.6: Comparação entre a relação do número total de pacotes perdidos e o alcance dos nodos

Na figura 8.7, onde tem-se em (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, os gráficos mostram o número de pacotes de dados perdidos pelo GPSAL com relação a variação

do alcance dos nodos. O comportamento dos experimentos variando o número de formigas tiveram nesta bateria de testes um comportamento mais homogêneo. Há uma clara diminuição no número de pacotes de dados perdidos com o aumento do alcance. Quando se aumenta o alcance da comunicação, menos nodos são necessários para entregar as mensagens, e mais rápida e precisamente a informação de roteamento caminha pela rede. Melhorando a qualidade das informações de roteamento, melhoram-se as rotas criadas pelo GPSAL e diminui-se a probabilidade do pacote se perder na rede. Pode-se observar ainda que o gráfico da figura 8.7 (a), com menor velocidade, teve um comportamento melhor que o gráfico da figura 8.7 (b), com maior velocidade. Novamente a explicação para este comportamento é a qualidade das informações de roteamento. Quanto maior a velocidade, mais complicada e difícil é a atualização dos dados dos nodos.

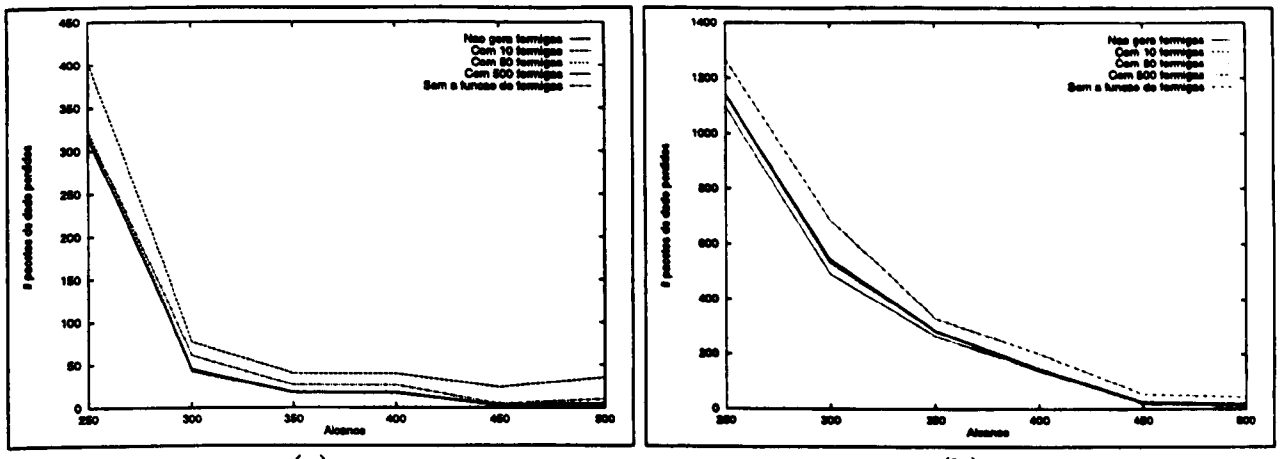
Novamente o comportamento dos dois testes, sem o uso da rede fixa (figura 8.7), e o caso da rede fixa (figura 8.8), onde tem-se em (a) cenários com nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, tiveram um comportamento similar. Entretanto a diferença desta vez foi mais visível e, novamente, o cenário que usa a rede fixa teve um melhor desempenho.



(a) (b)

Figura 8.7: Comparação entre a relação do número de pacotes de dados perdidos e o alcance dos nodos

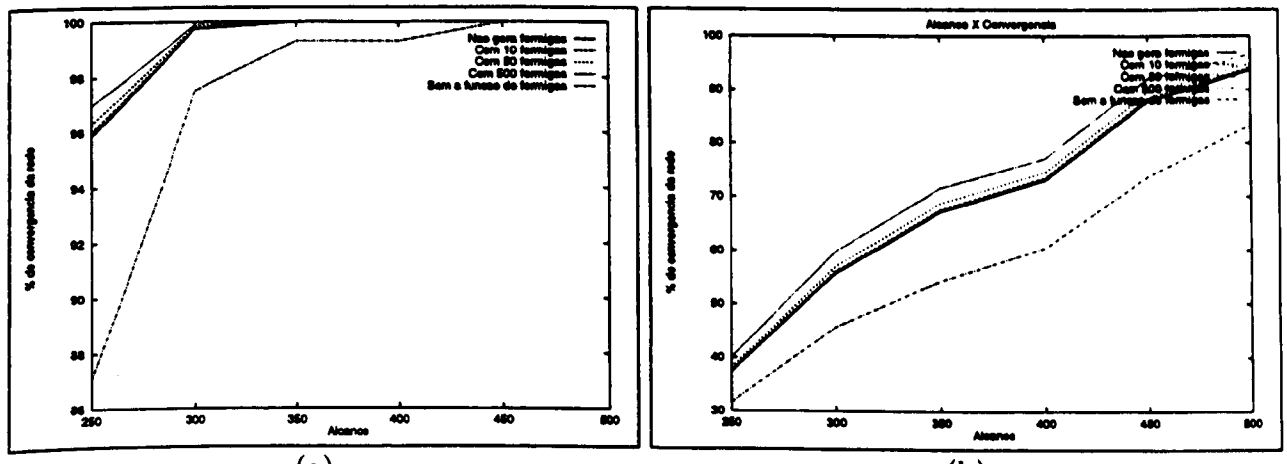
Os gráficos da figura 8.9, apresentam em (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, apresentam experimentos do GPSAL comparando a convergência média dos nodos da rede e o alcance. Na figura 8.9 (a) pode-se acompanhar a convergência do GPSAL em cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e na figura 8.9 (b) tem-se experimentos sem nodos fixos e com velocidade média de 25 unidades/iteração. Como se pode observar o alcance teve uma grande influência no comportamento do algoritmo. Quanto maior o alcance, maior a convergência média da rede. Isto ocorre pelo fato do algoritmo obter informações mais rapidamente e, por conseqüência, mais precisas. Outro fator determinante na convergência é a velocidade. Como se pode ver no gráfico da figura 8.9 (a), onde a velocidade é menor, tem-se uma maior convergência que no gráfico da figura 8.9 (b). Isto é esperado pois com o aumento da velocidade o nodo se desloca mais, piorando assim o conhecimento dos nodos sobre a rede. O comportamento das curvas foi



(a) (b)

Figura 8.8: Comparação entre a relação do número de pacotes de dados perdidos e o alcance dos nodos

também o esperado; os nodos que tinham mais formigas e por conseguinte mais informações sobre a rede, tiveram uma convergência melhor. Os gráficos da figura 8.10, onde tem-se em (a) cenários com nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, e que representam os mesmos experimentos considerando-se nodos fixos tiveram, um comportamento semelhante, só com uma convergência levemente superior, mostrando que a rede fixa não representa uma grande diferença na convergência do GPSAL, quando variamos o alcance.



(a) (b)

Figura 8.9: Comparação entre a convergência e o alcance dos nodos

Os gráficos da figura 8.11, onde tem-se em (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração, têm o intuito de caracterizar a variação do tamanho do caminho no GPSAL levando-se em consideração a variação do alcance. Estes gráficos apresentam um conjunto de informações muito interessantes. Podemos observar que o tamanho médio do caminho diminui sensivelmente com o aumento do alcance dos nodos. No gráfico da figura 8.11 (a), quando se tem um alcance de 250, o tamanho médio dos caminhos é de 3,4 nodos. Ao se aumentar o

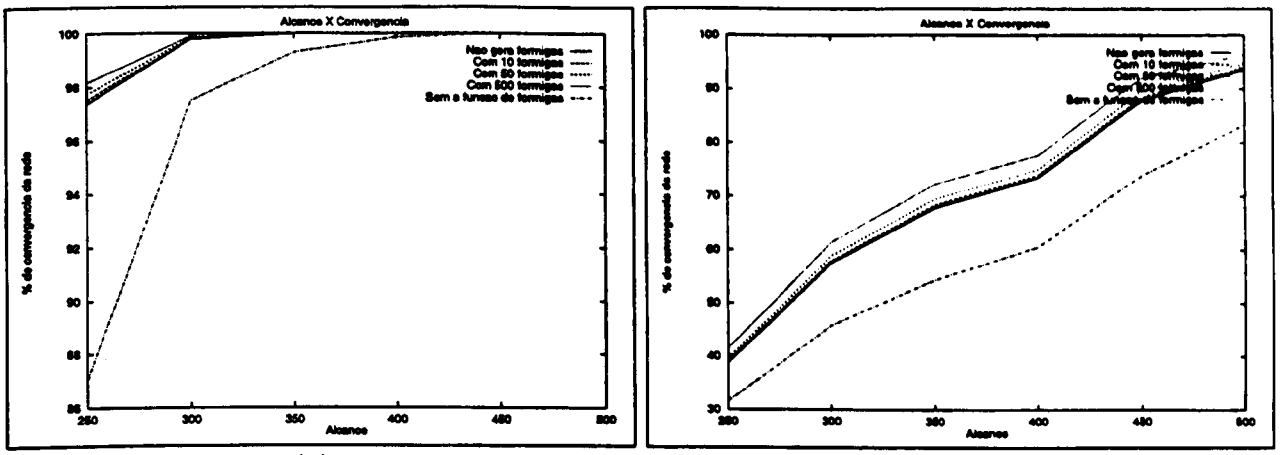


Figura 8.10: Comparação entre a convergência e o alcance dos nodos

alcance, este número médio diminui gradualmente até 1,7 nodos em média, com alcances a partir de 450. Isto fica claro quando se observa que os experimentos ocorrem em um cenário de 1000×1000. Assim sendo, o alcance representa por volta da metade da largura ou altura da área, sendo relativamente grande. Assim, em poucos casos serão necessários nodos para servirem de pontes para a comunicação. No gráfico da figura 8.11 (b), que apresenta velocidades médias de 25 unidades/iteração, tem-se um comportamento semelhante, mas a convergência não é tão rápida. Isto se deve aos erros que a mobilidade insere nas tabelas de roteamento. Como foi observado nos gráficos da figura 8.9, os experimentos em que os nodos tinham maior velocidade tiveram uma convergência média da rede pior que os experimentos com menores velocidades. Isto significa que suas tabelas estavam mais desatualizadas com relação aos experimentos com menor velocidade. Por conseqüência, as rotas não eram tão precisas acarretando não só este pequeno *overhead* no tamanho médio dos caminhos, mas também uma maior perda de pacotes, tanto de formigas da figura 8.6 quanto de dados, da figura 8.7.

