Uso de grafos evolutivos no roteamento em redes dinâmicas: algoritmos, fluxos e limites

Julian Geraldes Monteiro

DISSERTAÇÃO APRESENTADA
AO INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO
TÍTULO DE MESTRE EM CIÊNCIAS

Programa: **Ciência da Computação**
Orientador: **Prof. Dr. Alfredo Goldman vel Lejbman**

São Paulo, 20 de junho de 2007

(Este trabalho faz parte do projeto MOBYDIN *com o apoio da FAPESP/INRIA).*

Uso de grafos evolutivos no roteamento em redes dinâmicas: algoritmos, fluxos e limites

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Julian Geraldes Monteiro
e aprovada pela Comissão Julgadora.

São Paulo, 13 de julho de 2007.

Comissão Julgadora:

Prof. Dr. Alfredo Goldman vel Lejbman (orientador) – IME-USP

Prof. Dr. Sergio Takeo Kofuji – POLI-USP

Prof. Dr. Antonio Alfredo Loureiro – DCC-UFMG

à minha amada, Mayara

Agradecimentos

Primeiramente ao professor Alfredo Goldman, que muito confiou em mim e aceitou orientar este trabalho. Seu apoio, sua disponibilidade e enorme dedicação me motivaram a continuar os meus estudos e contribuíram para o meu amadurecimento como pesquisador. Além de sua alegria e amizade, levarei como exemplo a sua disciplina para os treinos de corrida.

Ao professor Afonso Ferreira, que abriu o caminho para este trabalho e esteve presente em momentos importantes. Aos estimados professores Yoshiko Wakabayashi, Carlos Eduardo Ferreira, Cristina Gomes Fernandes e Carlos Hitoshi Morimoto, que sempre estiverem presentes e muito me apoiaram.

Aos companheiros da AgilCoop e aos amigos do IME, que além das boas conversas compartilharam comigo suas idéas e experiências: Daniel Cordeiro, Danilo Sato, Giuliano Mega, Mariana Bravo, Hugo Corbucci, Roberto Speicys, Fabio Ide, Ricardo Bueno, Felipe Almeida e Maira Ramos.

Sou muito grato também aos meus pais, meus irmãos, minha querida esposa Mayara e a todos os amigos que me dão o prazer de viver a vida com alegria! Vocês são muito importantes para mim.

Muito obrigado!

Resumo

O comportamento dinâmico das redes sem fio as torna muito peculiares e de difícil análise. No entanto, algumas destas redes, como as de sensores com funcionamento intermitente, redes periódicas ou cíclicas e as do sistema de satélites de órbita baixa têm um comportamento dinâmico relativamente previsível, pois as variações da topologia da rede no tempo são quase que determinísticas.

Recentemente, um modelo teórico – *grafos evolutivos* – foi proposto com o intuito de capturar o comportamento dinâmico destas redes e formalizar algoritmos de roteamento de custo mínimo, além de outros. Os algoritmos e idéias obtidos com este modelo são teoricamente muito eficientes, mas, no entanto, antes deste trabalho não existiam estudos do uso destes modelos em situações práticas. Assim, o objetivo deste trabalho é analisar a aplicabilidade da teoria de *grafos evolutivos* na construção de protocolos de roteamento eficientes em cenários realistas.

Foram implementados dois protocolos de roteamento para redes móveis ad hoc baseados nos algoritmos de *grafos evolutivos*, são eles: Jornada que Chega Mais Cedo e Jornada Mais Curta. Extensivas simulações foram realizadas utilizando o simulador de redes NS2 e os resultados foram comparados com outros quatro protocolos clássicos para este tipo de rede: AODV, DSR, OLSR e DSDV. Os resultados preliminares mostram que este recente modelo tem muito potencial para ser uma ferramenta poderosa no desenvolvimento e análise de algoritmos para redes dinâmicas com comportamento previsível. No entanto, foram apontados alguns aspectos que precisam ser melhores estudados para que estes algoritmos possam ser utilizados em situações reais.

Abstract

The assessment of routing protocols for wireless networks is a difficult task, because of the networks' highly dynamic behavior and the absence of benchmarks. However, some of these networks, such as intermittent wireless sensors networks, periodic or cyclic networks, and low earth orbit satellites systems, have more predictable dynamics, as the temporal variations in the network topology are somehow deterministic, which may make them easier to study.

Recently, a graph theoretic model – the *evolving graphs* – was proposed to help to capture the dynamic behavior of these networks, in view of the construction of least cost routing and other algorithms. The algorithms and insights obtained through this model are theoretically very efficient and intriguing. However, before this work there was no study on the use of such theoretical results into practical situations. Therefore, the objective of our work is to analyze the applicability of the *evolving graph* theory in the construction of efficient routing protocols in realistic scenarios.

We use the NS2 network simulator to first implement two evolving graph based routing protocols: Foremost Journey and Shortest Journey, They are evaluated and compared to four major ad-hoc protocols: AODV, DSR, OLSR and DSDV. Interestingly, our experiments show that *evolving graphs* have all the potentials to be an effective and powerful tool in the development and analysis of algorithms for dynamic networks, with predictable dynamics at least. In order to make this model widely applicable, however, some practical issues still have to be addressed and incorporated into the model.

Lista de Abreviaturas

AODV *Ad hoc On-demand Distance Vector*

ARP *Address Resolution Protocol*

CBR *Constant Bit Rate*

DSDV *Destination-Sequenced Distance Vector*

DSR *Dynamic Source Routing*

DTN *Delay and Disruption Tolerant Networks*

EG *Evolving Graphs*

EG$_{fastest}$ Jornada Mais Rápida

EG$_{foremost}$ Jornada que Chega Mais Cedo

EG$_{shortest}$ Jornada Mais Curta

ERB Estação Rádio Base

FSDN *Fixed Schedule Dynamic Network*

HoldTime Tempo em espera

IFQ *Interface Queue*

LEO *Low Earth Satellite*

MANET *Mobile Ad hoc Network*

NS2 *Network Simulator 2*

OLSR *Optimized Link State Routing*

PauseTime Tempo de pausa

QoS *Quality of Service*

RREP *Route Reply*

RREQ *Route Request*

RSSF Rede de Sensores Sem Fio

RTS/CTS *Request to Send/Clear to Send*

SleepProb Probabilidade de dormir

RWP *Random Waypoint*

Lista de Figuras

Sumário

1. Introdução

Com seu início na década de 70, o interesse na comunicação sem fio vem se tornando cada vez mais popular. A partir de década de 90 o seu crescimento foi explosivo. Muito se deve à disseminação dos celulares e ao crescente uso de computadores pessoais portáteis (PDAs – *Personal Digital Assistants*), além da expansão da internet. Os baixos custos de aquisição e melhoras na qualidade dos serviços, como as altas taxas de transferência, fazem com que a computação móvel tenha um rápido desenvolvimento, tornando plausível a estimativa de que, em poucos anos, centenas de milhões de pessoas terão um dispositivo móvel em mãos, e a maior parte deles comunicar-se-ão entre si.

Logo, a utilização de redes de comunicação sem fio tem se tornado cada vez mais popular no nosso dia a dia. Uma área de pesquisa muito interessante é a das redes móveis ad hoc, conhecidas como MANETs (Mobile Ad hoc NETworks) [20]. Estas são compostas por uma coleção de dispositivos móveis que se conectam dinamicamente e de forma arbitrária, sem a necessidade de infra-estrutura fixa ou administração centralizada. Estes dispositivos móveis com capacidade de comunicação sem fio são chamados *nós*. Quando dois nós precisam trocar informações e não estão ao alcance um do outro, os vizinhos que estão fisicamente entre eles irão cooperar na comunicação, agindo como roteadores de dados, passando adiante a informação recebida até que esta chegue ao seu destinatário. Este tipo de rede é denominado **rede de comunicação sem fio ad hoc multi-saltos** (*multi-hop*).

Em diversas aplicações estes nós são livres para se movimentar e podem ter características não uniformes, o que torna estas redes complexas [25, 74]. Este comportamento altamente dinâmico e as suas características típicas [20] como, por exemplo, restrição no consumo de energia, limitação no processamento e pouca capacidade de comunicação (sujeita a altas taxas de erro), motivaram a pesquisa e o desenvolvimento de diversos algoritmos de roteamento multi-saltos [1, 52, 71, 78].

Na tentativa de desenvolver protocolos eficientes em relação a estas limitações, diferentes abordagens são utilizadas para otimizar o custo de roteamento, mas a maioria delas não leva em conta o possível fato de que algumas destas redes tem um **comportamento dinâmico previsível**, que é o caso de redes como as do sistema de satélites de órbita baixa (LEO), algumas redes de sensores, redes periódicas ou com comportamento cíclico e outras que de uma forma ou de outra tem um comportamento determinístico.

Sistemas de satélites de órbita baixa (LEO) [17,30,77] se comunicam através de conexões entre os satélites que estão ao alcance um do outro. A topologia dinâmica destas redes é fixa, pois as trajetórias dos satélites podem ser determinadas de antemão e é possível explorar este determinismo para otimizar estratégias de roteamento.

Uma Rede de Sensores Sem Fio (RSSF) [1,18,21,24,39,54,68,72,75] é um tipo especial de rede ad hoc, uma rede composta por centenas ou até milhares de nós inter-conectados que colaboram entre si para realizar alguma tarefa específica como, por exemplo, detecção de incêndios em florestas, mapeamento da bio-complexidade do meio-ambiente, rastreamento e detecção de veículos, etc. Devido às características destas redes e a limitações de energia, os nós podem ser desligados por certos períodos com o intuito de economizar recursos, portanto, por muitas vezes a sua topologia dinâmica pode ser pré determinada.

Estas redes com características determinísticas são denominadas **redes com comportamento previsível** (FSDN – *Fixed Schedule Dynamic Networks*) [11, 29], sendo que a topologia em diferentes instantes pode ser pré calculada. Contudo, o modelo clássico de grafos e suas teorias para o cálculo de caminhos de custo mínimo infelizmente não se aplicam a estas redes.

1.1. Proposta

A noção de ***Grafos Evolutivos*** (EG) introduzida recentemente [4, 5, 11, 12, 29, 31] foi proposta como uma abstração formal para redes dinâmicas, e pode muito bem ser aplicada às redes FSDN e na construção de algoritmos de roteamento ótimos em relação a determinadas métricas. As idéias e os algoritmos obtidos através deste recente modelo são teoricamente muito eficientes. No entanto, nenhum estudo havia sido realizado em relação à utilização deste modelo teórico e sua aplicabilidade em situações práticas de uso.

O objetivo deste trabalho é analisar a aplicabilidade dos algoritmos do recente modelo combinatório de EG em protocolos de roteamento para MANETs, redes de sensores sem fio e redes FSDN em geral. Em particular, foram implementado dois algoritmos de EG no

simulador de redes NS2 [61] e foram realizadas extensivas simulações.

1.2. Organização do Texto

Esta dissertação está organizada da seguinte forma: O Capítulo 2 apresenta uma introdução ao conceito de redes móveis ad hoc, alguns protocolos de roteamento e suas aplicações, seguido por características das RSSF, redes LEO e DTN. No Capítulo 3 resumimos os conceitos da teoria de *grafos evolutivos* e seus algoritmos de custo mínimo. O Capítulo 4 contém informações sobre o ambiente de simulação utilizado. Os resultados e a análise dos experimentos realizados estão no Capítulo 5. Por fim, no Capítulo 6 descrevemos as conclusões obtidas neste trabalho. O Apêndice A contém um exemplo de configuração de uma simulação no simulador NS2.

2. Redes Dinâmicas

Neste capítulo, abordamos as características básicas das redes móveis sem fio ad hoc e descrevemos algumas técnicas e protocolos de roteamento clássicos: AODV, DSR, OLSR e DSDV. Também é feita uma descrição das redes de sensores sem fio, redes de satélites e redes tolerantes a atrasos e desconexões.

2.1. Arquitetura IEEE 802.11

De acordo com padrão 802.11 [37], estabelecido pelo comitê IEEE (Institute of Electrical and Electronics Engineers), as redes móveis sem fio podem ser utilizadas em duas arquiteturas distintas:

1. A primeira, chamada ***rede infra-estruturada***, se utiliza de uma rede fixa para permitir a comunicação entre os seus participantes, que são os dispositivos móveis, unidades móveis ou simplesmente: *nós da rede* (veja figura 2.1.1). Um dispositivo móvel que pretende comunicar-se com outro precisa se conectar a um *Ponto de Acesso* (PA), *Estação Rádio Base* (ERB) ou um *Hotspot* (como é popularmente conhecido), geralmente o mais próximo ou com melhor qualidade de sinal, a partir daí toda comunicação entre eles é realizada através da ERB. A medida que o nó se move e sai do alcance da primeira estação, é necessário se conectar a uma nova ERB, este processo de troca é chamado de transição (*handoff*) e precisa ser realizado de maneira rápida e transparente para o usuário. Exemplos típicos destas redes são as redes locais sem fio (WLAN) muito utilizadas hoje em dia em escritórios e as de telefonia celular.

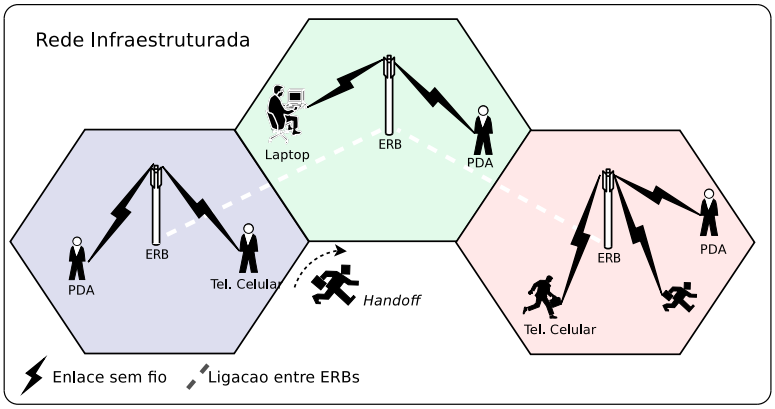


Figura 2.1.1.: Exemplo de rede infra-estruturada com 3 ERBs e 8 nós

2. O segundo tipo de arquitetura, que é a motivação deste trabalho e será estudada com mais detalhes, é a *rede independente* ou ***rede ad hoc*** (veja figura 2.1.2). Nesta, dois ou mais nós com capacidade de comunicação sem fio querendo se comunicar formam uma rede entre si, sem a necessidade de uma estrutura fixa de apoio. Computadores portáteis ou computadores de mão com capacidade de comunicação sem fio podem formar uma rede destas.

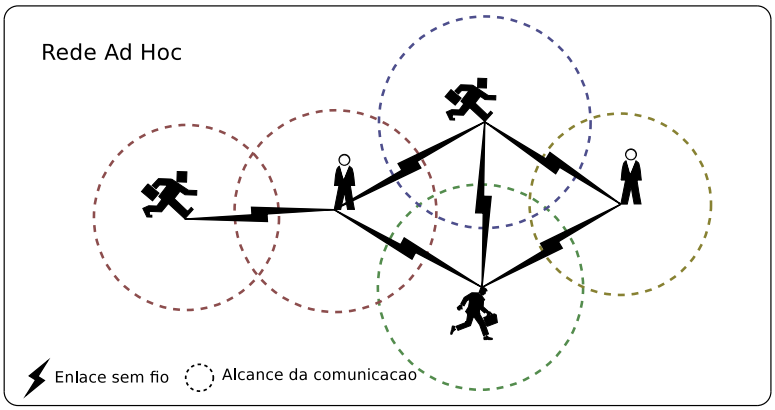


Figura 2.1.2.: Exemplo de rede ad hoc com 5 nós

Também é possível a existência de uma *rede híbrida*, na qual os nós de uma rede ad hoc podem se comunicar com os nós de uma rede infra-estruturada, por exemplo, um computador portátil sem acesso à internet, mas que faz parte de uma rede ad hoc, poderá

conseguir acesso à rede global através do auxílio de eventuais nós participantes que estejam conectados à ela.

O projeto OLPC (*One Laptop per Child*)[1] do MIT desenvolve um laptop de baixo custo voltado para crianças. Cada computador portátil terá embutido uma interface de rede no padrão IEEE 802.11s, sendo que o padrão 802.11s determina a arquitetura de funcionamento das redes Mesh [2]. Nesta arquitetura, os diversos computadores que compõe a rede poderão comunicar-se entre si quando estiverem ao alcance um do outro. Se algum dos dispositivos tiver acesso à rede internet, então todos os componentes a ele conectados também terão acesso.

Nestas redes altamente dinâmicas os nós podem entrar e sair a qualquer momento, bem como se mover arbitrariamente. A falta de estrutura prévia faz com os próprios nós participantes de rede ajam como roteadores, passando adiante pacotes de dados até que estes cheguem ao destinatário. O alcance desta rede está, então, limitado ao alcance de comunicação dos seus participantes. Redes ad hoc possuem algumas características que as tornam muito interessantes:

- **Instalação rápida** *(sob demanda)*: como não há a necessidade de infra-estrutura prévia, a instalação depende somente da disposição dos nós;
- **Tolerância a falhas**: o mau funcionamento, ou desligamento, de um nó não impede a continuidade do uso da rede pelos demais.

Portanto estas redes ad hoc são muito bem adaptadas para locais sem estrutura fixa de apoio, como em operações militares ou no auxílio ao salvamento em ambientes de desastre, em que a rápida configuração de uma rede é necessária e não há estrutura prévia suficiente. Além destes exemplos acima, novas tecnologias estão surgindo baseadas no uso do redes ad hoc, como é o caso das redes de sensores sem fio (RSSFs) que serão abordadas na próxima seção.

Dentre as restrições de uma rede ad hoc, os seguintes itens dificultam a sua disseminação (e motivam a pesquisa na área):

- **Taxa de transferência**: a banda de comunicação sem fio compartilha o mesmo meio físico e possui menos capacidade do que uma rede com fio, além de sofrer forte influência de intempéries do meio físico;

[1] One Laptop per Child :http://www.laptop.org/

- **Localização**: a posição de um nó da rede, geralmente, não é conhecida pelos outros nós, dificultando o envio de mensagens;

- **Roteamento dinâmico**: a topologia dinâmica requer o desenvolvimento de protocolos mais complexos para o roteamento de dados.

Devido à premissa de que os nós da rede, além de agir como participantes, também terão que retransmitir os pacotes de dados de seus vizinhos, é necessário que existam protocolos de roteamento específicos para desempenhar este papel.

2.2. Redes Móveis Ad Hoc

Para estabelecer a comunicação entre dois dispositivos móveis em uma rede ad hoc, pode ser necessário que os dados trafegados passem por diversos nós até chegar ao destino. Este processo é denominado comunicação multi-saltos *(multi-hop)* e, aliado à topologia dinâmica de uma rede ad hoc, exige que os protocolos de roteamento tenham algumas características especiais.

O Internet Engineering Task Force (IETF) [2], órgão responsável em padronizar os sistemas relacionados à internet, criou um grupo de trabalho com o intuito de padronizar os algoritmos de roteamento e arquiteturas com respeito às redes móveis. O grupo, chamado Mobile Ad hoc NETwork (MANET), publicou o primeiro trabalho em 1998, o RFC 2501 [20].

As seguintes métricas foram salientadas pelo grupo MANET como sendo as qualidades desejáveis para um protocolo de roteamento ad hoc:

1. **Operação distribuída**: o processamento não pode depender de um nó específico. Esta é uma propriedade fundamental em um algoritmo de roteamento para estas redes;

2. **Não apresentar ciclos no roteamento**: o algoritmo deve ser robusto em relação a pacotes trafegando na rede por períodos de tempo indeterminados;

3. **Operar de acordo com a demanda**: os algoritmos devem ser capazes de se adaptar a diferentes condições de tráfego;

[2] http://www.ietf.org

4. **Permitir períodos de sonolência**: capacidade de se adaptar aos períodos de inatividade de certos nós da rede (com o intuito de conservar recursos).

Em relação ao desempenho, o grupo MANET definiu as seguintes métricas quantitativas que devem ser otimizadas:

1. **Vazão e atraso origem-ao-destino dos pacotes de dados**: são características importantes pois medem o desempenho e a eficiência de um protocolo, isto é, quão bem ele está fazendo o seu trabalho;

2. **Tempo para aquisição de rotas**: característica importante principalmente nos algoritmos que obtêm a rota sob demanda, pois durante este período o pacote estará aguardando para ser enviado;

3. **Número de pacotes de controle**: quantidade de pacotes de dados de controle necessários para o correto funcionamento do protocolo (sobrecusto), isto é, são todos os pacotes trafegados entre os nós e que não sejam dados da aplicação;

4. **Razão de pacotes recebidos/enviados**: eficiência na quantidade de pacotes entregues em relação aos enviados.

Atualmente existem diversos algoritmos de roteamento que já foram amplamente estudados para este tipo de rede [20, 44, 52, 65, 66, 73, 74], cada um com suas qualidades e restrições, sendo que alguns são bem específicos para determinado domínio. Por enquanto nenhum protocolo se mostrou eficiente na totalidade das aplicações [71, 74] e portanto, não há um protocolo de roteamento padrão.

2.2.1. Classificação

Ao comparar as características dos diversos protocolos, é importante classificá-los em algumas categorias [52], de forma a ajudar na interpretação e análise dos mesmos. Dentre as inúmeras classificações existentes, as três a seguir são relevantes ao nosso estudo:

- **Reativos** *(sob demanda)* x **Pró-ativos** *(tabelas)*
- **Centralizados x Distribuídos**
- **Estáticos x Adaptativos**

A primeira grande classificação está relacionada com a maneira utilizada para obtenção da rota da origem ao destino. Protocolos **pró-ativos** tentam de alguma maneira armazenar as rotas existentes e seus respectivos custos, assim quando algum pacote precisa ser enviado, os nós já sabem de imediato qual será o caminho a ser percorrido. Dessa maneira, mesmo que nenhum participante da rede esteja transmitindo, os nós mantém comunicação para o estabelecimento das rotas.

Por outro lado, os protocolos **reativos**, procuram pela rota somente sob demanda. Quando uma rota é pedida, todo um processo de descobrimento de rotas origem-ao-destino é iniciado e somente depois disso o pacote é enviado. A vantagem deste último modelo é que a topologia da rede pode ser bem dinâmica, pois nenhuma rota é armazenada de antemão, já o modelo pró-ativo é mais rápido de imediato, pois a rota origem-destino já foi pré calculada, mas isto pode acarretar problemas de exatidão, principalmente se a topologia se alterar com muita freqüência.

Em relação ao modo de operação, podemos separá-lo em **centralizado** ou **distribuído**. Em um protocolo centralizado a decisão de roteamento é realizada por um nó central, enquanto que em um algoritmo distribuído a computação da rota é compartilhada pelos nós da rede. Roteamentos centralizados tem a vantagem de serem mais simples e os nós da rede não gastam processamento no cálculo da rota, mas, normalmente, são impraticáveis em redes ad hoc, tanto pela necessidade de existir um nó especial responsável, tanto pela topologia dinâmica não permitir rotas fixas de roteamento.

Outra maneira de categorizar os protocolos é através da capacidade ou não destes de alterar o roteamento devido ao padrão do fluxo de dados. Os protocolos **estáticos** usam a mesma rota origem-destino não importando se o fluxo de dados está sobrecarregando a rede. O algoritmo só irá alterar a rota em caso de alteração na topologia da rede. Já os protocolos **adaptativos** usam os dados de fluxo para melhor escolher as rotas e assim evitar congestionamentos.

Existem mais classificações possíveis além dos três grandes grupos acima, mas o estudo detalhado sobre estas características não está no escopo deste texto.

2.2.2. Protocolos e Técnicas Convencionais

Existem algumas técnicas clássicas de roteamento que não se aplicam diretamente às MANETs, mas que estão listadas a seguir pois muitos dos seus conceitos são utilizados em protocolos atuais e mais aprimorados:

1. **Inundação *(Flooding)***: é um algoritmo estático e seu modo de operação é muito simples. Cada pacote recebido por um nó é repassado a todos os vizinhos, a menos que ele já tenha repassado o mesmo pacote ou ele próprio seja o destinatário. Dessa forma garante-se que um pacote, caso seja entregue, chegará pelo caminho mais curto. Uma característica que chama atenção nesta abordagem é o grande número de pacotes que trafegam na rede;

2. **Vetor de Distâncias *(Distance Vector)***: é um conceito muito utilizado em protocolos de roteamento, conhecido por Belman-Ford Distribuído (DBF) [3]. Cada nó armazena uma tabela de roteamento com a distância e próximo salto para todos os outros nós da rede. Periodicamente os nós enviam a seus vizinhos tabelas de roteamento atualizadas, para que estes ajustem os seus valores. Cada nó da rede é mantido na tabela com uma entrada somente. Em caso de mudanças na topologia o protocolo tem um alto tempo de convergência das atualizações. Esta técnica pode gerar rotas contendo *ciclos*. É importante notar que:

   - Calcula a distância (custo) para *todos* os nós;
   - Troca essas informações somente com os *vizinhos*.

3. **Estado do Enlace *(Link State)***: algoritmos para redes fixas utilizam bastante este conceito, que não é muito popular em redes sem fio. Neste protocolo cada nó mantém uma visão completa da topologia da rede, com um custo para cada enlace. A manutenção destes custos é feita através de inundações na rede. A idéia do *link state* é análoga à de encontrar o caminho de custo mínimo (*shortest path*). Alguns custos mantidos pelos nós podem não ser corretos em razão do atraso na propagação da informação, particionamentos na rede, etc. É importante notar que:

   - Mede a distância (custo) para os *vizinhos*;
   - Troca essas informações com *todos* os outros nós.

4. **Roteamento na Origem *(Source Routing)***: *Source Routing* não é um protocolo, mas sim uma maneira de enviar dados relativos ao roteamento. Significa que cada pacote trafegado na rede carrega em seu cabeçalho a lista completa de nós que serão percorridos da origem até o destino. Portanto as decisões de roteamento são tomadas na origem. A vantagem desta abordagem é que não haverá *ciclos* no roteamento, e

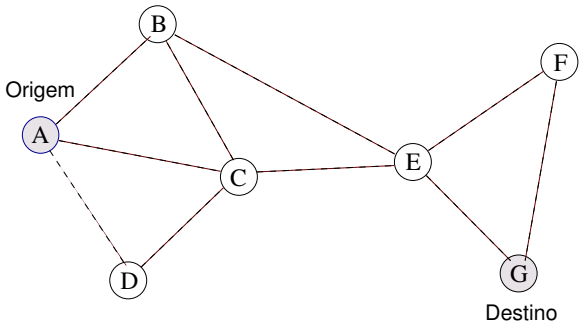
a desvantagem é que cada pacote de dados contém uma sobrecarga de informação, que poderá ser um problema se utilizado em redes com diâmetro grande.

A seguir será mostrada uma breve descrição do funcionamento de quatro protocolos clássicos para MANETs que serão utilizados nas simulações descritas no Capítulo 4. Estes foram escolhidos por haver diversos artigos importantes [10,22,43,67] que os utilizam para comparação e análise, e também pelo fato de que são distribuídos junto com simulador de redes NS2 (com excessão do OLSR, que ainda não faz parte do pacote).

2.2.3. DSDV – Destination-Sequenced Distance Vector

Destination-Sequenced Distance Vector (DSDV), descrito em [65], é um protocolo de roteamento que utiliza tabelas (*table-driven*) e é baseado no algoritmo de vetor de distâncias (Distributed Bellman-Ford [3]), com a vantagem de não conter *ciclos* no roteamento, o que é feito com a introdução de números de seqüência. Este é um dos primeiros protocolos desenvolvidos para redes ad hoc.

Cada nó da rede contém uma tabela de roteamento para todos outros nós alcançáveis, e para cada entrada na tabela são armazenados um número de seqüência e o próximo salto. Esta tabela é transmitida pelo nó para todos seus vizinhos, rotas com número de seqüência mais recente são guardadas e as antigas descartadas. Este é um protocolo **pró-ativo** que atualiza as tabelas de tempos em tempos ou quando alguma mudança na topologia da rede é detectada (i.e. alteração nos vizinhos). A figura 2.2.1 mostra a tabela de roteamento do nó A. Caso A queira transmitir dados para G, irá primeiramente enviar para o próximo nó da lista, que no caso é o C.



(a) Topologia da rede

Dest.	Próx.	Dist.	Seq.
B	B	1	18
C	C	1	24
D	D	1	60
E	C	2	92
F	C	3	124
G	C	3	196

(b) Tabela de roteamento do nó A

Figura 2.2.1.: Estabelecimento de rotas no DSDV

Características:

Por depender de transmissões periódicas, a utilização das rotas precisa esperar até que as tabelas de roteamento estejam completas. Este tempo de convergência pode ser negligênciado em redes estáticas com fio, nas quais a topologia não se altera com tanta frequência. Por outro lado, nas redes ad hoc este tempo de convergência pode significar a perda de diversos pacotes, pois as rotas válidas demoram para ser difundidas. Outro problema é o alto sobrecusto adicionado por essas transmissões periódicas.

2.2.4. DSR – Dynamic Source Routing

Dynamic Source Routing (DSR) [44] é um protocolo *reativo (sob demanda)*, e se utiliza da técnica de *Source Routing* (cada pacote carrega consigo a rota completa até o destino). Diferentemente do DSDV, que é um protocolo pró-ativo, o DSR não envia mensagens periódicas para obter as rotas.

Este protocolo trabalha sob demanda, a rota até o destino é descoberta antes do dito envio através do mecanismo RouteDiscovery. Estas rotas recém descobertas são armazenadas em um *cache* e podem ser aproveitadas oportunisticamente por um nó que está roteando algum pacote em trânsito. Se alguma conexão quebrada é detectada, uma mensagem RouteError é transmitida através da rede e um mecanismo de manutenção de rotas é ativado para tentar consertar os defeitos, se possível.