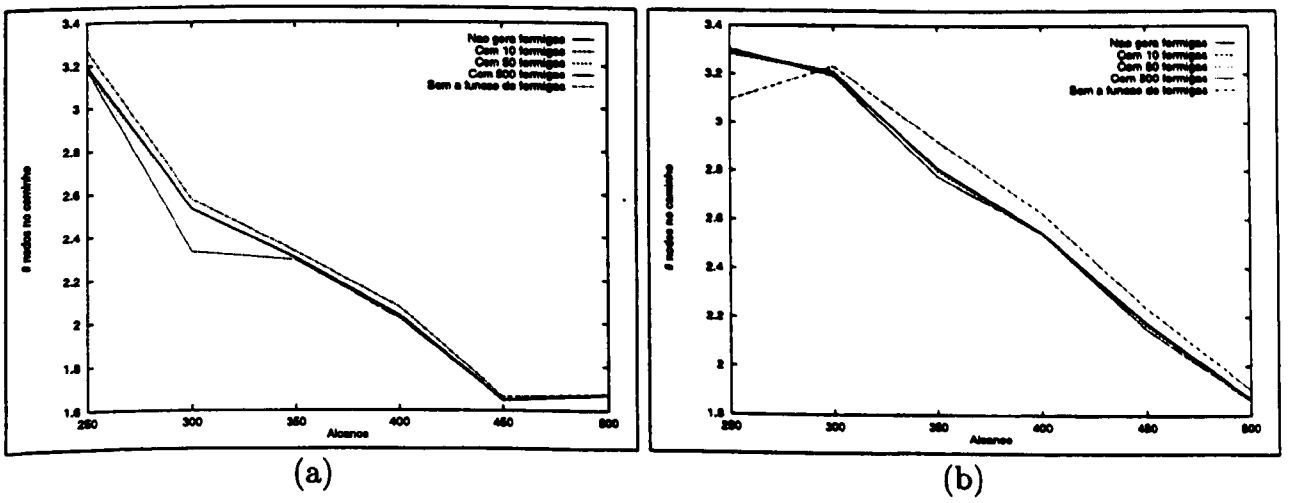
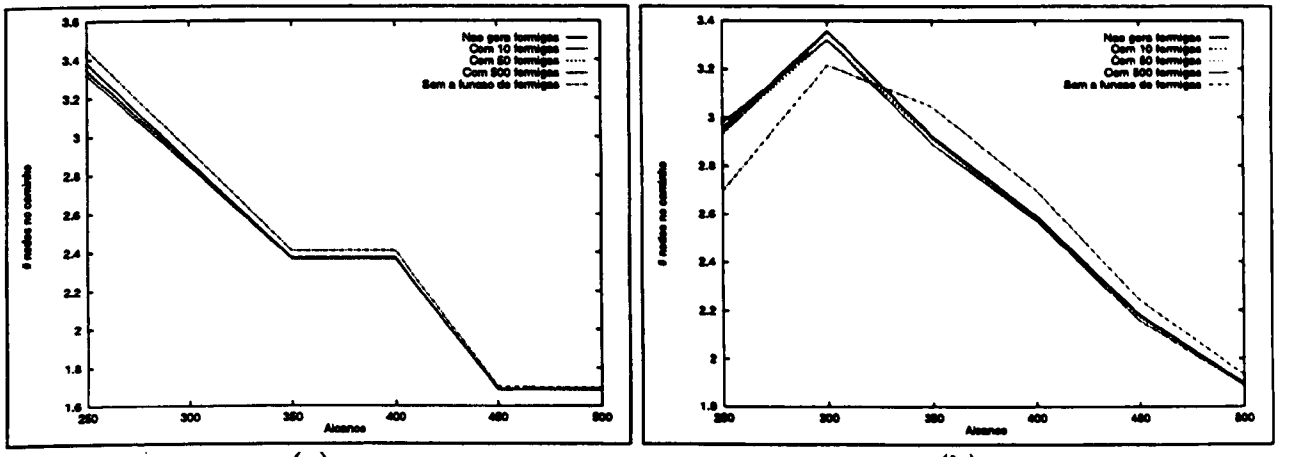


Figura 8.11: Comparação entre o tamanho médio dos caminhos e o alcance dos nodos

Novamente a rede fixa teve pouca influência no tamanho médio do caminho, como podemos ver nos gráficos da figura 8.12, onde tem-se em (a) cenários com nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com velocidade média de 25 unidades/iteração. A rede fixa apresentou uma melhora geral nos dados, diminuindo o número médio de nodos intermediários nas comunicações, mas este ganho existindo, não foi expressivo. Isto se deve provavelmente ao fato da rede ser relativamente pequena. Acredita-se que com o aumento do tamanho da rede, a possibilidade de uso da rede fixa melhore em muito o desempenho dos algoritmos de roteamento para redes ad hoc.



(a) (b)

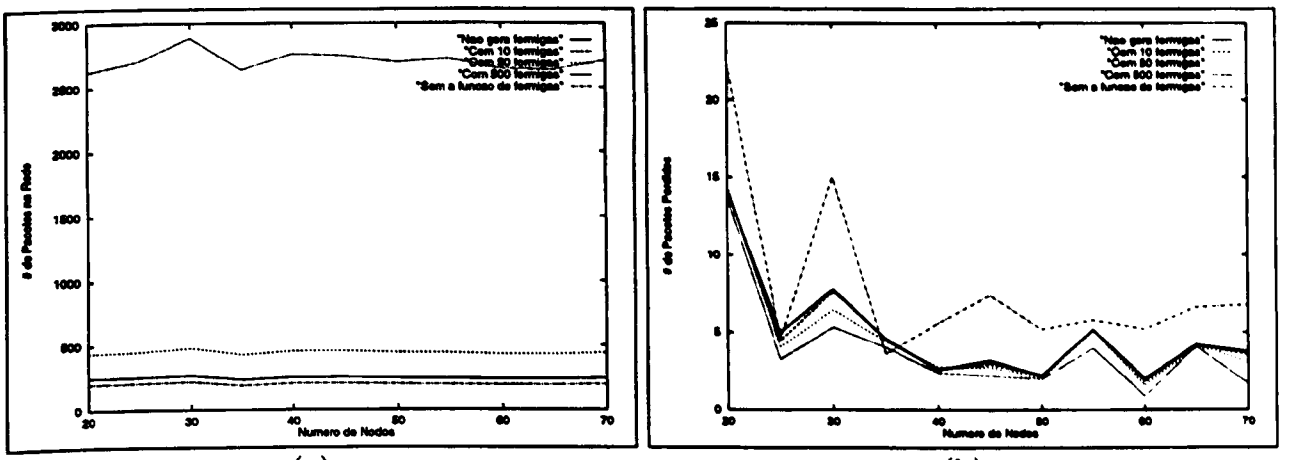
Figura 8.12: Comparação entre o tamanho médio dos caminhos e o alcance dos nodos

8.5.2 Número de Nodos na Rede

O objetivo destes testes é caracterizar o comportamento do GPSAL com relação ao número de nodos na rede. Serão analisados o número de pacotes na rede, o número de pacotes de dados perdidos, a convergência da rede e o tamanho dos caminhos.

Os gráficos da figura 8.13, apresentam em (a) cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e em (b) com nodos fixos e velocidade média de 5 unidades/iteração. Os gráficos mostram o número médio de pacotes trafegando na rede e o número de pacotes de dados perdido pelo GPSAL de acordo com a variação do número de nodos da rede. Pode-se observar na figura 8.13 (a) como o número de pacotes na rede não apresenta uma variação significativa com relação à variação do número de nodos da rede. O que se pode notar claramente é a variação do número de pacotes com relação ao número de formigas, mas isto é esperado, pois quanto maior o número de formigas maior o número de pacotes na rede. Já na figura 8.13 (b) tem-se um comportamento mais interessante, que é a perda de pacotes de dados com relação ao número de pacotes da rede. Pode-se notar que o número de pacotes de dados perdidos diminui significativamente com o aumento do número de nodos. Isto se deve ao fato de haverem mais opções de rota até o destino, sendo estas opções de rotas fundamentais no reroteamento de pacotes a partir dos nodos intermediários. Caso o pacote chegue em um "beco sem saída", é fundamental que um nodo que não tenha como repassar a mensagem pela rota previamente descrita,

tenha opções na criação de uma nova rota para tentar enviar o pacote até o destino. Caso o nodo não tenha estas opções ele é obrigado a descartar o pacote, aumentando o número de pacotes de dados perdidos. É importante ressaltar a escala do gráfico da figura 8.13 (b), sendo que, comparativamente com o número de pacotes na rede, o número de pacotes de dados perdidos é pequeno.

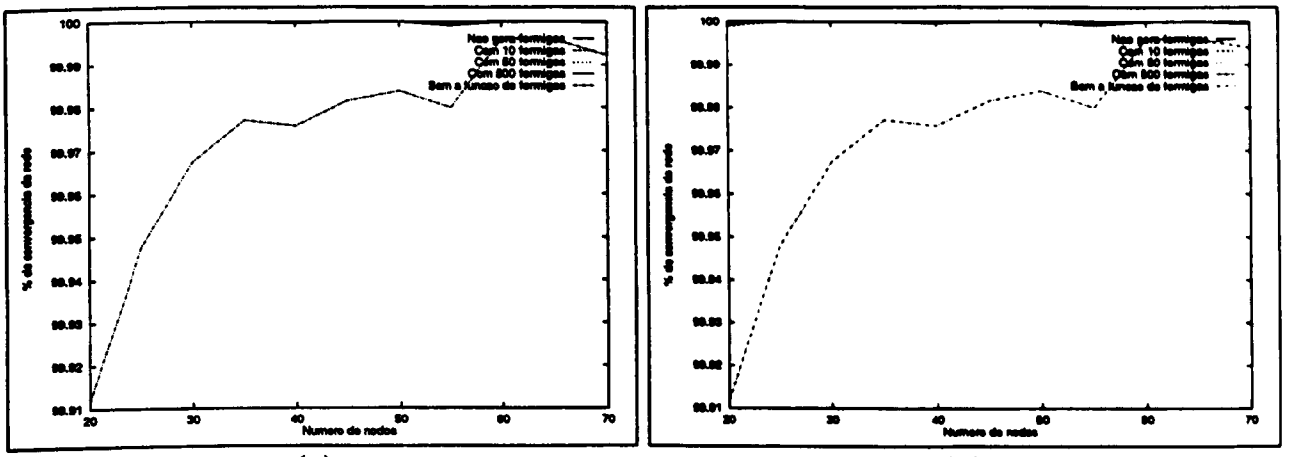


(a) (b)

Figura 8.13: Comparação entre a convergência e a variação do número de nodos

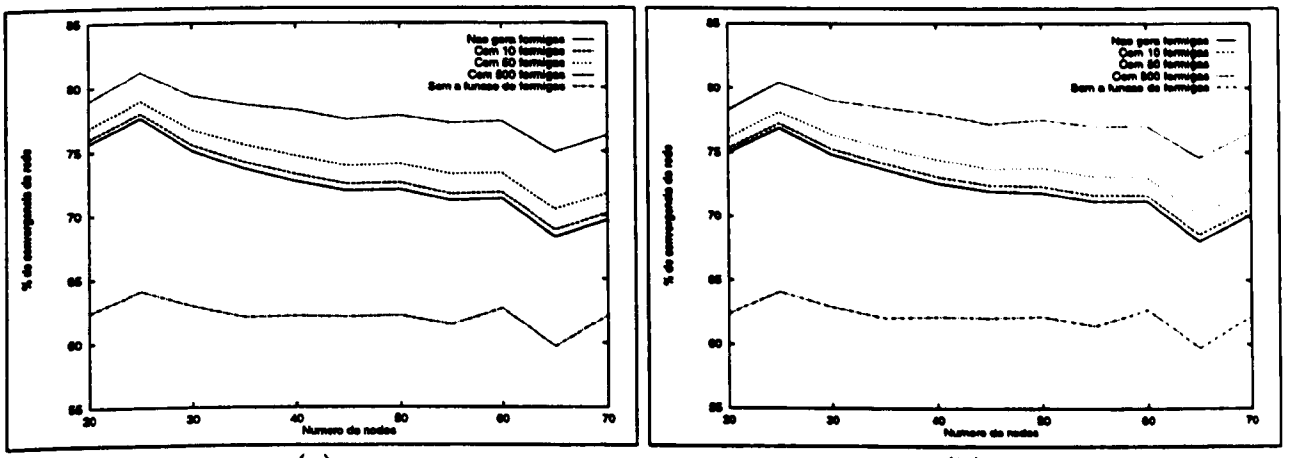
Na figura 8.14, onde se tem dois gráficos onde (a) representa cenários sem nodos fixos e com velocidade média de 5 unidades/iteração e (b) com nodos fixos e velocidade média de 5 unidades/iteração, os dois gráficos apresentam a convergência da rede e o número de nodos. Pode-se notar a alta convergência de todas as modalidades do GPSAL, com exceção da versão sem formiga e sem os pacotes de dados trabalhando como formigas. Isto se deve à baixa mobilidade dos nodos, O aumento de nodos, quando a mobilidade é baixa, só ajuda a convergência do algoritmo, pois mais informações sobre a rede estarão trafegando, melhorando assim a convergência da rede. Além disto, deve-se chamar a atenção para o fato de a inclusão da rede fixa não apresentar uma grande variação na convergência. A diferença entre os dois experimentos foi mínima. Há um pequeno ganho quando se usa a rede fixa no gráfico da figura 8.14 (b), com relação ao que não utiliza a rede fixa no gráfico da figura 8.14 (a), mas a diferença foi apenas um pequeno deslocamento dos pontos e que no gráfico quase não pode ser notado.

A figura 8.15 apresenta outros dois gráficos relacionando a convergência da rede e o número de nodos, onde em (a) tem-se cenários sem nodos fixos e com velocidade média de 25 unidades/iteração e em (b) com nodos fixos e velocidade média de 25 unidades/iteração. Pode-se notar que novamente a rede fixa, que é usada no gráfico da figura 8.15 (b), não representa um ganho significativo com relação ao cenário da figura 8.15 (a) onde não se usa a rede fixa. Mais uma vez, houve apenas um deslocamento dos pontos, agora um pouco mais visível, a favor do gráfico da figura 8.15 (b) onde se utiliza a rede fixa. O resultado da convergência também foi esperado, os experimentos que utilizaram mais formigas tiveram uma melhor convergência. Mas mesmo o experimento com pior convergência, que foi o GPSAL sem formiga e sem os pacotes de dados agindo como formigas, sabia a posição exata de mais de 60% da rede para a quase totalidade dos pontos do gráfico.