São necessários portanto, dois modos de operação, o de descoberta de rotas RouteDiscovery e o de manutenção Route Maintenance. Quando determinado nó quer enviar dados, uma mensagem de RouteRequest (RREQ) contendo o destino, um número de sequência e um tempo de vida (TTL) é difundida na rede. Cada pacote que recebe uma mensagem desta, procura em seu *cache* se esta rota já existe, caso não exista, o endereço do nó atual é adicionado ao pacote, que é então passado adiante a todos vizinhos.

Ao chegar no destino, uma mensagem de retorno RouteReply (RREP) é enviada à origem, utilizando o caminho inverso percorrido e armazenado no pacote (veja figuras 2.2.2 e 2.2.3). O número de seqüência no pacote é utilizado para evitar ciclos e também para evitar múltiplas retransmissões de um mesmo pedido de rota (RREQ).

A manutenção de rotas é utilizada quando um nó descobre que determinada rota está desatualizada, então um pacote do tipo RouteError é enviado à origem. Todos as rotas que contém o enlace quebrado serão removidas do cache.

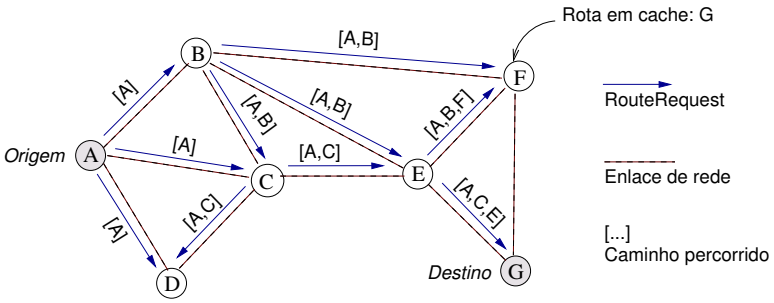


Figura 2.2.2.: Exemplo de requisição de rota no DSR. O nó A pretende enviar dados para G, então difunde a mensagem RREQ pela rede.

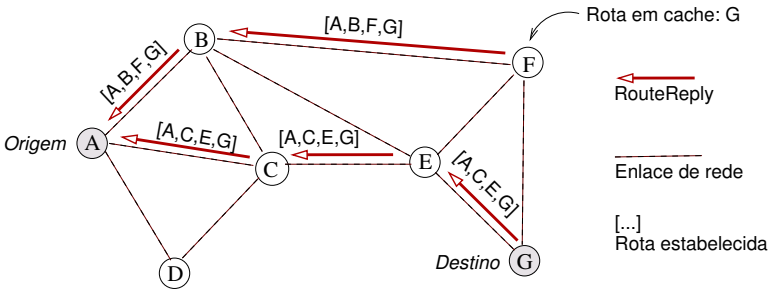


Figura 2.2.3.: Exemplo de resposta de uma requisição. O nó G respondem um RREP para A, usando a rota contida no pacote recebido. O nó F responde RREP para A usando a rota armazenada em *cache*

Características:

Uma característica deste protocolo é o uso do Roteamento na Origem *(Source Routing)*, o que faz com que os nós intermediários não precisem ter a rota atualizada para poder retransmitir os dados, basta então obter esta informação no cabeçalho do pacote. Também não é necessário o envio periódico de pacotes de controle (do tipo HELLO) para manter rotas, o que é importante em períodos de inatividade da rede. Uma vantagem deste protocolo é que nós podem aprender rotas ao retransmitir pacotes que estão de passagem. Estas rotas descobertas de forma oportunística são armazenas em um *cache*.

2.2.5. AODV – Ad hoc On-demand Distance Vector

Ad hoc On-demand Distance Vector (AODV) [66] é um dos protocolos mais discutidos e avançados para este tipo de rede e está num estágio bem maduro. Ele é baseado no DSDV, mas sem a necessidade de envios periódicos de dados, e trabalha de uma maneira **reativa** (sob demanda).

Se uma rota até um certo nó é desconhecida, um processo de descoberta é iniciado. Este processo consiste em difundir a mensagem ROUTE REQUEST (RREQ) pela rede, contendo o id e número de sequência da origem, o id e número de sequência do destino de destino e um id de atualização (figura 2.2.4).

Para que a rede não seja inundada com mensagens deste tipo, a técnica *'expanding ring search'* é utilizada: o tempo de vida (TTL) dos pacotes começam com um valor baixo e enquanto o destino não for encontrado, o pacote é retransmitido com um valor de TTL maior; Cada nó que repassa um pedido de RREQ adiciona os dados de origem, destino e números de sequência em uma tabela a fim de criar um caminho reverso (figura 2.2.5); se o pacote chegar ao destinatário ou a algum nó que contenha a rota, este irá responder com um ROUTE REPLY (RREP) à origem. Os nós intermediários armazenam em um cache a rota recém descoberta. Uma entrada na tabela de roteamento consiste de um destino, o próximo salto e um número de seqüência. As rotas são atualizadas se o número de seqüência da mensagem de atualização for mais recente. Estas entradas na tabela também têm um tempo de vida.

Características:

Este protocolo também não gera *ciclos* no roteamento e uma perda de comunicação em um enlace de rede é notificada imediatamente. O algoritmo também prevê o envio

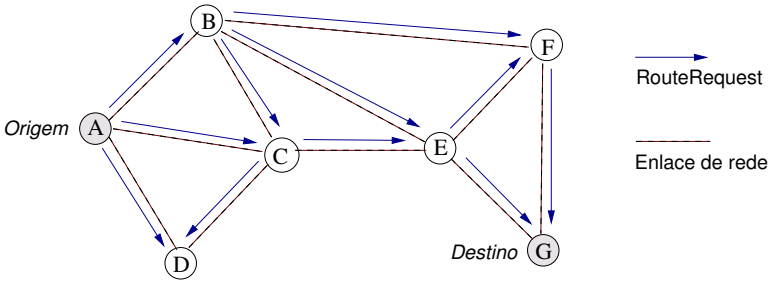


Figura 2.2.4.: Exemplo de requisição de rota no AODV. O nó A pretende enviar dados para G, então difunde a mensagem RREQ pela rede.

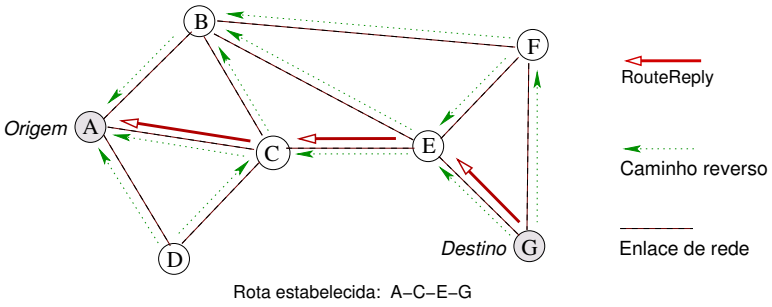


Figura 2.2.5.: Exemplo resposta no AODV. O nó G responde RREP para A utilizando caminhos reversos e estabele a rota de envio A-C-E-G.

pelos nós de mensagens periódicas de notificação (HELLO) a todos os vizinhos. Estas mensagens servem para notificar os vizinhos de que determinado nó está ativo e que as rotas armazenadas em cache para o nó em questão podem continuar sendo utilizadas. Caso o nó pare de receber mensagens de notificação de um vizinho anteriormente ativo, então a entrada deste vizinho na tabela é removida e uma mensagem de enlace inválido (*broken link*) é enviada para os nós que utilizam esta rota.

A vantagem deste protocolo em relação aos demais (que utilizam vetor de distâncias e estado do enlace) é que o número de mensagens de controle é bem reduzido. Esta característica é importante numa rede em que a topologia se altera com frequência. A diferença mais visível em relação ao DSR é que naquele a rota completa a ser percorrida está contida no pacote, já no AODV, cada nó mantém em uma tabela o próximo salto para aquela rota.

Outra característica deste protocolo é que ele armazena somente *uma* rota para cada destino.

2.2.6. OLSR – Optimized Link State Routing

Optimized Link State Routing (OLSR) [40] é um protocolo **pró-ativo** que utiliza a técnica de **link state**.

O seu funcionamento é baseado na eleição de alguns nós especiais, chamados *"MultiPoint Relays" (MPRs)*, com o objetivo de diminuir a quantidade de informação redundante na descoberta de rotas para posteriormente utilizar a técnica de inundação. Cada nó da rede irá escolher um conjunto de MPRs dentre seus vizinhos através do envio de mensagens periódicas do tipo HELLO com alcance de 2 nós, de posse da lista de vizinhos, o nó calcula um conjunto mínimo de nós (MPRs) dentre os vizinhos de primeiro nível, de tal maneira que qualquer um dos vizinhos de segundo nível seja alcançável através destes MPRs. As figuras 2.2.6 e 2.2.7 ilustram a escolha destes nós e a vantagem de sua utilização. Note que o cálculo do conjunto mínimo de MPRs é NP-completo, para tanto o problema é resolvido com o uso de heurísticas.

Características:

Os MPRs são utilizados para melhorar a eficiência na construção das tabelas de roteamento. Este protocolo tem se mostrado bem eficiente em alguns experimentos e simulações, principalmente em redes densas e de larga escala.

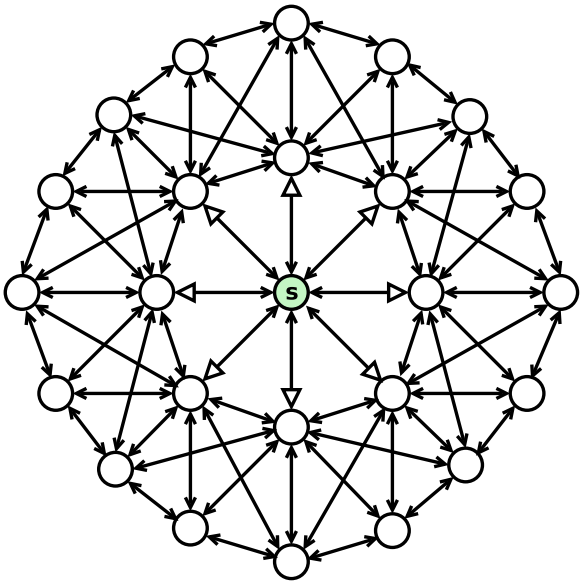


Figura 2.2.6.: Envio de um pacote usando inundação. As flechas mostram as transmissões.

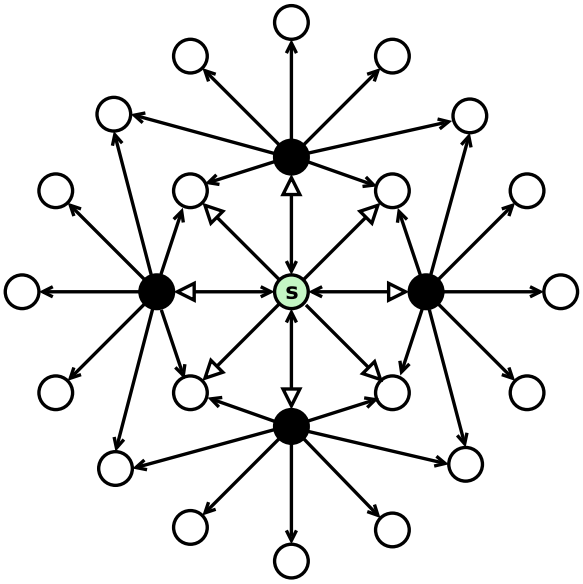


Figura 2.2.7.: Inundação de um pacote a partir do nó central utilizando MPRs (nós pretos). As flechas mostram as transmissões.

2.3. Rede de Sensores Sem Fio

Com os avanços da tecnologia, principalmente na área de redes sem fio e micro-eletrônica, tornou-se possível o desenvolvimento de equipamentos micro-processados com capacidade de comunicação sem fio de dados, baixo consumo de energia, tamanho extremamente reduzido e baixo custo de produção. Nestes dispositivos são acoplados sensores e/ou atuadores de diversos tipos e finalidades (ex: temperatura, pressão, infravermelho, químico, acústico, sísmico, etc.) para monitorar eventos ou executar ações. Estes equipamentos são chamados *Nós Sensores* que, em resumo, são capazes de realizar o sensoriamento de alguma variável do ambiente, o processamento e a comunicação sem fio de dados.

Uma Rede de Sensores Sem Fio (RSSF - *Wireless Sensor Networks*) [1, 18, 21, 54, 75, 78] é composta por centenas ou milhares de nós sensores inter-conectados que colaboram entre si para realizar alguma tarefa específica como, por exemplo, detecção de incêndios em florestas ou controle do sistema de ar condicionado em grandes edifícios.

Como são instalados em grande escala, e de forma não planejada, é possível realizar o sensoriamento remoto de uma grande região física e de difícil acesso [24]. Na maioria dos casos, o ambiente a ser monitorado não têm infra-estrutura para obtenção de energia ou comunicação de dados, portanto, a necessidade de otimizar o consumo de energia é um fator primordial.

As RSSFs podem por ser tratadas como um tipo específico de MANET, descrita na seção anterior, sendo que ambas trocam dados diretamente entre si, diferentemente das redes tradicionais sem fio, as quais utilizam estações rádio base fixas para estabelecer a comunicação com as unidades móveis (nós da rede). Por esta razão, a utilização de RSSFs está apoiada, muitas vezes, nas técnicas das MANETs, mas muitos dos protocolos e algoritmos propostos para estas redes não podem ser diretamente utilizados em RSSFs, pois possuem uma relativa sobrecarga de processamento e armazenamento, fatores estes que motivam a pesquisa de algoritmos e protocolos específicos para uso nas RSSFs, que possuem fortes restrições na utilização de recursos computacionais.

As principais características que diferenciam uma RSSF de uma MANET são:

- Maior limitação na quantidade de energia, memória e recursos computacionais;
- Formação de redes densas e de larga escala;
- Os nós sensores são ainda mais suscetíveis a falhas;

- Grande cooperação entre os nós sensores;
- Possibilidade dos nós sensores não possuírem identificadores globais;
- Os protocolos e algoritmos utilizados deverão ter maior capacidade de auto-organização.

2.3.1. Aplicações

São inúmeras as possibilidades de utilização desta nova tecnologia [25]. Esta cada vez mais claro que esta é uma área que irá revolucionar o processamento e a obtenção de informações remotas. A seguir podemos citar alguns exemplos de uso:

- [**Meio ambiente**] - monitoramento da qualidade do ar; comportamento da população de espécies animais e vegetais; monitoramento de variáveis diversas em florestas; detecção de queimadas, desmatamentos e enchentes; mapeamento da biocomplexidade do meio ambiente.
- [**Militar**] - rastreamento de tropas; monitoramento das tropas amigas, equipamentos e munição; detecção de ataques químicos ou biológicos.
- [**Saúde**] - monitoramento de pacientes; controle no uso de remédios em hospitais; rastreamento e monitoramento de médicos e pacientes dentro do hospital.
- [**Residência**] - automação de equipamentos; comunicação entre eletrodomésticos.
- [**Tráfego**] - monitoramento de automóveis, detectando furtos, monitoramento de vias, estacionamentos, etc.
- [**Aviação**] - sensoriamento de variáveis diversas durante vôos de ensaio.

Uma característica importante em Redes de Sensores sem Fio é a sua arquitetura de comunicação que está baseada na alta colaboração entre os nós sensores. Cada nó sensor espalhado em um campo de sensoriamento (figura 2.3.1) tem a capacidade de coletar dados (periodicamente ou por requisição), processá-los e, através do roteamento multi-saltos *(multi-hop)* realizado de forma colaborativa entre os nós sensores vizinhos, enviá-los a um nó de destino. Este nó, que normalmente é chamado de sorvedouro (*Sink node*), possui mais recursos que o restante dos nós da rede e pode, então, disponibilizar estes dados de alguma maneira como, por exemplo, através da Internet.

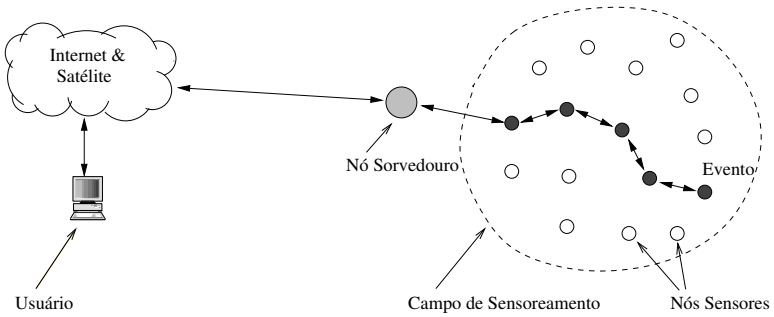


Figura 2.3.1.: Nós sensores espalhados em um campo de sensoriamento

Na prática, os protocolos de acesso ao meio (MAC - *Medium Access Control*) utilizados em Redes de Sensores sem Fio seguem os padrões IEEE 802.11 [37] e Bluetooth [6]. Mas, pesquisas atuais também usam métodos alternativos como, por exemplo, a utilização do meio ótico estudada no projeto *Smart Dust* [46].

2.3.2. Protocolos de Roteamento

Protocolos de roteamento em MANETs convencionais possuem diversas limitações quando utilizados em RSSF, devido à própria natureza destas redes, que pelo comportamento dinâmico e pela restrição de armazenamento de dados, não podem guardar grande quantidade de informações, como, as usuais tabelas de roteamento.

Um dos objetivos principais dos protocolos de roteamento em RSSF é minimizar o consumo global de energia da rede, existindo uma grande preocupação em não deixar áreas da rede incomunicáveis (desconexas). Portanto, um dos fatores importantes na construção de algoritmos é a equalização do consumo de energia entre os nós, não permitindo que uma dada região tenha uma prematura perda de comunicação [34, 36].

2.4. Redes de Satélites de Órbita Baixa (LEO)

Uma rede de satélites de órbita baixa (LEO – *Low Earth Satellite*) [30] consiste em um conjunto de satélites orbitando à terra em velocidade alta e constante (completam uma órbita em apenas 90 minutos), e a uma altura relativamente baixa (podendo chegar a alguns milhares de kilometros – entre 200 e 2000km). Como não são geo-estacionários, uma rede (ou "constelação") é necessária para prover uma cobertura completa às regiões atendidas.

Como vantagem tem-se que a proximidade com a terra reduz os atrasos na comunicação e na energia necessária para comunicação entre usuários na terra e os satélites.

Por outro lado, duas questões importantes aparecem em relação à baixa altitude: (1) cada satélite só consegue cobrir uma pequena área geográfica (denominada *pegada*) na superfície da terra; (2) também devido à proximidade, a *pegada* se movimenta rápida e constantemente, implicando em alta mobilidade do sistema.

No desenvolvimento do sistema de satélites, a quantidade, as direções e as altitudes das órbitas escolhidas podem variar muito. Dentre as diversas órbitas possíveis é interessante escolher uma em que os enlaces entre-satélites (ISLs – *Inter-Satellite Links*) permaneçam ativos pelo maior tempo possível. A figura 2.4.1 ilustra a topologia da rede de satélites Iridium, na qual, a cada instante metade dos satélites estão se movimentando da direção sul-norte e a outra metade está no sentido oposto, norte-sul. Essa composição faz com que na maior parte do tempo os satélites se movimentem lado a lado, mas ao cruzar o polo norte ou o polo sul as suas posições relativas se alteram.

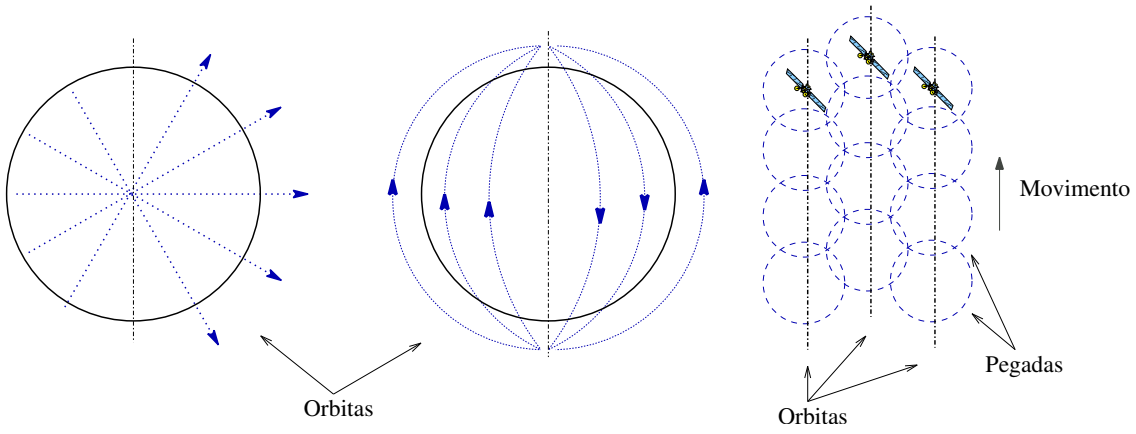


Figura 2.4.1.: Estrutura de uma constelação de satélites: à esquerda, vista do polo norte e ao centro, do equador. À direita as órbitas de satélites adjacentes e suas *pegadas*.

Essas características mostram a topologia dinâmica deste tipo de rede, o que nos leva a questões interessantes em relação ao roteamento de dados entre os pares de satélites [17].

2.5. Redes Tolerantes a Falhas e Desconexões (DTN)

As redes descritas nas seções anteriores (MANET, RSSF e LEO) têm em comum a topologia dinâmica, os atrasos longos ou variáveis, prevalência de desconexões, conectividade intermitente, alta taxas de erros, limitação de recursos (e.g. memória, bateria). Com isso, a troca de mensagens entre pares de nós nestas redes é dificultada por estes não serem alcançáveis em todos os instantes. Contudo, mesmo não existindo um caminho fim-a-fim entre determinado par origem–destino, é possível realizar a comunicação entre esses nós se os dados permanecerem armazenados por alguns instantes durante o percurso do roteamento multi-saltos [81].

Convencionou-se denominar a classe de redes com estas características específicas de Redes Tolerantes a Atrasos e Desconexões (DTN – *Delay and Disruption Tolerant Networks*) [26, 35, 45, 63, 64, 81].

Uma DTN é formada por regiões e mensageiros. As regiões correspondem a agrupamentos de nós que estão ao alcance um do outro (i.e. existe caminhos fim-a-fim) e mensageiros são os nós que fazem parte de mais de uma região e podem realizar a tarefa de "mula de dados", enviando e trazendo dados de uma região a outra. A maneira, a quantidade e os horários dos mensageiros que devem ser escolhidos são um dos tópicos de pesquisa desta área.

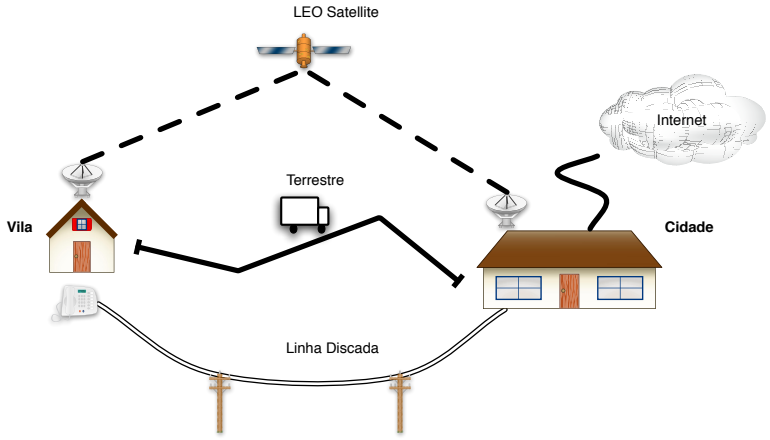


Figura 2.5.1.: Cenário ilustrando diversas conexões entre duas cidades

A figura 2.5.1 mostra uma vila hipotética servida por diferentes meios de comunicação (mensageiros): uma linha discada para acesso à internet, uma conexão por satélite e via

terrestre. A rede de satélites (e.g. PACSAT), tem largura de banda e latência moderadas, em torno de 10Kbps, e é visível de 4-5 períodos de tempo (passagens) por dia, permanecendo visível 10 minutos em cada passagem. A comunicação por via terrestre tem banda larga, com alta latência e está disponível em alguns momentos apenas, já a linha discada tem largura de banda estreita, com baixa latência e está sempre disponível. A escolha um desses meios de comunicação depende de vários fatores, que incluem a origem, o destinatário, o tamanho, o horário, a disponibilidade dos mensageiros, etc.

O grupo de pesquisas em Redes Tolerantes a Atrasos (Delay Tolerant Networking Research Group - DTNRG)[3] do Internet Research Task Force (IRTF) disponibiliza uma série de documentos de pesquisa, incluindo a proposta de uma arquitetura DTN definida em um Internet Draft [4]. Essa arquitetura é uma evolução da proposta do Projeto Internet Interplanetária [13, 16] e descreve como um conjunto de nós se organiza para armazenar e encaminhar mensagens cooperativamente.

Resumo

Neste capítulo, foi apresentada um introdução sobre a pesquisa na área de MANETs, RSSFs, LEOs e DTNs, temas que motivam a pesquisa na área de roteamento de dados. A seguir serão mostrados os conceitos de grafos evolutivos, os quais são úteis na elaboração de protocolos de roteamento ótimos para essas redes dinâmicas.

[3] http://www.dtnrg.org/
[4] http://www.ietf.org/internet-drafts/draft-irtf-dtnrg-ltp-motivation-04.txt

3. Grafos Evolutivos

A grande maioria dos estudos sobre otimização do roteamento de dados encontrados na literatura hoje estão relacionados a redes estáticas. Com isso conceitos já estabelecidos nestas redes, tais como, conectividade, árvores geradoras e caminhos de menor custo, foram revistos para redes dinâmicas e um dos resultados é o modelo teórico de *grafos evolutivos* (EG – *Evolving Graphs*).

Nas próximas seções serão apresentados o conceito de redes com comportamento previsível, que é a motivação para o nosso trabalho, juntamente com a teoria de grafos evolutivos e seus algoritmos de custo mínimo (jornada que chega mais cedo, jornada mais curta e jornada mais rápida).

3.1. Redes com Comportamento Previsível

Como visto nas seções anteriores, as redes móveis ad hoc (MANETs) tem um comportamento altamente dinâmico. A comunicação direta entre os nós da rede, a liberdade de movimentação e a não uniformidade de seus participantes tornam o desenvolvimento de algoritmos de roteamento eficientes uma tarefa complexa. No entanto podemos destacar alguns sub-tipos de MANET, nas quais é possível capturar a dinâmica das alterações na topologia da rede, este é o caso das redes de satélites (LEO – *Low Earth Satellite*), algumas Redes de Sensores Sem Fio (RSSF) e outras em que a topologia da rede possa ser pré-determinada no decorrer do tempo. Redes com esta característica foram denominadas: **Redes com Comportamento Previsível** (FSDN – *Fixed Schedule Dynamic Network*) [29, 41] e são o objeto de estudo neste trabalho.

No caso dos sistemas de satélites LEO [11, 30, 77], os nós se comunicam através de conexões entre os satélites que estão ao alcance um do outro. Enquanto as conexões

estão no mesmo plano orbital (*intraplanar*) a topologia não se altera, pois os satélites tem velocidade angular relativa zero entre si, mas as conexões em diferentes planos orbitais (*interplanar*) fazem com que a topologia se modifique quando os satélites se aproximam ou se afastam um dos outros. O resultado disso é uma rede com uma topologia dinâmica, contudo, as trajetórias dos satélites são previamente conhecidas e portanto a topologia da rede no tempo também o é. Esse determinismo pode ser explorado para otimizar estratégias de roteamento.

Certas redes de sensores sem fio também possuem características similares às redes FSDN, devido às suas limitações de energia, os nós destas redes podem ser programados para se desligar por certos períodos de tempo, com o intuito de economizar recursos. Definidos os horários de funcionamento dos sensores, é possível pré determinar as alterações da topologia da rede no tempo.

Este comportamento determinístico também é estudado pelas redes DTN [41], pois a questão da desconexão da rede no tempo requer protocolos específicos para lidar com a topologia dinâmica.

3.2. Conceitos

A noção de *grafos evolutivos*, introduzida recentemente [29], consiste basicamente em formalizar um domínio no tempo em grafos. Surpreendentemente, isto nos leva a inúmeras questões interessantes na teoria dos grafos, com aplicações imediatas no controle da topologia de MANETs, no qual incluem-se as RSSFs, relacionando às propriedades de roteamento ótimo e conectividade no tempo. *Grafos evolutivos* representam uma abstração formal de redes dinâmicas (veja exemplo na figura 3.2.1 e 3.2.2), mais precisamente:

> Um *grafo evolutivo* é uma seqüência indexada de τ sub-grafos de um dado grafo, onde um sub-grafo de um certo índice corresponde à rede de conectividade no intervalo de tempo indicado pelo número do índice.

O domínio no tempo é posteriormente incorporado ao modelo com a introdução do conceito de *jornadas*, equivalente a *caminhos no tempo*. As *jornadas* permitem a comunicação entre dois nós mesmo que estes não estejam ativos no mesmo instante no tempo (i.e. no mesmo sub-grafo). Com isso , uma *jornada* em um *grafo evolutivo* é um caminho nos grafos subjacentes nos quais o horário de existência das arestas estão em ordem não-decrescente.