(a) (b)

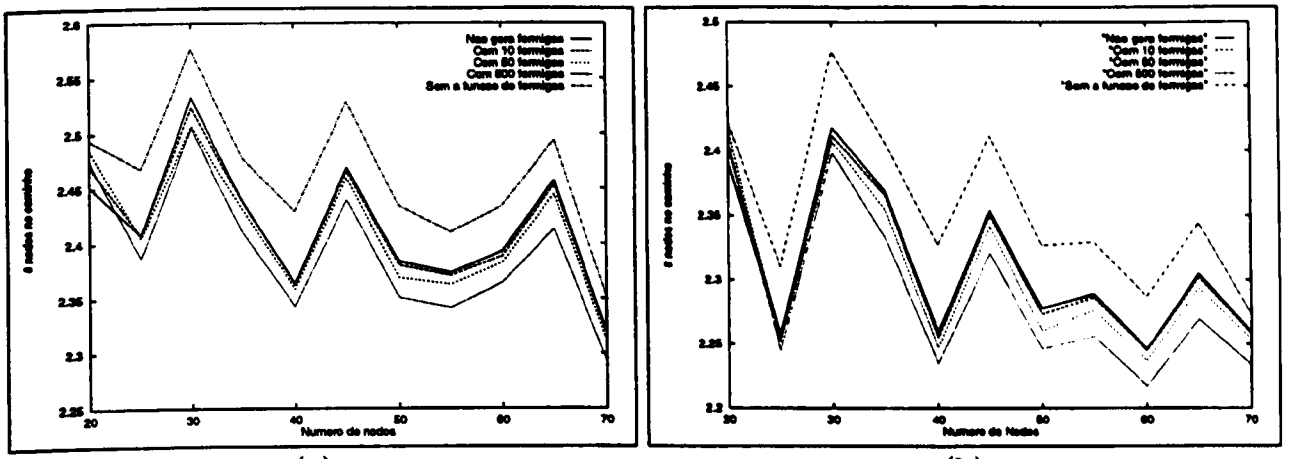
Figura 8.14: Comparação entre a convergência e a variação do número de nodos



(a) (b)

Figura 8.15: Comparação entre a convergência e a variação do número de nodos

Nos gráficos da figura 8.16, onde tem-se em (a) cenários sem nodos fixos e com velocidade média de 25 unidades/iteração e em (b) com nodos fixos e velocidade média de 25 unidades/iteração, são apresentadas comparações com relação à variação do tamanho dos caminhos e o número de nodos da rede. Os dois gráficos têm um comportamento semelhante, tanto o da figura 8.16 (a) que representa a rede sem nodos fixos, quanto o da figura 8.16 (b) que representa a rede com nodos fixos. Pode-se apenas notar que a figura 8.16 (b) apresenta um número médio de nodos no caminho levemente inferior ao gráfico da figura 8.15 (a) onde tem-se apenas nodos móveis. As curvas nos dois gráficos mostram que há uma tendência em diminuir o tamanho dos caminhos com o aumento do número de nodos. Isto pode ser explicado devido ao fato de os experimentos serem aleatórios e com o aumento do tamanho do número de nodos na rede, há uma maior possibilidade de os nodos destino estarem mais próximos à origem.



(a) (b)

Figura 8.16: Comparação entre o tamanho do caminho e a variação do número de nodos

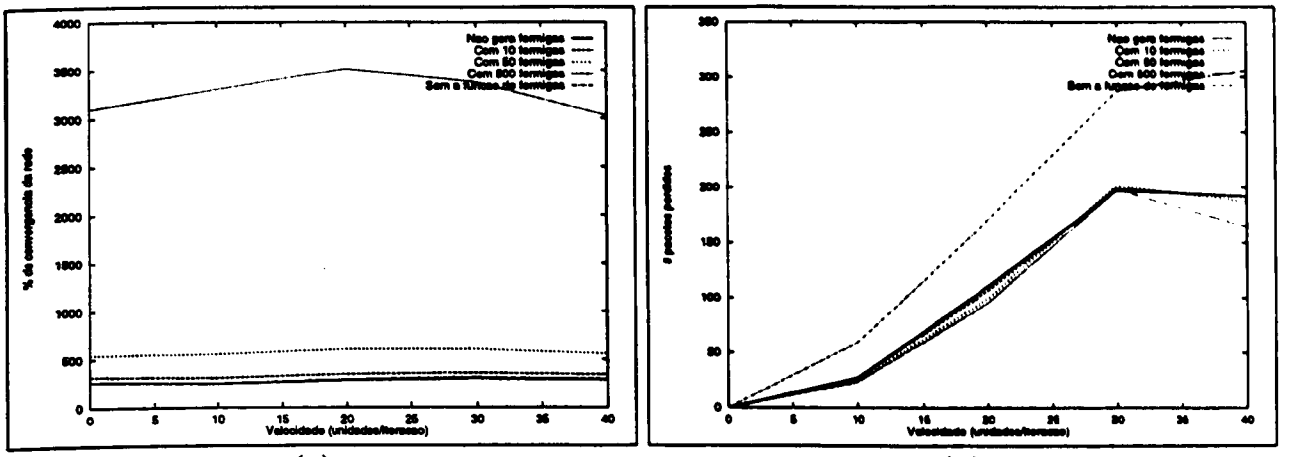
8.5.3 Velocidade dos Nodos

O objetivo destes testes é avaliar o comportamento do GPSAL com relação à velocidade dos nodos que participam da rede. Serão analisados o número de pacotes na rede, o número de pacotes de dados perdidos e a convergência da rede.

Os gráficos da figura 8.17, apresentam em (a) o número de pacotes na rede e em (b) o número de pacotes de dados perdidos, considerando o número médio de pacotes trafegando na rede e o número de pacotes de dados perdido pelo GPSAL de acordo com a variação da velocidade dos nodos da rede. No gráfico da figura 8.17 (a), que apresenta o número de pacotes trafegando na rede com o aumento da velocidade dos nodos, pode-se observar que a velocidade dos nodos não apresenta um grande impacto no número de pacotes trafegando na rede. A não ser no experimento onde não havia pacotes de formiga e nem os pacotes de dados agindo como formigas. O comportamento da curva pode ser explicado pelo grande número e tipo de mensagens trafegando na rede. As formigas são pacotes do algoritmo de roteamento, portanto não são consideradas informações úteis, já que só informações de dados são consideradas úteis. Sendo assim, as mensagens de formiga podem ser descartadas mais facilmente que pacotes de dados. Quando a velocidade é baixa, o número de pacotes é baixo pois os pacotes formiga vão e voltam do seu destino rapidamente, não ficam "vagando" atrás do destino, como é o caso quando se aumenta a velocidade dos nodos. O decaimento da curva quando há um aumento de mais de 25 unidades/iteração, se deve ao descarte de pacotes de formiga. Quanto maior a velocidade, mais difícil é manter as tabelas de roteamento atualizadas, e isto pode levar a rotas incorretas. Para pacotes de dados todos os dispositivos são tentados, mas se o pacote for de formiga, caso o reroteamento seja muito caro, esta formiga deve ser consumida pelo nodo.

O gráfico da figura 8.17 (a) mostra o volume de dados perdidos durante as simulações. Pode-se observar que o comportamento dos experimentos foi idêntico. O que apresentou maior perda de pacotes de dados foi exatamente o que tem a pior convergência, como pode ser visto no gráfico da figura 8.18.

O gráfico da figura 8.18 retrata a convergência da rede com relação à variação da velocidade dos nodos. Os experimentos com e sem o uso da rede fixa apresentaram resultados



(a) (b)

Figura 8.17: Comparação com relação à velocidade dos nodos

similares neste teste. Pode-se observar que o aumento da velocidade tem um impacto ruim na convergência do GPSAL. Quanto menor a velocidade dos nodos, mais fácil e precisas são as atualizações de tabelas, portanto melhor a qualidade das informações trocadas. Isto tem uma influência positiva na convergência, ao contrário do aumento da velocidade que torna mais difícil a atualização dos dados e como conseqüência acarreta em rotas maiores e menos precisas. O GPSAL sem formigas e sem pacotes de dados trabalhando como formigas teve uma pior convergência, pois as atualizações de suas tabelas de roteamento eram feitas apenas através das trocas de tabelas. Já o comportamento dos outros experimentos foi o esperado. Deve-se ressaltar que para velocidades até 10 unidades/iteração, o algoritmo teve uma convergência perto de 100%. Este fato explica, e confirma, os dados obtidos nos gráficos da figura 8.14, onde a velocidade foi baixa e os experimentos tiveram uma convergência de quase 100%.

A figura 8.19 apresenta um gráfico que mostra os tamanhos médios dos caminhos, com relação à velocidade dos nodos. Como já era esperado, com o aumento do tamanho da velocidade, tem-se uma pior convergência. Isto causa uma piora na qualidade do roteamento e um aumento do tamanho dos caminhos, decorrente desta imprecisão nas informações armazenadas nas tabelas. Novamente o comportamento das linhas foi similar, exceto pelo experimento que tem menos informações sobre a rede, que teve por conseqüência um maior número médio de nodos no caminho.

8.6 Comparação com o LAR

O objetivo desta seção é comparar o desempenho do GPSAL com um outro algoritmo geográfico de roteamento, para redes móveis ad hoc. O algoritmo escolhido foi o LAR (*Location-Aided Routing*), que foi apresentado no capítulo 6. O modelo de simulação para os testes seguintes, caso não seja explicitamente dito o contrário, são:

- Alcance: 300 *unidades*
- Número de Nodos: 30

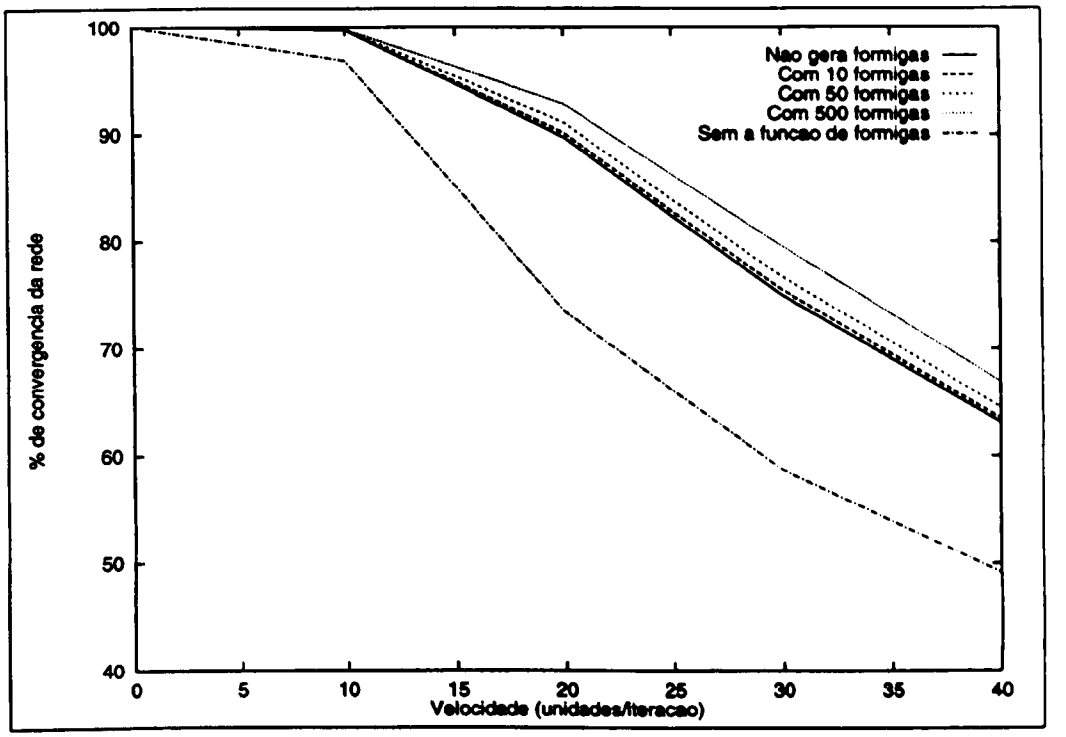


Figura 8.18: Comparação entre a convergência e a velocidade dos nodos.