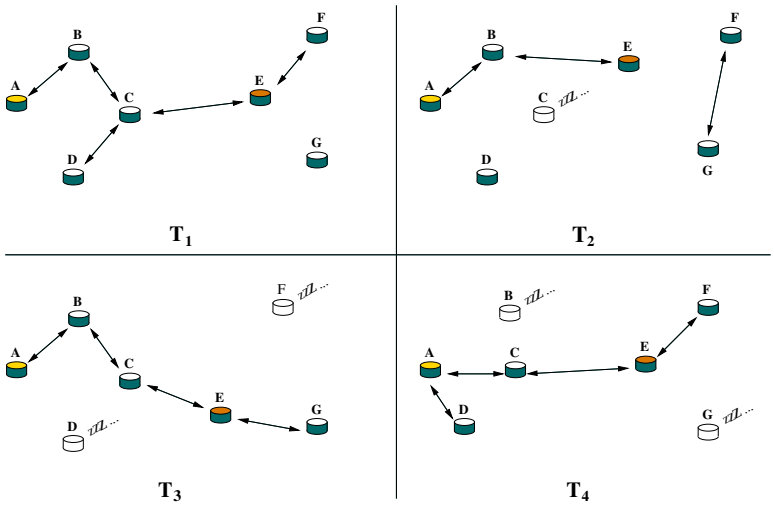


Figura 3.2.1.: Evolução de uma rede MANET no tempo. Os índices de **T** correspondem à sucessivos momentos da rede. Os Zzz indicam nós dormindo.

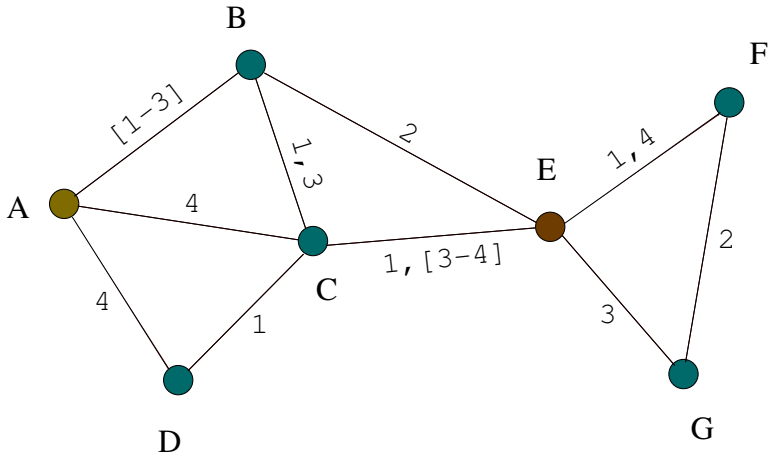


Figura 3.2.2.: Grafo evolutivo correspondente à MANET da figura 3.2.1. As arestas estão nomeadas com os intervalos correspondentes à existência de enlaces entre os nós. Observe que **E,G,F** não é uma jornada válida, pois a aresta **{G,F}** existe somente no passado em relação à aresta **{E,G}**

Pode-se ver facilmente na figura 3.2.2 que **{D,C,E,F,G}** é uma *jornada*, assim como **{D,C,E,G}** também o é, e tem um número menor de saltos, mas a informação só chega em **G** em um horário posterior (intervalo 3 ao invés de 2). Note que neste exemplo o

tempo para percorrer uma aresta é zero (instantâneo). Portanto, dados enviados por **A** no instante 1 podem chegar ao nó **F** no mesmo instante 1.

Minimizar a quantidade de saltos ou o horário de chegada são exemplos das otimizações que podem ser realizadas utilizando este modelo.

Assim como toda teoria em redes de comunicação sem fio, o conceito de *grafos evolutivos* é muito recente, então vemos a necessidade de desenvolvimento de modelos de referência. Os primeiros estudos sobre *grafos evolutivos* [11, 28, 29, 31, 58, 60], mostraram ser uma relevante ferramenta de modelagem para redes dinâmicas.

3.3. Métricas de Otimização

Com o intuito de desenvolver algoritmos de roteamento eficientes, três métricas de otimização já foram formalizadas até então nos estudos de *grafos evolutivos*. As *jornadas* de custo mínimo (*least cost journeys*) explicitadas em [11, 29] compreendem a elaboração de algoritmos ótimos em relação às métricas: Jornada que Chega Mais Cedo – menor horário de chegada (*Foremost Journeys*); Jornada Mais Curta – menor número de saltos percorridos (*Min-hop Count*); Jornada Mais Rápida — menor tempo gasto no percurso (*Fastest Journeys*). Cada uma delas pode ser calculada em tempo polinomial [11].

A seguir apresentamos um exemplo mais intuitivo para ilustrar as três métricas: imagine um trabalhador que realiza o trajeto casa–trabalho no período da manhã em uma grande metrópole (veja figura 3.3.1). Ao sair de casa às 7:00h, o trajeto dura 1h e 15min ele chega ao trabalho às 8:15h, precisando trocar de ônibus 1 vez. Mas, se ele sair de casa às 9:15h, quando há muito menos tráfego, o mesmo trajeto pode ser feito em 45min e irá chegar às 10:00h, mesmo que para isso precise trocar de ônibus duas vezes. Note que na primeira *jornada* o trabalhador chega mais cedo, e na segunda, mais rápido embora faça um número maior de paradas. Suponha, que exista um ônibus fretado, que pegue-o em casa às 7:30h e deixe-o no trabalho às 9:00h, sem precisar trocar de ônibus, esta seria a Jornada Mais Curta, que no entanto leva mais tempo para ser percorrida do que a Jornada Mais Rápida (1:30h ao invés de 45min) e chega depois da Jornada que Chega Mais Cedo (9:00h ao invés de 8:15h).

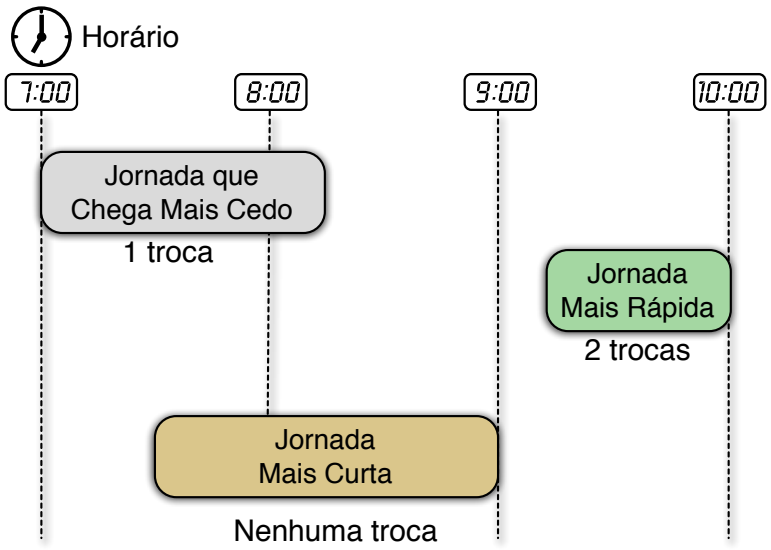


Figura 3.3.1.: Três métricas de otimização em grafos evolutivos

3.3.1. Jornada que Chega Mais Cedo (Foremost Journey)

Jornada que Chega Mais Cedo também pode ser entendida como "menor horário de chegada" e é utilizada para calcular os trajetos em que um pacote partindo de s no instante t chega ao nó u no horário mais cedo possível, independente do número de nós percorridos. Este algoritmo é uma adaptação do conhecido algoritmo de Dijkstra para caminhos de custo mínimo [19], mas com a inclusão de uma lista de horários nas arestas.

Lembrando que, para calcular os caminhos de custo mínimo, o algoritmo de Dijkstra procede construindo um conjunto C de vértices *fechados*, para os quais o caminho mais curto já foi calculado, dessa maneira, escolhendo um vértice $u \notin C$ com o custo $d(u)$ mínimo e adicionando u em C, i.e. *fechando* u.

Neste ponto, todos as arestas de u a $V - C$ estão *abertas*, i.e. elas já foram examinadas e seus respectivos custos mínimos, d, já foram atualizados para todos os destinos. Para se ter um acesso rápido ao caminho de menor custo estimado, o algoritmo mantém uma fila de prioridades (*heap*) Q com todos os vértices em $V - C$, com a chave no custo d. Note que d é inicializado com ∞ para todos os vértices, com exceção da origem s que tem custo $d = t$ (t é o horário atual do nó).

Uma das características do método de Dijkstra é que o custo de caminhos que são prefixos

de um caminho de custo mínimo, também são caminho de custo mínimo. Infelizmente, o prefixo de uma Jornada que Chega Mais Cedo não é necessariamente uma Jornada que Chega Mais Cedo. Como exemplo, considere o EG na figura 3.2.2, uma mensagem enviada de **A** para **G** no momento de índice 1 pode utilizar o trajeto **{A,B,E,G}**. O pacote irá chegar em **G** no momento de índice 3. Note que o trajeto **{A,B,E}** que é prefixo do anterior irá chegar em **E** no momento 2, mas **{A,B,E}** não é uma Jornada que Chega Mais Cedo entre **A** e **E**, que seria **{A,B,C,E}** e chegaria ao destino no momento 1, mas percorrendo um número maior de nós.

No entanto, foi provado em [11, 28] que existe pelo menos uma Jornada que Chega Mais Cedo no EG que satisfaz a propriedade anterior.

Para se calcular a Jornada que Chega Mais Cedo partindo de s no tempo t a todos outros nós utiliza-se uma adaptação direta do algoritmo de Dijkstra, a seguir temos um resumo do funcionamento do algoritmo, que é detalhado em [11, 28].

Algoritmo Jornada que Chega Mais Cedo

---

1. Inicialize $d(s) = t$, e $d(u) = \infty$ para todos outros nós.
2. Inicialize *min-heap* Q, ordenado por d, com somente s no topo do heap.
3. Enquanto Q $\neq \emptyset$ faça:
   a) $x \leftarrow$ raiz do heap Q.
   b) Remova a raiz do heap Q.
   c) Para cada vizinho aberto v de x faça:
      i. Calcule o primeira aresta válida com horário de existência maior ou igual ao horário atual t.
      ii. Insira v no heap Q, se ele não estiver lá ainda.
      iii. Se necessário, atualize $d(v)$ e suas chaves.
   d) Atualize o heap Q.
   e) Feche x. Insira-o na árvore de Jornadas que Chegam Mais Cedo.

Ao final da execução do algoritmo tem-se uma árvore geradora com os trajetos a serem percorridos pelas Jornadas que Chegam Mais Cedo partindo de s (no tempo t) a todos os outros nós.

Note que o cálculo da primeira aresta válida feita no laço interno (linha 3(c)i) deve levar em conta o tempo necessário para realizar o percurso daquela aresta (i.e. a duração da transmissão). Este seria o caso dos grafos evolutivos temporais, em que cada aresta pode ter um tempo de percurso distinto.

A Jornada que Chega Mais Cedo de s a todos os outros nós pode ser calculada em tempo $O(M(\log \delta_E + \log N))$, sendo N o número de vertices (nós), M o número de arestas (enlaces) e δ_E o maior número de intervalos de uma aresta dentre todas as arestas. O termo $\log \delta_E$ surge na procura pela aresta com menor horário de existência dentre todos intervalos daquela aresta (linha 3(c)i do algoritmo)

O protocolo de roteamento originado pelo algoritmo acima, que a partir deste ponto será referenciado por $\text{EG}_{foremost}$, está detalhado na Seção 3.5

3.3.2. Jornada Mais Curta (Min-hop Count Journey)

A métrica de Jornada Mais Curta pode ser entendida como "número mínimo de saltos" e compreende a elaboração de algoritmos em que o número de nós percorridos no roteamento de um pacote da origem s ao destino v é o menor possível.

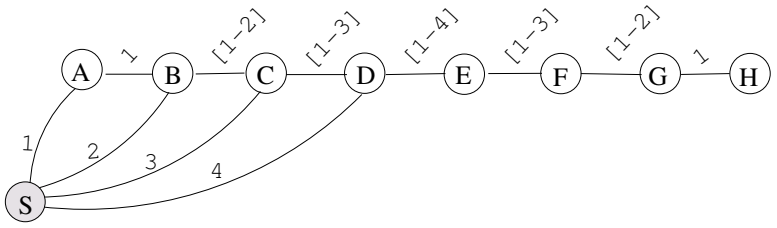


Figura 3.3.2.: Jornada Mais Curta de **S** a **H** tem 8 saltos e chega no tempo 1, enquanto a Jornada Mais Curta de **S** a **D** tem 1 salto mas chega no tempo 4.

A dificuldade em calcular este trajeto se deve ao fato do algoritmo levar em conta o tempo do percurso de uma aresta, e também que prefixos de Jornadas Mais Curtas não são necessariamente Jornadas Mais Curtas, como exemplo considere o EG da Fig. 3.3.2, uma mensagem enviada no índice de tempo 1 de **S** para **H** pode usar a jornada **{S,A,B,C,D,E,F,G,H}**, chegando em **H** no tempo 1 e percorrendo 8 saltos. Se pegarmos

o prefixo **{S,A,B,C}** desta jornada, ele irá chegar em **C** no instante 1 em 3 saltos, embora está não seja a JORNADA MAIS CURTA de **S** a **C**, que é, no entanto, a jornada **{S,C}**, com número de saltos igual a 1, chegando em **C** no instante 3.

Para calcular as JORNADAS MAIS CURTAS partindo de s no instante t a todos os outros nós, utilizaremos o algoritmo abaixo (detalhado em [11]):

Algoritmo Jornada Mais Curta

Entrada: Um grafo evolutivo G, um vértice $s \in V_G$ e o instante atual t_{now}.
Saída: A árvore de caminhos curtos T, um vetor $location$: $V_G \to T$.
Variáveis: A árvore T de pares $(u,t) \in V_G \times \Re_+^*$, um inteiro d e um vetor $earliest : V_G \to \Re_+^*$.

1. Inicialize $T = (s, t_{now})$.
2. $earliest(s) = t_{now}$, e $earliest(u) = \infty$ para todos os outros.
3. $d = 1$.
4. $location(s) = (s, t_{now})$.
5. Enquanto existir $u \in V_G$ tal que $location(u)$ não esteja definido, faça:
   a) Para todos os pares (u,t) na árvore na profundidade d, faça:
      i. Pegue todos os pares de vizinhos (v,t') de (u,t).
      ii. Se $location(v)$ não está definido, então $location(v) = (v,t')$.
      iii. Se $earliest(v) > t'$, então $earliest(v) = t'$ e (v,t') é filho de (u,t) em T.
   b) $d = d + 1$.
6. devolva a árvore T e o vetor $location$.

Neste algoritmo, a árvore T representa as JORNADAS MAIS CURTAS de (s, t_{now}) para os pares (u,t), e a posição $location(u)$ aponta para o local correto na árvore para obter a JORNADA MAIS CURTA de s a u, dado que um vértice pode aparecer mais de uma vez na árvore (em diferentes instantes).

O vetor *earliest* mostra o menor instante em que o vértice u é alcançado na árvore. É importante notar que, se existe uma *jornada* de comprimento maior que k saltos de s a u, essa *jornada* será relevante se o horário de chegada em u for menor do que o horário de chegada da *jornada* com número de saltos igual a k.

Depois da árvore estar construída, a obtenção da Jornada Mais Curta é feita da seguinte forma: como $location(u)$ aponta para um par (u, t) na árvore T, obtém-se a *jornada* percorrendo o caminho reverso do pai de (u, t) em T.