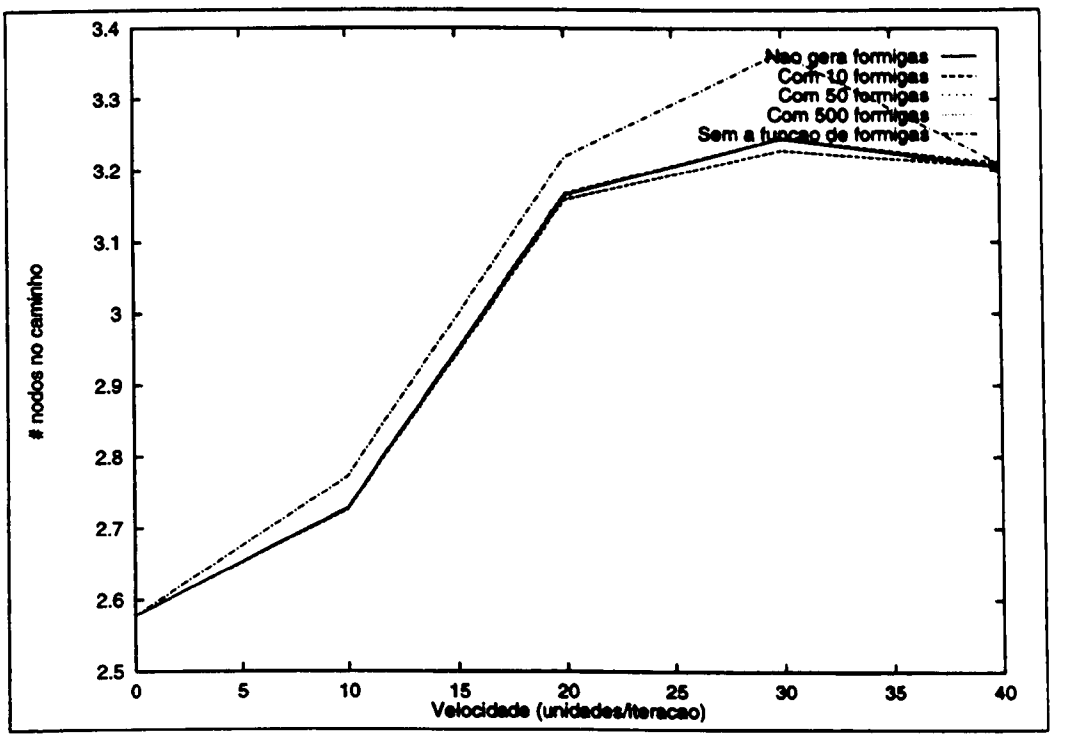


Figura 8.19: Comparação entre o tamanho do caminho e a velocidade dos nodos

- Tamanho do cenário: 1000 × 1000 *unidades*
- *Timeout* para o LAR: o suficiente para percorrer a diagonal principal, mais duas unidades. Por exemplo, com o alcance de 300 sendo o cenário de 1000 × 1000, o *timeout* é de 7

- Pacotes gerados: 10, sempre de várias origens para um mesmo destino
- Probabilidade de se gerar uma formiga: 0,5 por nodo a cada iteração

8.6.1 O Tratamento dos Pacotes no Simulador

O simulador deixa de contabilizar os pacotes do GPSAL assim que todos são entregues, e no LAR assim que o último pacote for descartado por *timeout*. No LAR são contabilizados apenas os pacotes resultantes dos pacotes de dados iniciais, não foi implementada a parte inicial do descobrimento da posição do nodo, que no caso do LAR é um *flooding* puro. O simulador fornece ao LAR a posição exata do nodo destino, sendo que desta forma o *flooding* inicial não é necessário. Para calcular a área de requisição de rotas é calculado a área onde se espera encontrar o nodo, com base na sua velocidade, e sobre esta área a zona de requisição. Para calcular o tempo que esta mensagem vai levar para chegar ao destino é encontrado o melhor caminho, por um algoritmo de menor caminho, obtendo assim o número de nodos intermediários necessários para encontrar o nodo. Dobramos este número, pois o pacote de requisição inicial precisaria ir e voltar. Esta parte fornece o tempo, em envio de pacotes, gasto pela requisição. Multiplica-se este valor pela mobilidade do nodo encontrando-se assim a zona de requisição. Para o GPSAL é considerado como tráfego, o número de trocas de tabelas da rede como um todo, pacotes de dados formigas e trocas de tabelas, mesmo dos nodos que eventualmente não estejam participando efetivamente da comunicação de pacotes de dados nesta simulação.

8.6.2 Testes

Nestes experimentos, foram tomados como base os experimentos gerados para o algoritmo LAR em [41]. Foram escolhidas 10 origens aleatórias e a partir de cada uma destas origens foram enviados pacotes a um único destino. Para cada um dos pontos obtidos nos gráficos foram criados 4 cenários de movimentação, onde são escolhidas de forma aleatória 10 origens e um único destino. Cada cenário é executado 32 vezes, sendo um total de 128 execuções. Os experimento foi analisado com um corte de 5 percentil, e um intervalo de confiança de 95%.

Variação do Alcance

A primeira série de testes tem por objetivo caracterizar o comportamento do algoritmo, com relação à variação de alcance dos nodos. Podemos observar na figura 8.20 que quanto maior o alcance maior o número de nodos que recebem a comunicação do *flooding* no LAR. Nem todos os nodos enviam resposta deste *flooding*, por não estarem na área de requisição de rotas, mas gastam energia recebendo e processando o pacote da mesma forma. O GPSAL não apresenta este problema pois quanto maior o alcance, mais rápido o pacote irá chegar ao destino, e menos pacotes estarão trafegando na rede. Por este motivo a curva do LAR é ascendente e do GPSAL é descendente no gráfico. Na figura 8.21 temos o que parece uma inconsistência com o gráfico anterior, pois a curva se apresenta mais comportada e com valores, em média, maiores que o gráfico da figura 8.20. O motivo

deste comportamento é, além de os cenários serem criados aleatoriamente, o gráfico da figura 8.20 representa uma rede com velocidade média de 25 unidades/iteração, e o gráfico da figura 8.21 representa uma rede com velocidade média de 4.5 unidades/iteração. Desta maneira, como a movimentação é pequena, a rede como um todo é mais comportada. Na figura 8.20 tem-se uma grande perda de pacotes no início, devido a grande mobilidade dos nodos, e no final tem-se um número maior de pacotes na rede que o gráfico da figura 8.21 Isto porque devido a movimentação, a zona de requisição é maior que a dos nodos do cenário com mobilidade média de 4, 5 unidades/iteração. Mas o mais importante é observar que nos dois cenários, mesmo no LAR sendo considerado apenas os pacotes de dados e no GPSAL sendo considerados dados, formigas e todas as trocas de tabelas, o algoritmo apresentou, em todos os momentos, uma menor taxa de pacotes na rede.

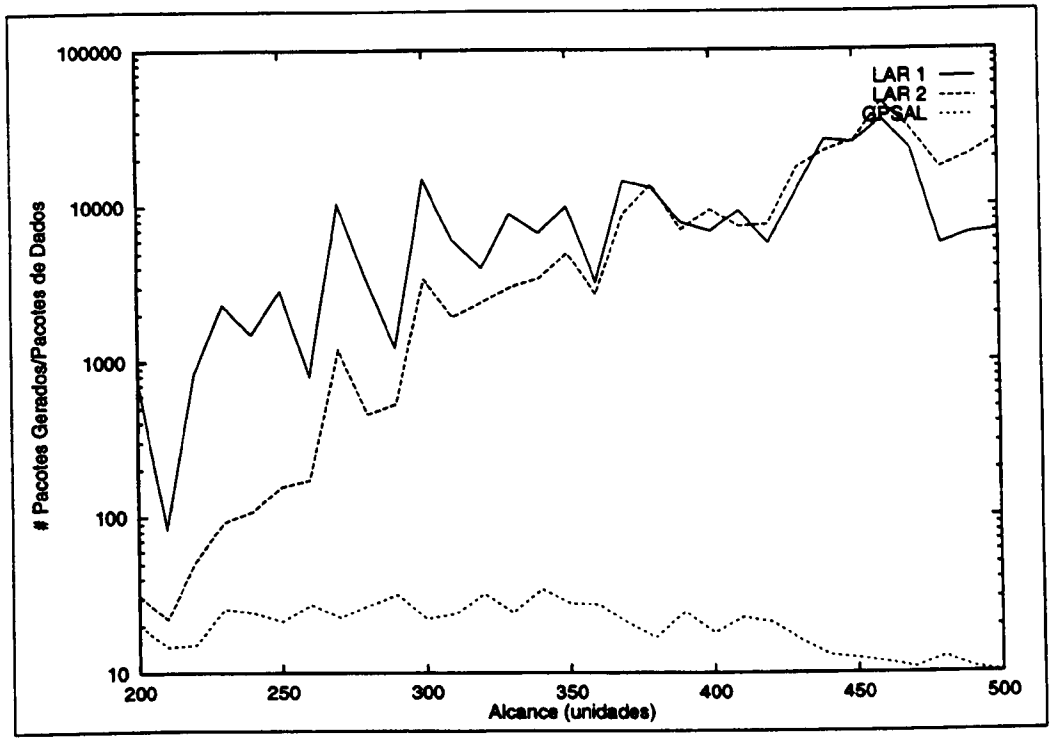


Figura 8.20: Comparação entre o número de pacotes gerados por pacotes de dados com a variação do alcance, sendo a velocidade média dos nodos de 25 unidades/iteração

Variação da Velocidade

O gráfico da figura 8.22 faz uma comparação entre o número de pacotes gerados por pacotes de dados e a variação da velocidade dos nodos. No LAR os nodos, com o aumento da velocidade, entram e sae em da zona de requisição mais freqüentemente, diminuindo o número de pacotes retransmitidos. O que acontece neste cenário, com relativa freqüência para o LAR é que o nodo está na área, recebe e contabiliza o pacote e no momento da retransmissão não está mais na área e não retransmite o pacote. Este processo diminui o número de mensagens na rede. A curva do GPSAL apresenta uma pequena elevação devido ao fato de que, com o aumento da velocidade, as informações das tabelas de roteamento apresentam um erro maior, acarretando em um erro no descobrimento das rotas e tornando

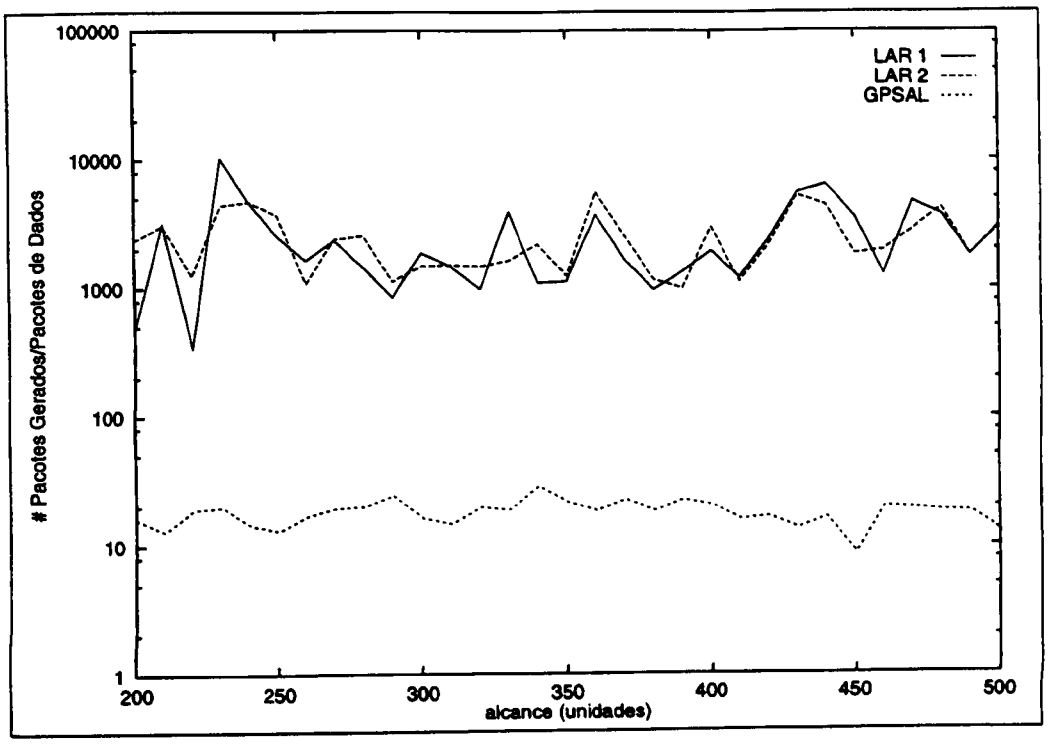


Figura 8.21: Comparação entre o número de pacotes gerados por pacotes de dados com a variação do alcance, sendo a velocidade média dos nodos de 4, 5 unidades/iteração

os caminhos maiores.