A complexidade deste algoritmo é $O(MD)$ [11], sendo M o número de arestas do grafo e D o diâmetro de G (D será também a altura máxima da arvore T). O protocolo de roteamento originado a partir deste algoritmo será referenciado como $\text{EG}_{shortest}$ e será detalhado na Seção 3.5.

3.3.3. Jornada Mais Rápida (Fastest Journey)

O objetivo nesta métrica é otimizar o tempo gasto no percurso entre a origem s e o nó u. Esta é a métrica mais complexa a ser calculada dentre as três estudadas. O trajeto mais rápido pode estar bem a frente no tempo (chegar mais tarde do que Jornada que Chega Mais Cedo) e, talvez, tenha um número maior de saltos se comparado com a métrica Jornada Mais Curta.

Devido a limitações de tempo, este algoritmo não foi abordado neste trabalho. Restringimos nossos esforços na implementação e simulações dos dois anteriores.

3.4. Estrutura de Dados

Uma premissa importante nos algoritmos de grafos evolutivos mostrados neste trabalho é a estrutura compacta de armazenamento da topologia da rede no tempo. Esta estrutura, que é de certa forma simples, possibilita a elaboração de algoritmos polinomiais sem que haja a necessidade de discretizar o tempo em períodos longos.

A estrutura de dados para o armazenamento dos nós, arestas e seus respectivos intervalos de existência pode ser definida como:

> Cada nó u contém a lista de seus vizinhos, sendo que para cada vizinho v é anexado uma lista de adjacência com os horários de existência das arestas (u,v), ordenadas pelos horários de início dos intervalos de tempo em que estas arestas estão presentes.

O tempo de percurso para cada enlace pode ser armazenado junto com os dados de cada vizinho. Além disso, para cada vértice da lista, também pode-se adicionar os horários em que eles próprios estão ativos.

3.5. Protocolos de Roteamento: $\mathsf{EG}_{foremost}$ e $\mathsf{EG}_{shortest}$

A seguir é apresentada uma descrição do funcionamento do protocolo de roteamento baseado nos algoritmos de EG, este procedimento é o mesmo independente da métrica utilizada: Jornada que Chega Mais Cedo ou Jornada Mais Curta.

Um dos principais objetivos deste trabalho é investigar como os algoritmos baseados em EG se comportam quando utilizados em protocolos de roteamento. Entretanto, é importante mencionar que os protocolos implementados como contribuição deste trabalho não contemplam uma operação distribuída, i.e. com mensagens de controle, mecanismos distribuídos para disseminação do EG, controle de falhas, etc., característica desejada por diversas aplicações práticas. Com isso assumimos que o protocolo de roteamento baseado em EG recebe como entrada um *grafo evolutivo* contendo todo o comportamento da rede no tempo, cada nó sabe o exato momento em que um vizinho estará disponível para comunicação ou quando deixará de estar.

Seja Aresta-Horários um conjunto de intervalos de tempo representando o horário de existência de um canal de comunicação entre dois nós. Uma aresta existe quando dois nós estão ao alcance um do outro, isto é, um enlace de comunicação pode ser estabelecido pois os nós estão fisicamente próximos e ativos. O grafo evolutivo (EG) de uma rede dinâmica pode ser representado por uma lista de Aresta-Horários para cada par de nós, em outras palavras, cada nó tem uma lista de todos os vizinhos alcançáveis em um determinado instante.

Quando um pacote enviado por s chega à camada de roteamento do nó u no tempo t_{now}, este calcula todo o trajeto a ser percorrido pelo pacote desde a origem (s) a todos os outros nós da rede, utilizando um dos algoritmos: Jornada que Chega Mais Cedo ou Jornada Mais Curta (como mostrado nas seções anteriores).

Suponha que o próximo salto da jornada seja o nó v no instante t_v. Se $t_v = t_{now}$, então os dois nós já estão ao alcance um do outro (i.e. existe uma aresta (u, v) no tempo t_{now} em Aresta-Horários), e o pacote é enviado a v. Senão, caso $t_v > t_{now}$, não existe um enlace entre os dois nós neste instante, então o nó u irá armazenar e reagendar a transmissão do pacote. Sendo o instante t_v o primeiro momento de existência da aresta

(u, v) em Aresta-Horários com $t > t_{now}$.

Tabela 3.1.: Aresta-Horários para o EG da figura 3.2.2

Aresta	Horários
A – B	1, 2 , 3
A – C	4
A – D	4
B – C	1, 3
B – E	2
C – D	1
C – E	1, 3, 4
E – F	1, 4
E – G	3
F – G	2

A tabela 3.1 mostra os horários de existência das arestas do exemplo da figura 3.2.2 (página 27). Note que os horários de existência são os intervalos mostrados nas arestas do grafo, e compreendem os instantes em que a comunicação pode ser estabelecida entre os nós. Como mencionado anteriormente, neste exemplo não estamos levando em conta o tempo gasto na transmissão, portanto, um pacote enviado em um determinado instante pode chegar ao destino no mesmo exato instante, caso o trajeto exista. Na implementação dos protocolos (Seção 4.4, página 45), cada intervalo em Aresta-Horários possui um horário exato de início e término, e o tempo estimado da transmissão (tempo de percurso) é levado em conta.

Como exemplo, a Jornada que Chega Mais Cedo para uma mensagem enviada no instante 1 de **D** a **G** será **{D,C,E,F,G}**. O pacote irá chegar em **F** no mesmo momento 1, este irá agendar o envio do pacote para **G** no próximo momento em Aresta-Horários(F,G), que será o instante 2. Se existe mais de uma rota possível naquele momento, aquela com menor número de saltos (mais curta) será escolhida, se ambas tem o mesmo número de saltos, a rota do nó com o menor identificador[1] será escolhida. Dessa maneira garante-se uma ordem total na escolha das *jornadas* (em outras palavras: estamos definindo como os nós serão ordenados no *heap* utilizado pelo algoritmo).

Por simplicidade na nossa implementação, cada nó do percurso irá recalcular *todo* o trajeto da origem ao destino, para depois escolher para qual vizinho (e quando) irá retrans-

[1]Identificador é um número único atribuído a cada nó na construção do grafo evolutivo.

mitir os dados. Feito desta maneira, o protocolo implementado possui uma sobrecarga de processamento, pois todos os nós do percurso irão recalcular todo o trajeto do pacote.

É possível, no entanto, com poucas modificações alterar o protocolo para que somente o nó que originou o pacote precise calcular a rota de envio, feito isto ele poderia utilizar a técnica de *Source-Routing* e adicionar no cabeçalho do pacote o trajeto completo a ser percorrido. Cabendo então aos nós no meio do percurso apenas retransmitir o pacote para o próximo da lista no momento apropriado.

3.6. Trabalhos Relacionados

Outros trabalhos teóricos relacionados a redes dinâmicas (variações na topologia) podem ser encontrados em [33, 50], sendo que estes estão, em sua maioria, relacionados ao fluxo em redes quando as suas arestas tem custos que variam no tempo. Kotnyek faz em [48] uma revisão geral sobre este tema.

O conceito de grafos expandidos no tempo (*time expanded graph*) definido por Ford e Fulkerson em [33] cria um grafo estático equivalente a partir de um grafo dinâmico (com arestas variando no tempo). Para isso é necessária uma discretização do tempo, e para cada instante t, uma nova *camada* contendo os mesmos vértices do original é adicionado ao grafo, mas as arestas são atualizadas para aquele instante. Feito isso, os algoritmos combinatórios para redes estáticas podem ser aplicados diretamente no grafo expandido. Se as variações nas arestas no tempo não forem discretas, a solução do grafo expandido pode até ser utilizada, mas a complexidade para o cálculo dos caminhos mínimos poderá se tornar inviável. Note portanto que esta solução é viável apenas quando estamos utilizando tempos discretos.

Infelizmente, a abordagem de [33] não se aplica à FSDN, pois o fator de expansão será enorme devido à quantidade de camadas que serão necessárias para representar um grafo com variações não-discretas no tempo. O conceito de grafos evolutivos foi desenvolvido com o objetivo de representar redes dinâmicas de forma compacta, com ênfase em redes FSDNs.

Köhler et al., em [50], estudaram grafos expandidos em que o custo das arestas (e.g. tempo de percurso) variam com o tempo e a carga do sistema.

Outro projeto que está relacionado à teoria de grafos evolutivos é o arcabouço MERIT [27], que usa o conceito de análise competitiva [19] em um ambiente dinâmico para qualificar o desempenho de protocolos de roteamento através do histórico da topologia da rede.

Os trabalhos realizados até este momento na teoria de grafos evolutivos está restrito à formalização de uma FSDN e a exploração do modelo com o intuito de diminuir a complexidade de algoritmos conhecidos e criar novos modelos de referência. O conceito de EG e algumas métricas de maximização já foram detalhados em alguns artigos, mas, até o presente momento em nenhum destes trabalhos foram realizados testes experimentais ou simulações realistas [2].

Resumo

Neste capítulo, foram mostrados os conceitos básicos de grafos evolutivos no que tange algoritmos de roteamento ponto a ponto. Dois algoritmos de custo mínimo foram definidos e uma discussão sobre a implementação de protocolos de roteamento utilizando estes algoritmos foi realizada. Na próxima seção serão apresentados os testes experimentais realizados.

[2] Com exceção dos artigos publicados durante este trabalho [58–60]

4. Experimentos

Neste capítulo apresentamos o ambiente de simulação escolhido para realizar a implementação e testes dos protocolos de roteamento baseados em grafos evolutivos. Descrevemos aqui com detalhes as características das simulações realizadas e os cenários utilizados para testes.

O código-fonte da implementação está publicamente disponível na Internet, no endereço: `http://julian.com.br/mobidyn/software`, sob a licença de software livre GPL [38].

4.1. Simulador de Redes NS2

A parte experimental foi realizada com o auxílio do simulador de redes NS2 (*Network Simulator 2*) [61] desenvolvido em Berkeley. O NS2 é um simulador baseado em eventos discretos, e foi desenvolvido para auxiliar a pesquisa na área de redes de comunicação. É um projeto muito abrangente e genérico, que permite a simulação de várias camadas de rede, incluindo protocolos de roteamento, transporte (TCP, UDP, ICMP, etc.), aplicação e tráfego (FTP, HTTP, Telnet, CBR, etc.).

Esta ferramenta vem sendo aprimorada desde 1989, e em 1995 teve o apoio da DARPA através do projeto VINT (Virtual InterNetwork Testbed) [76] e hoje está sob a licença GPL. O seu desenvolvimento continua pleno e é realizado por diversos pesquisadores ao redor do mundo. A próxima versão NS3 já está em desenvolvimento, contando com o apoio de diversas instituições financiadoras [62].

Inicialmente o projeto do NS2 não contemplava a simulação de redes sem fio ou redes móveis. Esta extensão foi desenvolvida pela CMU (Carnegie Mellon University) no projeto Monarch [69] e agregou as características do padrão IEEE 802.11, protocolos de acesso ao meio (MAC) [37] e modelos de transmissão via rádio.

4.2. Arquitetura e Funcionamento

O simulador é implementado utilizando duas linguagens: C++ e OTcl [1]. A parte de simulação de eventos e protocolos, que precisa ser eficiente, foi desenvolvida em C++, já a parte "configurável", ou parte "do usuário", é feita em OTcl, que por ser interpretada deixa o processo de criação e execução de testes mais prático e ágil.

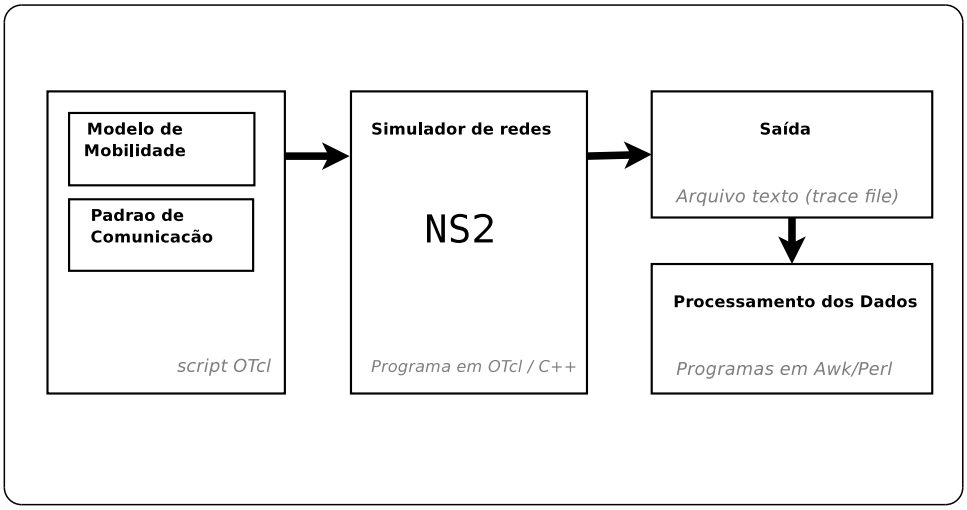


Figura 4.2.1.: Fluxo de uma simulação no NS2

Uma simulação no NS2 segue o fluxo representado na figura 4.2.1, sendo que o usuário precisa criar um *script* na linguagem OTcl definindo as características da simulação, que são:

1. **Características da Rede**: Tipo de rede (com fio, sem fio), número de nós, protocolo de roteamento, arquivo com modelo de mobilidade, arquivo com padrão de comunicação, alcance da antena, modelo de energia, modelo de falhas, etc. (veja exemplo no Apêndice A);

2. **Modelo de Mobilidade**: Seqüência de movimentos bem definidos para os nós da rede (algo como: no momento t o nó n deve se deslocar para a posição x, y com velocidade v), esses movimentos geralmente são gerados por um programa auxiliar, que recebe as características da mobilidade e gera uma seqüência de comandos OTcl representando aquele modelo (veja exemplo no Apêndice A);

[1] Object Tool Command Language

3. **Padrão de Comunicação**: Seqüência de comandos que definem os enlaces de comunicação entre os nós da rede, definem o fluxo e o tipo da comunicação que será realizada (algo como: a partir no instante t, o nó s irá transmitir dados para r, utilizando o protocolo FTP com uma taxa de transmissão de $10kbps$ – veja exemplo no Apêndice A).

O resultado de uma simulação no NS2 é um arquivo em formato texto com os rastros da execução (*trace file*), este arquivo contém detalhes de toda a comunicação realizada entre os nós da rede, desde as camadas mais altas (aplicação) até as trocas de mensagens para acesso ao meio (MAC). Este arquivo, que potencialmente tem um tamanho grande (centenas ou milhares de megabytes), deve ser posteriormente analisado através de programas auxiliares (geralmente *scripts* Awk ou Perl) para obtenção e agregação dos dados a serem analisados: quantidade de pacotes perdidos, atraso no recebimento, vazão, etc.

É importante notar que uma simulação no NS2 pode ser totalmente reprodutível, isto é, o simulador apenas interpreta os arquivos de entrada (*scripts*) que contêm toda a seqüência bem definida da simulação e comunicação entre os nós. Para se simular, por exemplo, diferentes padrões de mobilidade, é necessário criar diversos *scripts*, sendo que cada um deles contém a instância de uma simulação.

Para se analisar o comportamento de diversos protocolos de roteamento, deve se rodar o simulador diversas vezes, uma para cada protocolo, sendo que a descrição da movimentação dos nós e fluxos de transmissão serão exatamente os mesmos.

Devido ao grande número de simulações necessárias a este trabalho, também foram implementados *scripts* em Bash para o lançamento de simulações em lote e *scripts* em Awk e Perl para posterior análise dos traços. Os gráficos foram plotados utilizando o aplicativo `gnuplot`.

4.3. Ambiente de Simulação

Para analisar o comportamento dos recentes protocolos baseados em EG estamos utilizando a versão 2.29 do NS2, com extensões para redes móveis desenvolvidas pelo grupo CMU Monarch e características de uma rede IEEE 802.11 para transmissão de rádio, camada física e controle de acesso ao meio (MAC). O modelo de comunicação utiliza características similares ao produto comercial *Lucent WaveLAN*, modelado como meio compartilhado (*shared-media*), com uma banda nominal de 2 MB/segundo e raio de pro-

pagação nominal médio de 250m. O esquema de redução de colisões RTS/CTS (*Request to Send/Clear to Send*) está ativo em todas as simulações.

50 nós foram espalhados aleatoriamente em um campo com dimensões de **1500m x 500m**. A idéia de definir um campo retangular foi com o intuito de aumentar o diâmetro da rede (i.e. número médio de saltos necessários no roteamento de informações de uma ponta a outra). O tamanho da fila na interface de rede (IFQ – *Interface Queue*) é de 50 pacotes. A duração de cada simulação é de **900 segundos**.

As medições de energia estão sendo feitas utilizando o modelo simples **EnergyModel** do NS2, com os seguintes valores de consumo: $75mW$ para transmissão e recepção, e $1mW$ no estado dormente. A energia inicial de cada nó é de 100 Joules, o suficiente para que todos os nós estejam com carga até o final da simulação. Não chegamos a testar a extensão para o modelo de energia proposto em [47], o qual inclui maior controle no estado (ligado, desligado, alerta, etc.) de um nó.

Não utilizamos TCP para comunicação de dados, pois não estamos interessados em analisar o protocolo TCP e suas características como: controle de fluxo, retransmissão, etc., mas queremos porém uma visão geral de como os protocolos se comportam utilizando um fluxo comunicação de dados simples, como o UDP.

Nestes experimentos nos decidimos desabilitar o uso do Protocolo para Descoberta de Endereços (ARP – *Address Resolution Protocol*) dos nós móveis. No NS2, a resolução de nomes é uma aproximação do ARP no BSD Unix, desta forma a resolução é realizada sob demanda assim que os pacotes são recebidos da camada da aplicação, e o tamanho da fila para pacotes que tem o mesmo destino é de apenas um pacote. Com isso, durante o processo de resolução de endereços, todos os pacotes subseqüentes endereçados para o mesmo destino são descartados. Esse comportamento leva ao descarte de uma grande quantidade de pacotes quando os dados são enviados em forma de rajada para um destino que ainda não esteja na tabela ARP. Concordando com a argumentação de Carter, Yi e Kravets em [15], na qual as MANETs devem ter os seus próprios meios de descoberta de vizinhos, com intuito de dimuir colisões e possíveis perdas, principalmente em protocolos do tipo *manycast*. Logo, o ARP nas simulações deste trabalho será feito de forma automática (i.e. todos os nós "conhecem" o endereço de seus vizinhos).

Para simular a transmissão de dados foram criados **10 fluxos** de comunicação do tipo UDP entre pares de nós, com taxa de transmissão constante (CBR – *Constant Bit Rate*) de **2 pacotes/segundo** e tamanho de **256 bytes** cada pacote. Tem-se que 20% dos nós estão transmitindo dados, sendo que os nós de número 1 a 10 são os transmissores e os nós de 40 a

49 são os receptores. Cada fluxo inícia a transmissão de pacotes em um momento aleatório entre 0 e 10 segundos de simulação. Estes valores são similares aos usados em artigos conhecidos de análise de desempenho neste tipo de rede: [9, 10, 20, 22, 43, 67]. No entanto, a baixa taxa de transmissão foi escolhida com o intuito de simular o comportamento de redes de sensores coletando dados.

Não esperamos que este modelo seja uma precisa reprodução de uma FSDN, LEO ou RSSF real, mas desejamos neste primeiro passo comparar os protocolos de roteamento baseados em EG ($EG_{foremost}$ e $EG_{shortest}$) em relação aos protocolos bem difundidos para redes móveis em geral (AODV, DSR, OLSR e DSDV).

4.3.1. Modelos de Mobilidade

O Modelo de Mobilidade das simulações foi dividido em dois tipos de cenários distintos: o simples e bem conhecido *Random Waypoint* (RWP) apresentado por Johnson e Maltz [44], e o *Modelo Intermitente*, definido em [58, 60] e mais apropriado a redes FSDN. Discussões sobre modelos de mobilidade podem ser encontradas em [7, 8, 14, 42, 79].

Modelo de Mobilidade Random Waypoint

No cenário de mobilidade *Random Waypoint* (RWP) os nós se movem para destinos escolhidos aleatoriamente, com velocidade constante entre 1 e 20 m/s escolhidas com uma distribuição normal. Entre as movimentações, os nós permanecem parados por um determinado tempo, denominado aqui PauseTime. As simulações neste cenário foram realizadas utilizando valores de PauseTime variando de 0 (sem pausa, alta mobilidade) a 900 segundos (longas pausas, baixa mobilidade). Para evitar problemas discutidos por Yoon et al. em [79] sobre o modelo RWP, estão sendo utilizados somente valores de velocidade não nulos. O uso deste cenário, mesmo com as suas conhecidas limitações, é importante para que consigamos comparar os resultados com outros estudos.

A geração dos *scripts* com os modelos de mobilidade foram realizadas usando o programa `setdest` desenvolvido pela CMU e disponibilizado juntamente com o pacote do NS2. Também foi testada a geração de modelos de mobilidade usando a ferramenta `BonnMotion` [23], a qual está num estágio maduro e contém diversos modelos de mobilidade disponíveis.

Modelo de Mobilidade Intermitente

O *Modelo Intermitente*, definido aqui neste trabalho, baseia-se em nós com posição fixa e que ininterruptamente se ligam e desligam (acordam e dormem) por certos períodos, veja figura 4.3.1. Aqui, os parâmetros que são alterado nas diferentes simulações são: SleepProb (variando de 0 a 50%), e HoldTime (variando entre 15, 60 e 180 seg.).

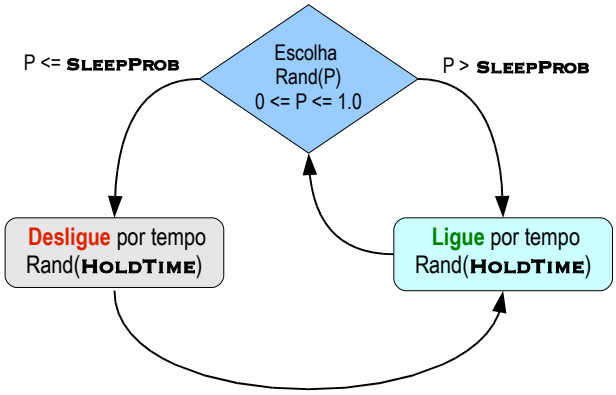


Figura 4.3.1.: Sequência de estados de um nó no Modelo de Mobilidade Intermitente

No início da simulação cada nó está ligado (acordado), e por toda a período da simulação ele terá a probabilidade SleepProb de desligar (ir dormir). Se um nó vai dormir, ele permanece desligado por um tempo uniformemente aleatório denominado HoldTime (retângulo da esquerda na figura 4.3.1). Ao término deste período, o nó obrigatoriamente acorda e permanece acordado por mais um período aleatório delimitado por HoldTime (retângulo da direita na figura). Após esse tempo acordado, o processo é reiniciado (losango central na figura). Caso o nó não vá dormir (probabilidade de $1 - \text{SleepProb}$), então ele ficará acordado por mais um tempo HoldTime e após o processo é reiniciado. É importante perceber que HoldTime é recalculado a cada troca de estado.

Este modelo foi chamado de *Intermitente* e tem a intenção de reproduzir, mesmo que de forma simplista, o modelo de redes de sensores, no qual os nós podem ficar indisponíveis por determinado tempo (por economia de energia, horários de funcionamento, etc.) para depois voltar à operação normal. Este modelo também representa de forma mais clara uma rede com comportamento previsível (FSDN). Alguns modelos parecidos podem ser encontrados em [49] (Capítulo 7).

4.3.2. Cenários de Teste

Em cada modelo, para cada combinação de parâmetros foram realizadas 10 simulações com diferentes sementes aleatórias. Assim, para o primeiro modelo, RWP, foram executadas 110 simulações, i.e. 10 vezes para cada valor de PauseTime: $0, 50, 100, 200, ..., 900$ segundos. Para o segundo modelo, *Intermitente*, foram realizadas 300 simulações, sendo 10 vezes para cada combinação de SleepProb: $0, 5, 10, ..., 50\%$ e HoldTime: $15, 60$ e 180 segundos. Note que no segundo modelo de mobilidade estamos variando 2 parâmetros diferentes nas simulações, e que o valor de 0% de SleepProb significa que nenhum nó vai dormir, portanto a topologia de rede permanece estática do início ao fim da simulação, sendo estes resultados importantes como referência de desempenho.

Todos os 6 protocolos de roteamento (DSDV, DSR, AODV, OLSR e $\text{EG}_{foremost}$, $\text{EG}_{shortest}$) foram executados nos mesmos 410 cenários. Os mesmos exatos cenários de mobilidade e fluxos de comunicação foram utilizados nas simulações dos diferentes protocolos.

4.4. Detalhes de Implementação

A implementação utilizada para simular os protocolos de roteamento, DSDV, AODV e DSR foi a padrão fornecida no pacote do NS2. Todos eles foram implementados pelo grupo CMU Monarch [69]. A implementação do protocolo AODV, na versão 2.29 do NS2, já inclui algumas otimizações implementadas por *Marina e Das* [55, 67]. Em relação ao OLSR, foi utilizada a implementação de Francisco J. Ros [70] (UM-OLSR) versão 0.8.8. Em todas elas, as constantes e os parâmetros de configuração foram deixados com valores padrão.

Protocolos $\text{EG}_{foremost}$ e $\text{EG}_{shortest}$

Como já mencionado, um cenário de mobilidade no NS2 é geralmente representado por uma seqüência de comandos OTcl e armazenado em um arquivo separado que é usado na configuração de uma simulação. O protocolo de roteamento definido na Seção 3.5 recebe como entrada um grafo evolutivo (representação das modificações da topologia da rede no tempo). Para isso, foi criado um programa – **calcEG** – que recebe como entrada esta seqüência de comandos de um cenário de mobilidade do NS2 e, através de uma simulação prévia, gera um *grafo evolutivo* correspondente àquele cenário.

Antes da simulação começar, este *grafo evolutivo* é distribuído entre os nós da rede. Com isso, cada nó sabe de antemão todo o comportamento da rede durante o período da simulação, i.e. cada nó sabe o exato momento em que um vizinho estará disponível para comunicação ou quando deixará de estar.

Em um primeiro momento, esta importante suposição não parece ser realista, no entanto, existem diversas situações [1, 17, 25, 30, 56, 78, 81] em que o EG pode ser construído antes da fase de roteamento. Note que somente a topologia da rede no tempo será disseminada pelos nós, sendo que os fluxos de dados que irão trafegar não são conhecidos.

Difculdades Encontradas

No caso do cenário Intermitente, a mudança de estado de um nó de "ligado" e "desligado" é feita atráves dos comandos *on* e *off* já implementados no `MobileAgent` do NS2. Mas, na versão 2.29 estes comandos não funciavam a contento, foi necessário alterar o código e remover a linha que executa um "`reset-state`" no comando *off* do arquivo `mobilenode.cc` (este comando só era utilizado pelo `DiffusionAgent`).

Ao simular o protocolo DSDV, também no cenário Intermitente, o protocolo por muitas vezes entrava em um laço infinito (ao fazer a manutenção de algumas filas internas). Esse possível comportamento, que é conhecido pela comunidade do NS2, pode ser contornado com o esvaziamento de uma fila utilizada para o reenvio de pacotes, linha 794 do arquivo `dsdv.cc`.

4.5. Detalhes de Execução

O pacote com o simulador NS2 pode ser baixado diretamente da Internet, sendo que a distribuição completa *ns-allinone*, na versão 2.29, ocupa aproximadamente 200 MB quando descompactada no disco. Esse tamanho é proveniente dos diversos anos de desenvolvimento e a agregação de inúmeras funcionalidades.

Porém, para se executar diversos experimentos em lote não há uma ferramenta padrão, tão pouco encontramos alguma que pudesse nos ajudar. Para isso, foi necessário construir um arcabouço com diversos *scripts* nas linguagens *Bash* e *Perl* para automatizar a execução dos testes programados para este trabalho. Estes *scripts* criados para automatizar a realização dos diversos experimentos são configuráveis e incluem as seguintes fases de execução:

1. **scen:** geração dos cenários de mobilidade (RWP ou Intermitente);
2. **eg:** criação do grafo evolutivo para cada cenário;
3. **sim:** executar um lote de simulações;
4. **stats:** coletar estatísticas, agregar os dados e consolidar resultados;
5. **graphs:** gerar diversos gráficos;
6. **clean:** compactar ou remover arquivos de traços.

Tempo de execução

Os experimentos foram realizados em computadores da plataforma Intel Xeon™ e AMD Atlhon™ 64, com processadores em torno de 3GHz. Utilizamos o sistema operacional Linux. O tempo de execução dos diferentes protocolos variam bastante, mas na maioria dos casos, cada simulação de 900 segundos, demorou entre 3 minutos e 12 minutos para ser finalizada.

Ao todo, para executar a bateria de pelo menos 410*6*2 = 4920 simulações (410 cenários, 6 protocolos, e 2 tamanhos de IFQ) prevista neste trabalho, os computadores foram utilizados a pleno por vários dias. Foi necessário buscar algumas alternativas para tentar melhorar o desempenho do simulador NS2, como por exemplo, trocar o escalonador de eventos original por uma versão mais rápida.

Espaço em disco

Os arquivo contendo os traços de simulação ocupam grande espaço no disco. As nossas simulações foram realizadas utilizando o novo esquema de *log* do NS2, denominado *wireless new trace file* (opção: *use-newtrace*), com isso, cada traço da simulação ocupou em torno de 400 a 600 MB, que após compactação é reduzido a 30 MB cada. Estes valores nos impossibilitaram de armazenar os traços de todas as simulações. Portanto, após o término de cada simulação, as estatísticas eram coletadas e o arquivo com os traços compactados ou apagados.