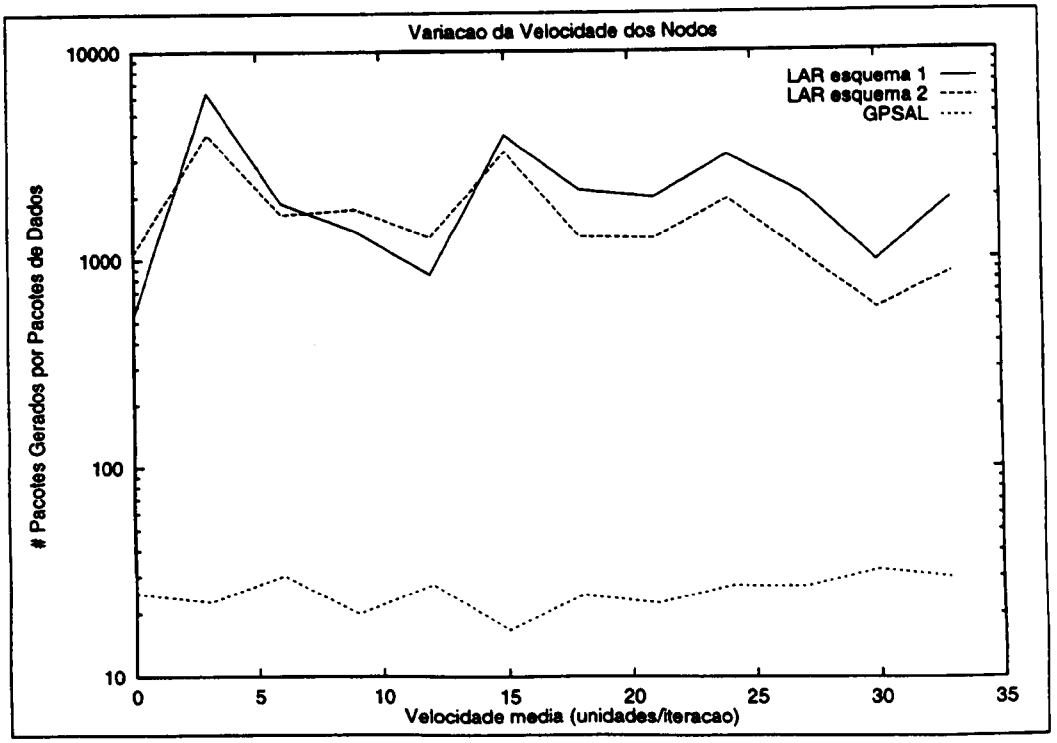


Figura 8.22: Comparação entre o número de pacotes gerados por pacotes de dados e a variação da velocidade

Variação do Número de Nodos

O gráficos das figuras 8.23 e 8.24 apresentam uma comparação entre o LAR e o GPSAL com relação ao número de pacotes gerados por pacotes transmitidos e o número de nodos participando da rede. No LAR, com o aumento do número de nodos, há um aumento do número de comunicações recebidas por nodos, participantes ou não da zona de requisição. O problema se agrava com a velocidade média sendo de 25 *unidades/segundo* pois a zona de requisição fica maior, envolvendo mais nodos. O GPSAL também apresentou um aumento no tráfego gerado, mas este foi referente principalmente às trocas de tabelas inerentes aos nodos e não à retransmissão dos pacotes de dados iniciais, como no LAR.

Em todos os experimentos o GPSAL apresentou um comportamento mais homogêneo que o LAR1 e LAR2. Este tipo de comportamento, mais que desejável, é essencial no momento de se prever e projetar o crescimento ou upgrade de equipamentos da rede. É importante salientar que os gráficos estão em escala logarítmica, o que diminui o impacto da diferença entre os números do GPSAL e do LAR. Outro ponto a ressaltar é que as informações nas trocas de tabelas são somente as atualizações que o GPSAL herda dos algoritmos do tipo *link state*, sendo assim os pacotes são pequenos, normalmente menores que pacotes típicos de dados.

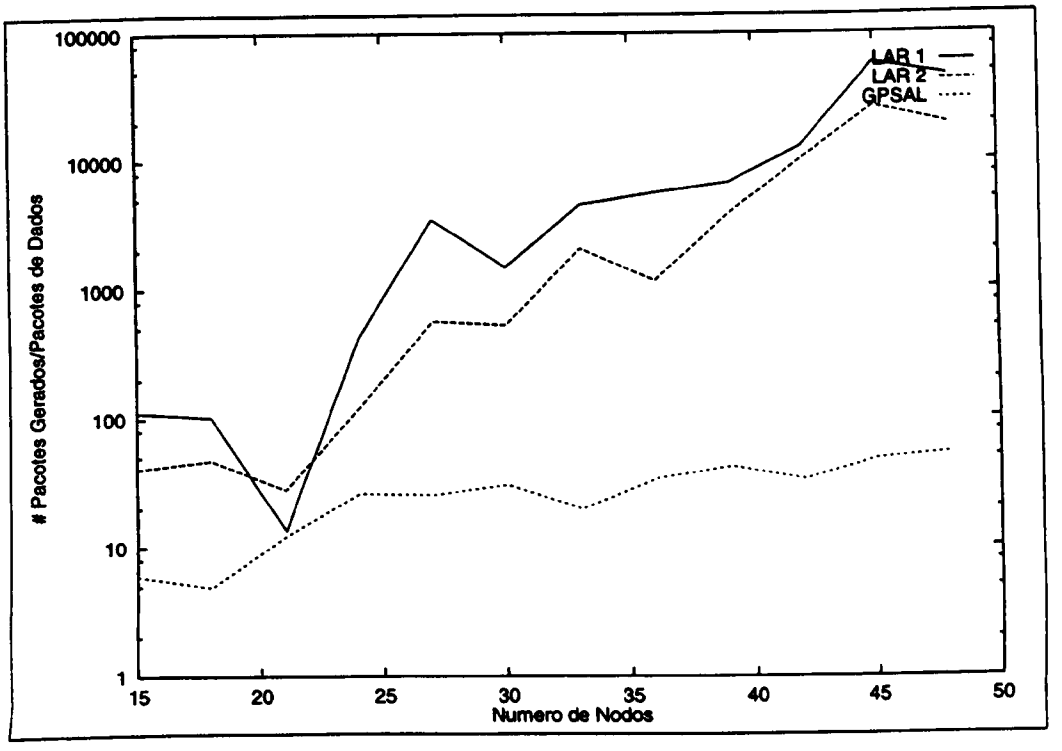


Figura 8.23: Comparação entre o número de *pacotes gerados por pacotes de dados* e a variação do número de nodos, para a velocidade média de 25 *unidades/segundo*

8.7 Comentários

Neste capítulo, vários aspectos do GPSAL foram testados oara se compreender melhor o comportamento do algoritmo. A análise dos dados das simulações forneceu um melhor

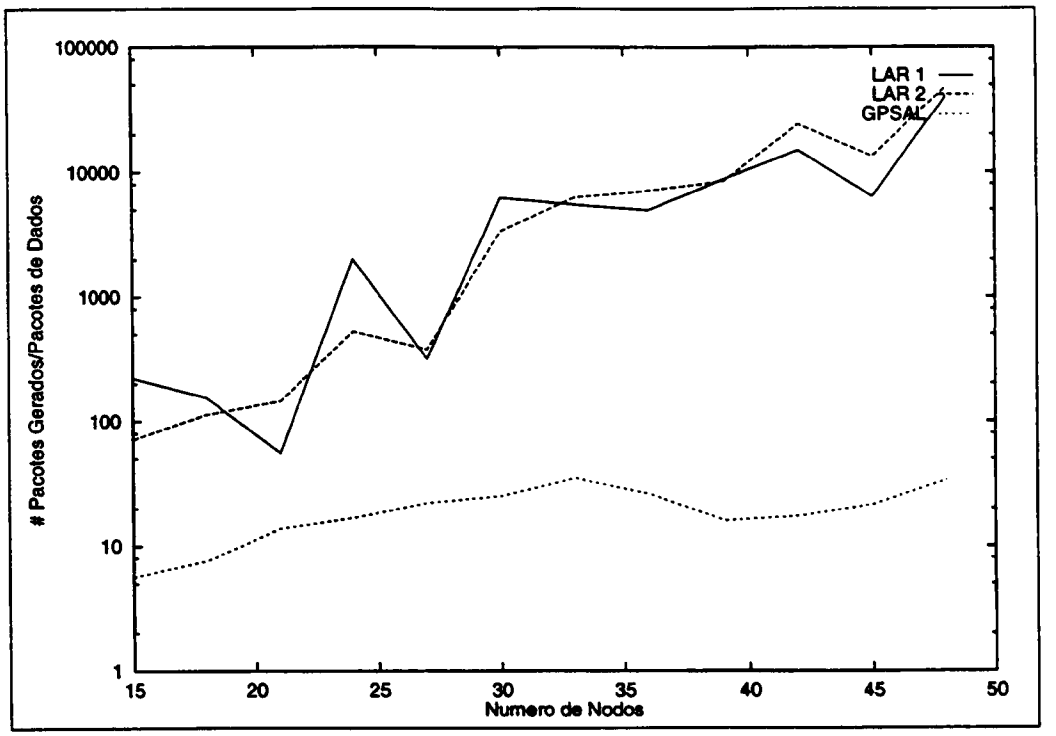


Figura 8.24: Comparação entre o número de pacotes gerados por pacotes de dados e a variação do número de nodos, para a velocidade média de 4.5*unidades/segundo*

entendimento do funcionamento do GPSAL e do LAR. Além disto possibilitou uma visão mais ampla da forma de funcionamento de protocolos de roteamento de forma geral. Vários outros parâmetros poderiam ser variados nestes experimentos e certamente trariam resultados interessantes, mas acredita-se que os experimentos aqui descritos dão uma boa visão do modo de funcionamento do GPSAL.

Capítulo 9

Comparação

O objetivo deste capítulo é apresentar uma comparação entre as principais características dos algoritmos de roteamento para redes ad hoc existentes. Serão apresentadas as características dos algoritmos, não sendo feitas críticas ou apontadas falhas. Entende-se que cada algoritmo tem suas próprias características e objetivos, sendo portanto difícil definir o que são falhas ou qualidades em um algoritmo. Características que para um objetivo podem ser consideradas como falha, para outro podem ser consideradas desejáveis. Um bom exemplo disto é o LAR. Ao observar que o algoritmo faz *flooding* do pacote de dados na rede, poderia-se considerar isto uma péssima característica, ou até mesmo uma falha de projeto. Mas se o objetivo é fazer um *multicast* da mensagem, esta característica pode ser muito útil. Sendo assim este capítulo se limita a mostrar as diferenças e fazer pequenos comentários, quando necessário.

9.1 Tabela de comparação

A tabela 9.1 apresenta uma comparação entre os algoritmos de roteamento para redes ad hoc discutidos anteriormente. Nessa comparação são consideradas algumas das mais importantes e desejáveis características de algoritmos de roteamento para redes ad hoc. Será apresentado em seguida um breve comentário sobre cada ponto apresentado na tabela, tentando mostrar a importância dos pontos e o porquê destes pontos estarem presentes nesta comparação.

- **Livre de *loops*** - É importante saber se o algoritmo é livre de loops. Loops de roteamento são casos em que a mensagem é enviada sempre entre os mesmos nodos, não saindo deste circuito e não alcançando o destino. Desta forma, o algoritmo está desperdiçando recursos da rede com uma mensagem "inútil", pois não irá alcançar o destino.

- **Múltiplas rotas** - Existem casos em que é interessante que o algoritmo possibilite a existência de mais de uma rota. Isto é especialmente interessante no caso de quebra de *link*, onde o caminho auxiliar pode ser utilizado como *backup*. Outro caso onde esta característica é interessante é para o envio de mensagens *multicast*.

- **Distribuído** - Saber se o algoritmo é distribuído ou centralizado é importante para saber qual é a carga de processamento atribuída a cada um dos nodos.

- **Reativo** - O algoritmos de roteamento podem ser pró-ativos ou reativos, como visto na seção 3.3. A diferença está entre o tempo necessário para encontrar uma nova rota e o custo necessário para manter atualizadas as informações da tabelas de roteamento. A vantagem ou desvantagem deste tópico é dependente das aplicações que podem estar usando o protocolo de roteamento.

- **Suporte a *links* Unidirecionais** - Em redes móveis podem haver situações em que a comunicação só ocorre em um sentido, do nodo A para o nodo B, mas não de B para A por exemplo. O problema é que alguns algoritmos consideram que o caminho por onde o pacote de descobrimento de rotas passou para encontrar o destino pode ser utilizado pelo destino para responder a requisição, o que nem sempre pode ser verdade.

- **Suporte a QoS** - Para várias aplicações é fundamental que o algoritmo de roteamento possa aceitar parâmetros de QoS e que encontre rotas com características bem determinadas. Por exemplo, as características que um programa de vídeo sob demanda exige se comparado com um programa de transferência de arquivos diferem completamente, e é importante que o algoritmo de roteamento consiga encontrar a melhor rota para cada caso.

- *Multicast* - A capacidade do algoritmo de roteamento fazer *multicast* é importante pois, além deste tipo de comunicação ser requerida por alguns aplicativos, ela pode também diminuir o volume de pacotes na rede.

- **Preocupação com segurança** - Como é comentado na RFC 2501 [13], o algoritmo de roteamento por si só é vulnerável a diversas formas de ataque. A caracterização apresentada aqui considera apenas os algoritmos que têm algum tipo de preocupação explícita com segurança ou não.

- **Preocupação com economia de energia** - Certamente, mais que qualquer um dos outros problemas enfrentados pela computação móvel, a quantidade limitada de energia é o pior deles. Desta maneira, é muito importante que o algoritmo tente economizar energia ao máximo.

- **Necessidade de mensagens periódicas** - É importante saber se o algoritmo depende de mensagens periódicas. Além destas mensagens gastarem recursos da rede apenas com mensagens de roteamento, a perda de alguma destas mensagens pode causar falhas no funcionamento do algoritmo.

- **Requer informações confiáveis ou em seqüência** - Esta característica é importante, pois não se pode esquecer que o meio sem fio não é estável. Desta forma é desejável que o algoritmo possa receber mensagens fora de ordem, ou mesmo que suporte perda de mensagens.

- **Características de *Bellman-Ford*** - O objetivo aqui é verificar se o algoritmo apresenta alguma das características do algoritmo de Vetor de Distâncias.

- **Características do *Link State*** - Quer-se aqui verificar se o algoritmo apresenta alguma das características do algoritmo *Link State*.

- **Faz *flooding* para transmitir informações de roteamento** - É importante saber se o algoritmo utiliza *flooding* em algum momento, pois isto pode acarretar em um número muito grande de pacotes na rede, e por conseqüência, um gasto excessivo de recursos da rede.

- **Tem suporte a circuito virtual** - Quer-se aqui saber se o algoritmo tem suporte a fluxo de dados. Esta característica é exigida por várias aplicações nas mais diversas situações.

- **Tem suporte a datagrama** - Outra forma de se transmitir dados é por datagrama. As duas formas, tanto circuito virtual quanto datagrama, são necessárias em uma rede de computadores. O primeiro possibilita uma transmissão mais confiável, e o segundo uma maior agilidade na transmissão de pacotes, o que em determinadas situações é necessário.

- **Leva em consideração informações geográficas** - O roteamento geográfico está se demonstrando uma das mais promissoras soluções para o problema de localização dos nodos em computação móvel. Com a informação de posicionamento do nodo, vários problemas são resolvidos, pois mesmo com a movimentação do nodo podemos ter uma estimativa apurada de sua posição se avaliarmos a idade da informação e a sua velocidade provável do nodo.

- **Informações de roteamento são trocadas com mais de um nível de nodos** - Alguns algoritmos de roteamento utilizam-se apenas de informações sobre seus nodos vizinhos, outros já buscam informações de nodos mais distantes. Esta diferença pode ter uma grande influência no algoritmo, pois obter informações sobre nodos mais distantes pode melhorar as suas rotas. Por outro lado, a aquisição destas informações pode ser difícil, e em certos casos, a falta destas pode acarretar em deficiências no desempenho do algoritmo de roteamento. Outro fator importante é que normalmente são necessários diversos pacotes, que têm o objetivo de somente atualizar as informações dos nodos distantes.

- **Considera diferenciação entre nodos** - Em ambientes reais existem diferentes tipos de *devices* que podem acessar a rede ad hoc. Estes dispositivos podem variar com relação a sua autonomia e capacidade de transmição. É desejável que os algoritmos de roteamento tenham a capacidade de diferenciar estas diferentes classes de nodos e utilizar esta informação da melhor maneira possível.

- **Faz *flooding* para encontrar o caminho** - Esta característica é importante pois o processo de descobrimento de rotas é um processo delicado em redes móveis ad hoc.

Normalmente não se tem nenhuma idéia de onde está o nodo, e o meio mais confiável para se encontrar o destino é o *flooding*.

- **Apresenta mais de uma cópia do pacote de dados na rede, ao mesmo tempo** - É importante observar se o algoritmo de roteamento faz cópias do pacote de dados durante o processo de entrega da mensagem, já que isto pode tanto representar uma vantagem, como uma desvantagem. Pode ser uma vantagem, pois ao fazer uma cópia da mensagem, provavelmente está se aumentando a probabilidade de entregar essa mensagem ao destino. Por outro lado, esta característica pode representar um desperdício de recursos da rede, pois o algoritmo está utilizando mais recursos que o estritamente necessário para entregar a mensagem.

CARACTERÍSTICA	DSR	AODV	GSR	ZRP	TORA	ABR	LAR	GPSAL
Livre de *loops*	■	■	■	■	◧	■	□	■
Múltiplas rotas	■	□	■	□	□	□	■	■
Distribuído	■	■	■	■	■	■	■	◧
Reativo	■	■	□	◧	■	■	■	□
Suporte a *links* Unidirecionais	■	□	□	□	□	□	■	■
Suporte a QoS	□	□	■	□	□	■	□	■
Multicast	□	■	□	□	□	□	□	□
Preocupação com segurança	□	□	□	□	□	□	□	□
Preocupação com economia de energia	□	□	□	□	□	□	□	■
Necessita de mensagens periódicas	■	■	■	■	■	■	□	■
Requer informações confiáveis ou em seqüência	□	□	□	□	■	□	□	□
Características de *Bellman-Ford*	□	■	□	◧	■	■	□	◧
Características do *Link State*	□	□	■	◧	□	■	□	◧
Faz *flooding* para transmitir informações de roteamento	■	■	□	■	■	■	■	□
Tem suporte a Circuito Virtual	■	■	■	□	■	■	□	■
Tem suporte a datagrama	□	□	□	■	□	□	■	■
Leva em consideração informações geográficas	□	□	□	□	□	□	■	■
Informações de roteamento são trocadas com mais de um nível de nodos	■	■	□	■	■	□	■	■
Considera diferenciação entre nodos	□	□	□	□	□	□	□	■
Faz *flooding* para encontrar o caminho	■	■	□	◧	■	■	■	□
Apresenta mais de uma cópia do pacote de dados na rede, ao mesmo tempo	□	□	□	□	□	□	■	□

Legenda: ■ : SIM ◧ : PARCIALMENTE □ : NÃO

Tabela 9.1: Comparação entre algoritmos de roteamento para redes ad hoc

9.2 Fatores que não podem ser enquadrados na tabela

Agora serão discutidos alguns pontos que não podem ser definidos com um simples Sim, Não ou Parcialmente.

Necessita de confiabilidade nas ligações entre *links*

A importância deste tópico se deve ao fato de o algoritmo depender de uma camada de mais baixo nível que garanta, além da entrega da mensagem, que esta ocorra livre de erros. Isto é importante para o algoritmo de roteamento pois ele fica dependente do bom funcionamento de outra camada tornando-se suscetível a possíveis falhas e erros de outras camadas e seus protocolos.

- DSR - Pode ou não enviar uma mensagem de *Acknowledgement* (ACK) para o nodo anterior, dependendo da forma como a comunicação em questão está definida
- AODV - Usa o 802.11 como base para a comunicação, ou seja CSMA/CA+ACK para transmissão dos pacotes, o que subentende-se uma necessidade de uma camada de mais baixo nível confiável
- GSR - Não está explícito na definição do protocolo, mas na parte de testes a simulação assume que não existem erros na camada de mais baixo nível
- ZRP - Utiliza o protocolo ICMP (*Internet Control Message Protocol*) ?? para garantir a confiabilidade na camada de baixo
- TORA - Trabalha em conjunto com o protocolo IMEP (*Internet MANET Encapsulation Protocol*) ?? que controla toda a parte de baixo nível e garante a confiabilidade da mensagem
- ABR - O nodo, ao receber um pacote, deve enviar uma confirmação a quem lhe enviou o pacote
- LAR - Não está explícito na definição do algoritmo, mas não tem nenhuma função para contornar falhas do nível inferior
- GPSAL - Confia que a camada MAC irá entregar os pacotes corretamente

Qual a decisão tomada em caso de perda de rota

Este ponto é fundamental no bom desempenho de algoritmos de roteamento para redes móveis ad hoc. Devido a mobilidade dos nodos, uma rota previamente estabelecida pode ser rompida, e quanto antes, e com maior economia possível, o algoritmo detectar e corrigir a rota, melhor é para o bom desempenho da rede.

- DSR - O nodo que encontrou a falha envia um aviso para a origem. Esta por sua vez verifica se tem alguma rota no cache de rotas, não encontrando faz uma nova requisição de rotas ou simplesmente finaliza a transmissão
- AODV - Tenta encontrar uma nova rota para o próximo nodo. Se não conseguir, envia um pacote de perda de rota para a origem que se encarrega de tentar encontrar um novo caminho ou finalizar a comunicação

- GSR - O nodo intermediário tenta encontrar uma nova rota
- ZRP - O nodo intermediário pode avisar a origem ou tentar encontrar uma nova rota. Mas nenhuma das duas opções está bem definida
- TORA - O nodo intermediário tenta encontrar uma nova rota ajustando a sua altura a uma altura maior que a dos vizinhos com relação ao destino
- ABR - A atitude tomada pelo algoritmo é dependente da forma como ocorreu a alteração. Se a movimentação ocorreu no nodo origem um novo descobrimento de rotas é necessário. No caso de movimentações do destino, o nodo logo anterior tenta um descobrimento parcial de rotas tentando encontrar o destino. Caso não consiga o nodo anterior é acionado. A movimentação dos nodos intermediários também acarreta em descobrimentos parciais de rota
- LAR - Não explicita nenhum procedimento especial, mas como usa *flooding*, não tem problema de uma rota se perder, pois "todas as rotas" são testadas e havendo algum outro caminho este será automaticamente utilizado
- GPSAL - É iniciado um redirecionamento no nodo intermediário. Mas os nodos estão constantemente monitorando a movimentação dos outros nodos, efetuando um roteamento pró-ativo. Sempre se tenta contornar a situação de quebra de *link* antes que esta ocorra

Utilizam tabelas de roteamento

Queremos aqui caracterizar o algoritmo com relação à utilização de alguma tabela para fazer o roteamento. Isto tem impacto no tempo e na quantidade de memória e pacotes gastos na atualização e manutenção das tabelas

- DSR - Não se utiliza de tabelas de roteamento, possui apenas uma tabela que indica as rotas que o próprio nodo originou.
- AODV - Utiliza uma tabela que indica as rotas das quais o nodo participa, e outra de *Multicast*
- GSR - Sim
- ZRP - Sim
- TORA - Possui uma tabela que indica apenas quais são seus vizinhos
- ABR - Sim
- LAR - Não, possui apenas uma tabela de pacotes recentemente recebidos, mas não uma tabela de roteamento propriamente dita
- GPSAL - Sim

9.3 Comentários

Em [6] pode-se encontrar uma comparação entre os algoritmos DSDV, TORA, DSR e AODV. A conclusão, de forma geral, desse trabalho é que alguns protocolos funcionam bem em algumas situações e outros não. Em [59] também é feita uma comparação entre alguns algoritmos de roteamento para redes ad hoc, dentre eles o TORA, AODV, DSR e ABR, e a conclusão é semelhante.

Capítulo 10

Problemas a Serem Explorados

Serão discutidos aqui alguns tópicos que podem ser estudados para melhorar o desempenho de algoritmos de roteamento, bem como para identificar suas limitações. Os itens aqui apresentados podem, e devem, ser utilizados não só para melhorar o desempenho do GPSAL, mas como também podem ser aplicados a outros algoritmos de roteamento em redes ad hoc. Este é um aspecto importante do trabalho, já que a maior parte dos protocolos para redes ad hoc ainda não trata os temas abordados nesta seção. Pretende-se, no futuro, avaliar e classificar diversos outros algoritmos, apontando suas deficiências e o seu comportamento de acordo com os pontos abordados abaixo.

10.1 Estudo Analítico

Na seção 7.4 apresentou-se a complexidade do GPSAL com relação ao processamento, mas diversos outros aspectos ainda devem ser observados. Como exemplo deve-se analisar sua complexidade com relação ao número de mensagens enviadas. É necessário fazer um estudo analítico do protocolo, para que se tenha uma base sólida com relação a seus requerimentos e extensibilidade. Sabendo da complexidade em termos de processamento, de memória e de troca de mensagens, pode-se prever o desempenho do algoritmo em diversas faixas de número de nodos e transmissões de mensagens, sem a necessidade de simulação. Hoje ainda são poucos os algoritmos que apresentam estas diversas análises.

10.2 Verificação Formal

Várias falhas em sistemas, já em produção, têm sido descobertas através de técnicas de verificação formal. Verificar formalmente um sistema distribuído significa saber se o sistema possui ou não uma propriedade. Deste modo pode-se encontrar falhas ou características, que de outra forma seria quase impossível. Por exemplo, simular um algoritmo é uma forma tradicionalmente aceita de provar que o mesmo funciona. Infelizmente pode acontecer de se testar um sistema através de simulações exaustivas sem nunca testar um caso específico. Quando verificamos formalmente um sistema, obtemos uma avaliação mais consistente, e

precisa, do problema. Não existe ainda na literatura nenhuma referência a algoritmos de roteamento para redes ad hoc testados formalmente de forma genérica.

10.3 Modelagem de Falhas

Pode-se, através de simulações ou de verificação formal, criar cenários considerando falhas no modo de operação nos nodos móveis. Estas falhas podem variar desde o desligamento de um nodo, ou falha na comunicação, até problemas de endereçamento de nodos dentro da rede fixa. Estes testes, com relação falhas, podem fornecer informações precisas com relação robustez do algoritmo. São raros os protocolos que apresentam tratamento a cenários de falhas imprevistas e quando apresentam, quase sempre, a solução adotada é a de reiniciar todo o processo.

10.4 Considerações Sobre Classes de Computadores

Um aspecto que agora começa-se a discutir mais seriamente é a possibilidade de haver "classes" entre os nodos da rede. Por exemplo, pode-se classificar nodos de uma rede como nodos roteadores e outros que não estão aptos a rotear mensagens [50], que é a classificação que o GPSAL adota. Normalmente os algoritmos de roteamento consideram todos os nodos como roteadores e com igual capacidade. O problema é que em um cenário real podemos ter *laptops, palmtops*, ou mesmo *pagers*, que não apresentam entre si as mesmas capacidades de processamento, ou autonomia com relação à fonte de energia. A idéia de classes de dispositivos computacionais permite modelar mais realisticamente os cenários existentes na prática. O GPSAL, como já foi dito, tem a capacidade de lidar com duas classes de dispositivos, mas diversas outras devem ser criadas, de forma a aumentar a capacidade e generalidade do algoritmo.

10.5 Qualidade de Serviço

Tão importante quanto a entrega da mensagem, é a forma como esta é entregue. Várias aplicações em redes de computadores necessitam não só que as mensagens sejam entregues, mas que sejam de forma a satisfazer um conjunto mínimo de requisitos. O conjunto de requisitos de Qualidade de Serviço (QoS–*Quality of Service*) é fundamental ao bom funcionamento dos protocolos das camadas superiores.

Os requisitos de QoS podem variar dependendo do objetivo da aplicação. Por exemplo uma aplicação de vídeo conferência além de uma banda constante alocada, necessita que se respeite um atraso máximo entre duas seqüências de imagens. O não cumprimento destes parâmetros pela rede compromete o bom desempenho das camadas superiores. A negociação destes parâmetros e a formalização de quais parâmetros podem ser trocados é um ainda tema de estudos. Mas devido a sua necessidade é importante que existam algoritmos de roteamento para redes móveis ad hoc que levem em consideração parâmetros de QoS. O GPSAL apresenta todo o arcabouço necessário a utilização de QoS em redes ad

hoc, mas os parâmetros de QoS necessários a redes ad hoc ainda precisam ser estudados mais profundamente. Ainda não se tem parâmetros específicos para redes ad hoc, o que é usado hoje são parâmetros próprios para redes fixas, que são adaptados a redes ad hoc. O algoritmo deve, idealmente, ser modularizado a fim de permitir a inclusão de novos parâmetros com características distintas.

10.6 Circuitos Virtuais Flexíveis

Uma das contribuições do trabalho é a idéia de Circuitos Virtuais Flexíveis (CVF). Circuitos virtuais, de uma forma geral, não permitem um redirecionamento dos pacotes. A partir do momento em que se estabeleceu o circuito este não é mais alterado. Em caso de problemas em algum nodo intermediário, o CV é fechado e é necessário que outro seja estabelecido. Em computação móvel os nodos participantes do circuito podem alterar sua posição com relativa freqüência, podendo inclusive sair da área de cobertura dos vizinhos. No algoritmo DSR [37] é proposto uma forma de redirecionamento, que consiste em uma reinicialização na origem. Reinicializar o CV sempre que acontece uma alteração de configuração pode acarretar em uma sobrecarga de processamento e de número de pacotes na rede móvel. A idéia que o GPSAL traz é que a qualquer momento, se um nodo perceber alguma falha no CVF, ou possibilidade de falha, ele pode alterar sua rota. Alguns dos pontos que devem ser levados em consideração no redirecionamento foram abordados na seção 7.2. O estudo do redirecionamento, nas duas formas propostas, deve dar uma idéia melhor sobre o impacto que o CVF tem no trafego da rede móvel.

10.7 Heurísticas

Técnicas heurísticas podem ser utilizadas, de forma eficiente, em algoritmos de roteamento. O estudo destas técnicas, não só de colônias de formigas [7] mas também, algoritmos genéticos como visto em [19, 30, 39], e outras podem ser utilizadas com sucesso para resolver problemas complexos de grafos, entre eles roteamento. Desta forma pode ser uma nova direção para os algoritmos de roteamento o uso destas técnicas como forma de aumentar a capacidade e o desempenho de algoritmos de roteamento.

10.8 Multicast

Um fator que deve ter uma relevância grande no desempenho nos futuros algoritmos, e que ainda não é explorado com eficiência pelos algoritmos atuais para redes ad-hoc, é o atendimento ao tráfego *multicast*. Cresce cada vez mais o número de aplicações que necessitam deste tipo de serviço. O crescimento do número destes serviços implica na necessidade cada vez maior de algoritmos que lidem com este tipo de comunicação de forma eficiente. A forma de se fazer multicast de modo eficiente em redes ad hoc ainda não é um consenso. Existem algumas abordagens iniciais, como a do AODV que basicamente é um flooding, mas que de modo algum é eficiente.

10.9 Estudo de Compromissos

Outra questão a ser estudada diz respeito às relações de custo benefício que os algoritmos apresentam. Alguns dos compromissos a serem analisados são:

- Para que valores de formigas na rede há uma melhoria na convergência do protocolo. Se o número de formigas crescer indefinidamente a rede vai ficar saturada de informações e a sobrecarga de processamento e de pacotes na rede aumenta. Logo, é importante determinar um ponto de equilíbrio.
- Qual o impacto da utilização da rede fixa no tráfego dentro da rede móvel. A partir de que número, ou porcentagem, de computadores fixos há uma melhora significativa no desempenho do algoritmo.
- Deve-se determinar qual é a porcentagem mínima de computadores da rede que deve atuar como roteador para que se consiga uma cobertura total. Outro problema tão importante quanto este, é determinar, de forma eficiente, quais serão estes nodos.
- A partir de que ponto a falta de precisão do GPS afeta o desempenho do protocolo. O que se deve observar no protocolo para que este erro não tenha uma influência significativa.
- Existe um compromisso entre a velocidade de processamento da mensagem e a velocidade com que o computador se locomove. Deve-se estudar a partir de que ponto o atraso no processamento do pacote pode causar ineficiência na forma de funcionamento do protocolo.
- Deve-se determinar a velocidade com que a rede toma conhecimento da chegada de um novo nodo na rede, e que técnicas devem ser adotadas para que este tempo seja minimizado.

10.10 Uso das Novas Técnicas em Algoritmos Existentes

Uma outra possível vertente do trabalho é utilizar a idéia de agentes formigas, CVF e GPS em outros protocolos, como o *Zone Routing* [27] ou *Hierarchical Routing* [44] e estudar qual o impacto que estas novas idéias causam nestes protocolos.

10.11 Comentários

Neste capítulo foram vistos alguns pontos que podem ser explorados para aumentar a eficiência de algoritmos de roteamento para redes ad hoc. É possível que em poucos anos existam grandes redes móveis, as quais podem conter tráfego de aplicações multimídia, ou de tempo real, que exigirão, para trabalharem de forma eficiente, algoritmos de roteamento melhores que os atuais. Acredita-se que muitos problemas ainda precisam ser solucionados para que as redes ad hoc sejam viáveis, mas certamente, quando isto ocorrer, os pontos acima listados devem ter sido observados nos futuros e eficientes algoritmos.

Capítulo 11

Conclusões e Observações

A integração de computadores com comunicações e outras formas de tecnologias de informação estão criando novas formas de sistemas e serviços de informação distribuída. A tendência é haver computadores mais poderosos, conectados a uma rede mundial de serviços e recursos através de uma infra-estrutura de alto desempenho. Este é o surgimento dos ambientes de computação ubíquos que deverá ser a nova forma de trabalho da próxima década. Este é o cenário altamente desafiador e excitante que motiva a computação móvel. Nesse cenário as redes móveis ad hoc terão uma importância cada vez maior. Sendo um dos problemas fundamentais deste tipo de rede, o roteamento de mensagens [49].

O acesso a rede, independente de um computador fixo, não é mais sonho ou ficção científica, mas sim está cada vez mais próximo de se tornar a nossa realidade diária. Nem mais o céu é o limite, já que os satélites de comunicação o cruzam interligando as pessoas das mais diferentes partes do globo. Hoje a terra não tem mais locais incomunicáveis, graças a projetos como o *Globalstar* [25], hoje é possível que pessoas possam trocar informações em praticamente todas as partes do globo terrestre.

Os avanços em redes de comunicação móvel não param. Projetos como *wearable computers* [5, 60] e robôs cooperativos [66], por exemplo, representam um horizonte cada vez mais amplo para redes móveis ad hoc. Contudo estes temas são indicativos também, da necessidade de eficientes algoritmos de roteamento, que realizem, de forma eficiente, a comunicação entre os nodos móveis. Existe muito ainda para ser feito, não só em redes ad hoc, mas em redes sem fio de um modo geral.

O objetivo deste trabalho foi, desde o início, caracterizar este novo ambiente computacional, as redes ad hoc. Diversos pontos foram levantados durante o estudo dos algoritmos existentes, e das outras diversas técnicas que foram surgindo durante o estudo de algoritmos de roteamento de forma geral. Este estudo levou a proposta de um novo algoritmo, o GPSAL. Este novo algoritmo, que apresenta diversas características novas, apresentou-se, de acordo com os testes realizados, eficiente e ao mesmo tempo robusto. No capítulo 9 apresenta-se uma comparação entre alguns dos mais importantes algoritmos de roteamento para redes móveis ad hoc, incluindo o GPSAL, e lá pode-se observar, comparativamente com outros algoritmos, as características do GPSAL.

O GPSAL (*GPS Ant Like Routing Algorithm*), nas comparações apresentadas no capítulo 8, se mostra eficiente tanto com relação ao número de mensagens enviadas na rede

como com relação a economia de processamento. As comparações entre o GPSAL e o LAR podem apresentar uma comparação mais clara com relação a eficiência do GPSAL. O mais importante com relação ao GPSAL, é que mesmo apresentando uma grande eficiência e um bom desempenho, muito ainda pode ser feito para melhorar os seus resultados. Como apresentado no capítulo 10, vários tópicos podem ainda ser abordados para aumentar, ainda mais, a eficiência do protocolo. Novas técnicas heurísticas podem ser abordadas, diversos os limites da usabilidade do GPSAL devem ser estabelecidos, questões como o *multicast* e os parâmetros ideais de QoS devem ainda ser adicionados ao protocolo, aumentando ainda mais a capacidade do GPSAL.

Muito ainda pode ser feito, não só com relação a ajustes no GPSAL, que já demonstrou ser um grande avanço com relação a diversos protocolos existentes, mas também na área de redes ad hoc em geral. Diversos dos tópicos abordados no decorrer deste trabalho, que foram frutos de exaustivas pesquisas, só agora começam a ser abordadas pelo grupo de estudos para redes móveis ad hoc do IETF. Um bom exemplo disto é o uso de diferentes classes de algoritmos, que é um problema que já foi identificado a algum tempo e que vem sendo estudado pelo GEDOC/DCC (Grupo de Estudo em Redes Móveis Ad Hoc) já a algum tempo, foi um dos temas dos últimos e-mails enviados a lista do MANET. O uso de classes no GPSAL é abordado de forma inicial, o computador pode ou não ser roteador, mas já é um avanço na área, mesmo assim este é um tópico que ainda deve ser melhorado no GPSAL. O uso da rede fixa no roteamento de mensagens, que o próprio GPSAL já prevê, também é um tema que começa a ser observado agora no grupo MANET. Pode-se observar com isto que os estudos realizados pelo grupo de redes móveis do DCC/UFMG, mais do que estar de acordo com as mais recentes pesquisas na área de redes móveis, estão conseguindo se manter um passo à frente na previsão das tendências de pesquisas de um dos mais novos e desafiadores ramos da area de redes de computadores.

Capítulo 12

Glossário de termos

- ABR: *Associativity-Based Routing*
- AODV: *On-Demand Distance Vector Routing*
- CVF: Circuito Virtual Flexível
- DBF: *Distributed Bellman-Ford*
- DSDV: *Destination-Sequenced Distance-Vector Routing Algorithm*
- DSR: *Dynamic Source Routing*
- DVMRP: *Distance Vector Multicast Routing Protocol*
- ESM: Estação de Suporte à Mobilidade
- GEDOC: Grupo de Estudos de Redes Móveis Ad hoc
- GPS: *Global Positioning System*
- GPSAL: *GPS Ant Like Routing Algorithm*
- GSR: *Global State Routing*
- HM: *Host* Móvel
- IEEE: *Institute of Electrical and Electronics Engineers*
- IETF: *Internet Engineering Task Force*
- IMEP: *Internet MANET Encapsulation Protocol*
- IP: *Internet Protocol*
- LAR: *Location-Aided Routing*
- LS: *Link State*

- LSI: Ideal *Link-State*
- MAC: *Medium Access Control*
- MANET: *Mobile Ad hoc Networks*
- MS: *Mobile Station*
- OSPF: *Open Shortest Path First*
- PIM: *Protocol-Independent Multicast*
- QoS: *Quality of Service*
- RIP: *Routing Information Protocol*
- TCP: *Transfer Control Protocol*
- TORA: *Temporally-Ordered Routing Algorithm*
- TTL: *Time To Live*
- UDP: *User Datagram Protocol*
- VSN: Vetor de Seqüências de Números
- ZRP: *Zone Routing Protocol*

Referências Bibliográficas

[1] K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows Theory, Algorithms and Applications.* Prentice Hall, 1993.

[2] Sameh Asaad. Simulation environment for an ad-hoc wireless network running the AODV routing algorithm. http://www.ctr.columbia.edu/ãngin/e6950/sameh/aodv_final.html, 1996.

[3] Phil Belanger and Wim Diepstraten. MAC entity: MAC basic access mechanism privacy and access control. Technical report, IEEE 802.11, March 1996.

[4] D. Bertsekas and R. Gallanger. *Routing in Data Networks.* Prentice-Hall, 1997.

[5] Mark Billinghurst and Thad Starner. Wearable devices: New ways to manage information. *Computer*, 1(1):57–64, January 1999.

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *In Proceedings of MOBICOM'98*, Dallas Texas USA, October 1998. ACM.

[7] G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *31st Hawaii International Conference on Systems*, Big Island of Hawaii, USA, January 6–9 1998.

[8] Tsu-Wei Chen. *Efficient Routing and Quality of Service Support for Ad Hoc Wireless Networks.* PhD thesis, University of California Los Angeles, April 1998.

[9] Tsu-Wei Chen and Mario Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of IEEE International Conference on Communications (ICC'98)*, Atlanta, June 1998.

[10] M.S. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *ACM Journal on Wireless Networks*, 1(1):61–81, 1995.

[11] M.S. Corson and V. Park. An internet MANET encapsulation protocol (IMEP) specification. Internet Draft, February 1999.

[12] S. Corson, S. Batsell, and J. Macker. Architectural considerations for mobile mesh networking. Internet draft RPC, version 2, May 1996.

[13] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. IETF RFC 2501, January 1999.

[14] B. Das, E. Sivakumar, and V. Bhargavan. Routing in ad-hoc networks using a spine. In *IEEE International Conference on Computer Communications and Networks*, 1997.

[15] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[16] G. Dommety and R. Jain. Potential networking applications of global positioning systems GPS. Technical Report TR-24, Department of Computer Science, The Ohio State University, April 1996.

[17] M. Dorigo. Optimization, learning and natural algorithms. In *Ph.D. Thesis*, Politecnico di Milano, Italy, 1992.

[18] R. Dube, D.D. Rais, K. Wang, and S.K. Tripathi. Signal stability based adaptive routing (SSA) for ad hoc mobile networks. *IEEE Personal Communications*, February 1997.

[19] Herman Ehrenburg. Improved direct acyclic graph handling and the combine operator in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 285–291, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[20] Greg Ennis. 802.11 architecture. Technical report, IEEE 802.11, March 1996. Tutorial of draft Standard IEEE 802.11.

[21] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. RFC 2362: Protocol independent multicast-sparse mode (PIM-SM): Protocol specification, June 1998.

[22] R. W. Floyd. ACM algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, June 1962.

[23] Internet Engineering Task Force. http://www.ietf.org.

[24] J. J. Garcia-Luna-Aceves. A unified approach to loop-free routing using distance vectors or link states. In *Proc. ACM SIGCOMM'89*, pages 212–223, Austin, TX, September 1989.

[25] Globalstar. http://www.globalstar.com/.

[26] UCB Multicast Network Research Group. Ucb/lbnl/vint network simulator - ns. http://www-mash.cs.berkeley.edu/ns/.

[27] Z.J. Haas and M.R. Pearlman. The zone routing protocol (ZRP) for ad hoc networks. Internet-draft, August 1998.

[28] Z.J. Haas and S. Tabrizi. On some challenges and design choices in ad-hoc communications. *MILCOM'98*, October 1998.

[29] Zygmunt J. Haas. A new routing protocol for the reconfigurable networks. *ICUPC'97*, October 1997.

[30] Thomas Haynes, Roger Wainwright, Sandip Sen, and Dale Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 271–278, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[31] Hewlett-Packard. GPS and precision timing applications. Application Note 1272 Publication Number:5965-2791E, Hewlett-Packard, May 1996.

[32] R. Hinden and S. Deering. Ip version 6 addressing architecture. IETF RFC 1884, Dec. 1995.

[33] Christian Huitema. *Routing in the Internet.* P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1995.

[34] Cisco Systems Inc. Routing basics. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_d
1999.

[35] M. Jiang, J. Li, and Y.-C. Tay. Cluster based routing protocol (CBRP) functional specification. Internet Draft, August 1998.

[36] D. Johnson, D.A. Maltz, and J. Broch. Dynamic source routing in ad hoc wireless networks. Mobile Computing, edited by Tomasz Imielinski and Hank Korth, 1996.

[37] D. Johnson, D.A. Maltz, and J. Broch. The dynamic source routing protocol for mobile ad hoc networks. Internet Draft, March 1998.

[38] F. Kamoun and L. Kleinrock. Stochastic performance evaluation of hierarchical routing for large networks. *Computer networks*, 3:337–353, 1979.

[39] I. M. A. Kirkwood, S. H. Shami, and M. C. Sinclair. Discovering simple fault-tolerant routing rules using genetic programming. In *ICANNGA97*, University of East Anglia, Norwich, UK, 1997.

[40] L. Kleinrock and K. Stevens. Fisheye: A lenslike computer display transformation. Technical report, UCLA, 1971.

[41] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 66–75, Dallas, Texas, USA, October 25–30 1998.

[42] P. Krishna, M. Chatterjee, N.H. Vaidya, and D.K. Pradhan. A cluster-based approach for routing in ad hoc networks. In *USENIX Symposium on Location Independent and Mobile Computing*, April 1995.

[43] Tony Larsson and Nicklas Hedman. Routing protocols in wireless ad-hoc networks - a simulation study. Master's thesis, LuleåUniversity of Technology, 1998.

[44] P. Lauder, R. Kummerfeld, and A. Fekete. Hierarchical network routing. In *Proceedings of TriComm'91*, Chapel Hill North Carolina, April 1991. IEEE Conference on Communications Software.

[45] G. Malkin. Rip version 2 carrying additional information. RFC 1723, November 1994.

[46] Mobile Ad-hoc Networks (manet). http://www.ietf.org/html.charters/manet-charter.html.

[47] J. Moy. The OSPF specification; RFC 1131. *Internet Request for Comments*, (1131), October 1989.

[48] J. Moy. Multicast extensions to OSPF; RFC 1584. *Internet Request for Comments*, (1584), 1994.

[49] Robert Casta neda and Samir R. Das. Query localization techniques for on-demand routing protocols in ad hoc networks. Technical Report CS-99-1, Divison of Computer Science, The University of Texas at San Antonio, San Antonio, January 1999.

[50] Abdhay K. Parekh. Selecting routers in ad-hoc wireless networks. *Journal of the Brazilian Computer Society*, Volume 1(Number 2):13–21, November 1994.

[51] V.D. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. Proceedings of IEEE INFOCOM'97, April 1997.

[52] V.D. Park and M.S. Corson. Temporally-ordered routing algorithm (TORA) version 1 functional specification. Internet Draft, August 1998.

[53] Vincent D. Park and M. Scott Corson. A performance comparison of the temporally-ordered routing algorithm and ideal link-state routing. In *Proceedings of IEEE Symposium on Computers and Communication'98*, Athens, Greece, June 1998. IEEE.

[54] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM Symposium on Communication, Architectures and Protocols*, 1994.

[55] C.E. Perkins and E.M. Royer. Ad hoc on demand distance vector (AODV) routing. Internet Draft, August 1998.

[56] Charles E. Perkins. Ip mobility support. IETF RFC 2002, Oct. 1996.

[57] Charles E. Perkins and David B. Johnson. Mobility support in ipv6. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking (MobiCom'96)*. ACM/IEEE, November 1996.

[58] CMU Monarch Project. Cmu monarch project extensions to ns-2. http://www.monarch.cs.cmu.edu/cmu-ns.html.

[59] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, April 1999.

[60] Jane Siegel, Robert E.Kraut, Bonnie E.John, and Kathleen M.Carley. An empirical study of collaborative wearable computer systems. *Human factors in computing systems*, 1995.

[61] I. Stojmenovic and X. Lin. GEDIR: Loop-free location based routing in wireless networks. In *IASTED Int. Conf. on Parallel and Distributed Computing and Systems*, pages 1025–1028, Boston,MA, USA, November 1999.

[62] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall, third edition, 1996.

[63] The GPS Store's automated online shopping site makes buying GPS units and accessories easy and inexpensive. http://www.thegpsstore.com/site/.

[64] C.-K. Toh. A novel distributed routing protocol to support ad-hoc mobile computing. In *IEEE 15th Anual Int. Phoenix Conf. Comp. and Commun.*, March 1996.

[65] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol; RFC 1075. *Internet Request for Comments*, (1075), 1988.

[66] B. B. Werger. Multiple agents from the bottom up. In *Proceedings of AAAI-97*, Providence, RI, 1997.