4.6. Estimando o Tempo de Percurso de um Pacote

Ao simular protocolos de roteamento baseados em EG, uma variável importante a ser configurada no algoritmo é o tempo de percurso de um pacote (ou em outras palavras: o custo de uma aresta no grafo). Na teoria de EG é possível que cada aresta do grafo possua um custo diferente, mas, no entanto, nas simulações realizadas neste trabalho foi utilizado um tempo homogêneo para todos os enlaces. Este parâmetro é definido na configuração dos protocolos $\mathrm{EG}_{shortest}$ e $\mathrm{EG}_{foremost}$ no NS2.

Realizamos diversos experimentos para estimar um valor médio de percurso razoável para ser utilizado no simulador de redes. Para isso, foi criado um cenário "estressante", em que os nós estão relativamente próximos entre si, gerando colisões no acesso ao meio.

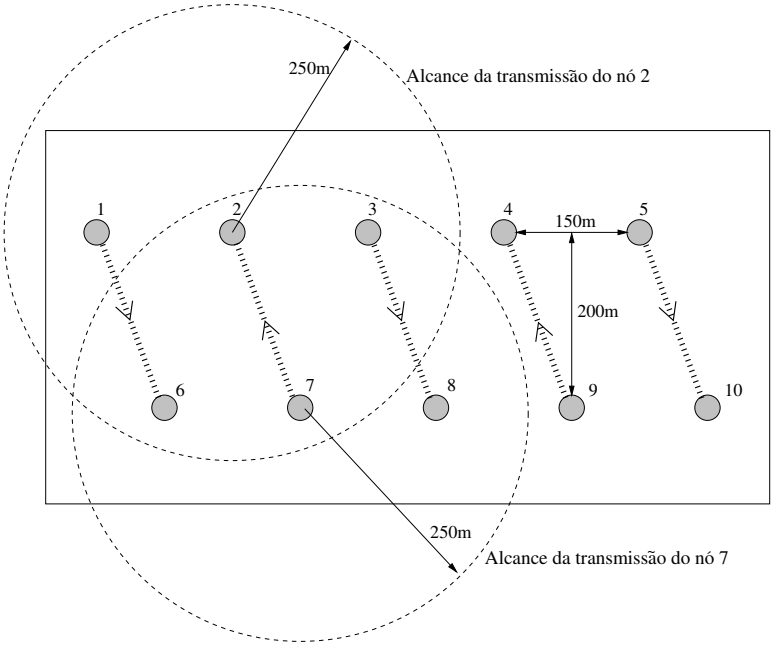


Figura 4.6.1.: Cenário estático para estimar o tempo de percurso de um pacote entre pares de nós.

Foram utilizados 10 nós espalhados em um cenário fixo como mostrado na figura 4.6.1, dentre estes, 5 nós estão ininterruptamente transmitindo pacotes de dados a taxas constantes utilizando o protocolo UDP. Variamos a taxa de transmissão de 1 a 32 pacotes/segundo e o tamanho dos pacotes de 64 a 1000 bytes. Como mostra a figura, os nós impares são os transmissores e os nós pares os receptores

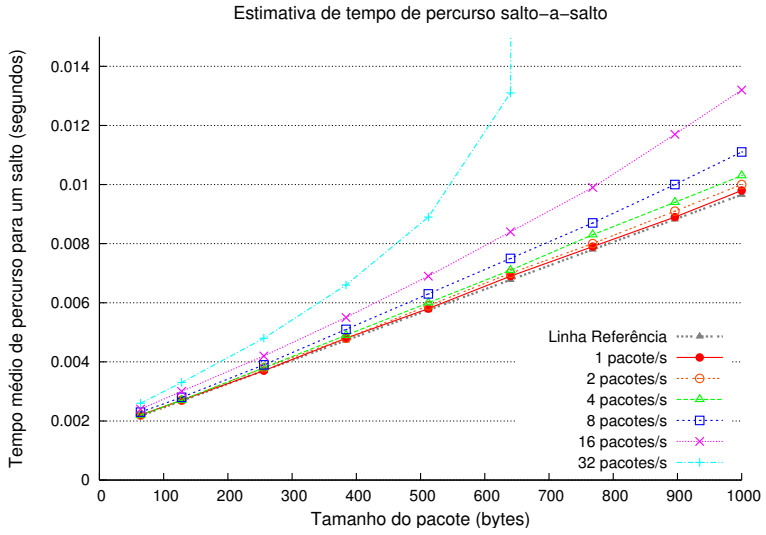


Figura 4.6.2.: Tempo médio para diferentes taxas de transmissão e diferentes tamanhos de pacotes.

As curvas na figura 4.6.2 mostram que o aumento no tempo de percurso entre pares de nós é linearmente dependente com o tamanho do pacote. Ainda na figura 4.6.2 podemos observar que para taxas de transmissão de 32 pacotes/segundo, e pacotes a partir de 650 bytes, a rede colapsa, e o tempo para envio/recebimento de um pacote aumenta drasticamente, fato que é seguido (mas não ilustrado no gráfico) por uma grande quantidade de perda de pacotes por colisão no meio físico.

Como resultado deste experimento, para transmissão de um pacote de **256 bytes** a uma taxa de **2 pacotes/segundo**, chegamos ao valor médio de **3,5 ms**, o qual, portanto, será utilizado em todas os experimentos do próximo capítulo.

Na linha mais inferior do gráfico tem-se a curva de referência, equivalente à simulação de apenas 2 nós, sendo um transmissor e o outro receptor, neste caso não há colisões para acesso ao meio.

Resumo

Neste capítulo, foi apresentado o ambiente de simulação para a realização dos experimentos com os protocolos de roteamento, bem como os detalhes de configuração e dois cenários de mobilidade que serão utilizados. Também fizemos uma breve discussão sobre o

tempo de percurso nó-a-nó, que é um parâmetro importante para os algoritmos de grafos evolutivos. No próximo capítulo serão mostrados os resultados das simulações nos cenários apresentados aqui.

5. Resultados

Extensivas simulações foram realizadas no simulador de redes NS2. Neste capítulo serão apresentados os resultados obtidos, bem como a análise de algumas propostas para aprimorar a utilização de EG em protocolos de roteamento. Cada simulação executada gera um arquivo para posterior análise, contendo toda a comunicação realizada entre os nós, incluindo a camada MAC. Este arquivo é então analisado e os resultados são consolidados para posterior análise.

Como descrito no capítulo anterior, os resultados mostrados aqui estão divididos em duas partes, a primeira utilizando os cenários com o modelo de mobilidade RWP e a segunda utilizando o recente Modelo Intermitente.

5.1. Métricas de Desempenho

As seguintes métricas quantitativas e qualitativas serão analisadas:

- **Vazão média**: número médio de pacotes que chegam ao destino por segundo;
- **Razão de pacotes perdidos**: razão de pacotes perdidos em relação ao número total de pacotes enviados. Esta métrica foi subdividida em duas partes:
  - **Razão de pacotes perdidos por falta de rota (NRTE)**: razão de pacotes perdidos por *não existir uma rota* em relação ao número total de pacotes enviados;
  - **Razão de pacotes perdidos por fila cheia no enlace (IFQ)**: Razão de pacotes perdidos por IFQ em relação ao número total de pacotes enviados. Esta é a fila na camada de enlace, sendo que os pacotes são enviados da camada de roteamento para a fila IFQ;

- **Número médio de saltos**: número médio de saltos percorridos da origem ao destino para os pacotes entregues com sucesso;

- **Atraso médio fim-a-fim**: o tempo médio de percurso do pacote desde a criação até a entrega no destino;

- **Consumo médio de energia**: gasto médio de energia nas fases de transmissão, recepção e sem atividade (*idle*) por pacote entregue com sucesso.

Devido à grande quantidade de resultados obtidos, não iremos mostrar os dados de todos os protocolos em todas as métricas, mostraremos apenas aqueles que contém algum resultado relevante e/ou que exponha alguma característica peculiar de algum algoritmo. No apêndice A apresentamos um exemplo de traços de uma simulação no NS2 e os campos referentes a cada métrica.

Os protocolos baseados em EG utilizados nestes primeiros experimentos não possuem nenhum sobrecusto relativo a pacotes de controle, pois da maneira como foram implementados, cada nó da rede contém uma cópia do EG, não sendo necessária nenhuma comunicação extra entre os nós para sua disseminação. Portanto, não estamos interessados em analisar o sobrecusto dos protocolos.

Em praticamente todos os gráficos que serão apresentados a seguir, os protocolos aparecerão na seguinte ordem: $\text{EG}_{foremost}$, $\text{EG}_{shortest}$, OLSR, AODV, DSR e DSDV, e serão representados com os seguintes símbolos e cores (quando aplicável), respectivamente, círculo cheio (vermelho), circulo vazado (laranja), triângulo (verde), quadrado (azul), sinal de cruz (rosa) e sinal de mais (ciano). As barras de erro (superior e inferior) nas curvas dos gráficos representam um intervalo de confiança de 95%.

5.2. Resultados - Random Waypoint

Como mencionado anteriormente, nos cenários de mobilidade RWP, o parâmetro que será alterado nas sucessivas simulações é o PauseTime. Sendo que valores baixos de PauseTime significam *alta mobilidade*, e em contrapartida, valores altos de PauseTime significam *pouca mobilidade*, pois os nós permanecem muito tempo parados.

5.2.1. Vazão Média nos Cenários RWP

Como pode-se ver na figura 5.2.1, o protocolo $\text{EG}_{foremost}$ tem melhor desempenho na métrica de **vazão** para todos os valores de PauseTime. O limite teórico para vazão média é de $2 * 256 * 8 = 4096$ bits/segundo, e o $\text{EG}_{foremost}$ alcançou valores bem próximos.

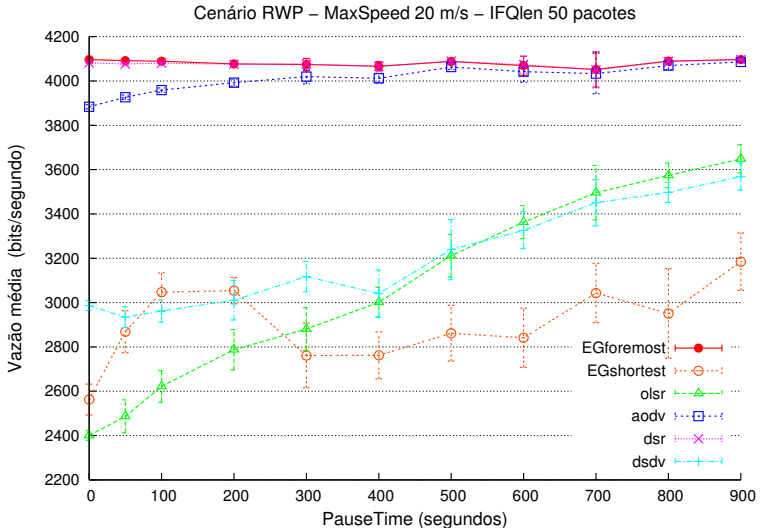


Figura 5.2.1.: Vazão média em função do PauseTime (mobilidade) no modelo RWP

Como suspeitado, o comportamento dos protocolos baseados em tabelas, OLSR e DSDV, não obtiveram um bom desempenho em cenários de maior mobilidade, dado que a manutenção das tabelas não é feita na velocidade necessária, segundo a mobilidade da rede. Com a redução da mobilidade – lado direito do gráfico, o desempenho destes protocolos melhora.

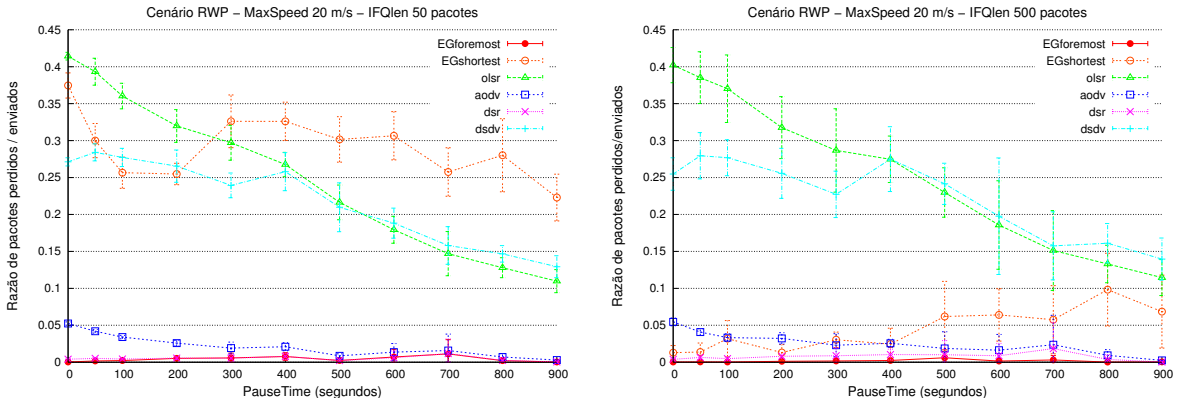
Ainda em relação à figura. 5.2.1, note que surpreendentemente o $\text{EG}_{shortest}$ não obteve valores altos de vazão, se opondo ao bons resultados do $\text{EG}_{foremost}$. Esse comportamento será discutido de forma mais detalhada na próxima seção [1].

5.2.2. Perda de Pacotes nos Cenários RWP

Na figura 5.2.2(a), o número de **pacotes perdidos** pelo $\text{EG}_{foremost}$ foi praticamente zero para todos os valores de PauseTime, isto é, menos de 0,2% de perda de pacotes.

[1] Devido a forma de apresentação incremental, este capítulo deve ser lido de preferência seqüencialmente.

A razão de pacotes perdidos no DSR foi muito boa também, com uma média de 0,4% de perdas. Foi uma surpresa o fato do AODV não apresentar desempenho tão bom em valores de alta mobilidade, se opondo aos valores mostrados por Perkins, Royer, Das, and Marina em [67]. Nos atribuímos este comportamento às características dos nossos testes, que possuem cargas relativamente baixas de comunicação (10 fluxos de tráfego com 2 pacotes/segundo cada), ou talvez se deva por algum ajuste por nós desconhecido nos parâmetros do algoritmo.



(a) IFQ de 50 pacotes

(b) IFQ de 500 pacotes

Figura 5.2.2.: Número de pacotes perdidos em função do PauseTime (mobilidade) para diferentes tamanhos de fila IFQ no modelo RWP. Observe a melhora no desempenho do $EG_{shortest}$ com o aumento da fila.

Na figura 5.2.3, tem-se que o número de pacotes perdidos por transbordo da fila no enlace, IFQ, é alto (em torno de 29%) somente no protocolo $EG_{shortest}$. Experimentamos aumentar o tamanho da fila de 50 para 500 pacotes, e foi observada uma boa melhora no desempenho deste protocolo, veja figura 5.2.2(b).

Portanto, nos próximos resultados do cenário RWP, o tamanho da fila IFQ foi aumentado para 500 pacotes, com o intuito de evitar interpretações erradas dos resultados devido ao grande número de pacotes perdidos pelo protocolo $EG_{shortest}$.

Notamos que essa perda acentuada no protocolo de Jornada Mais Curta ocorre em razão da existência de gargalos [58–60], e deixamos para o final deste capítulo uma discussão mais detalhada sobre esse problema, onde apresentaremos algumas soluções empíricas para diminuir estas perdas.

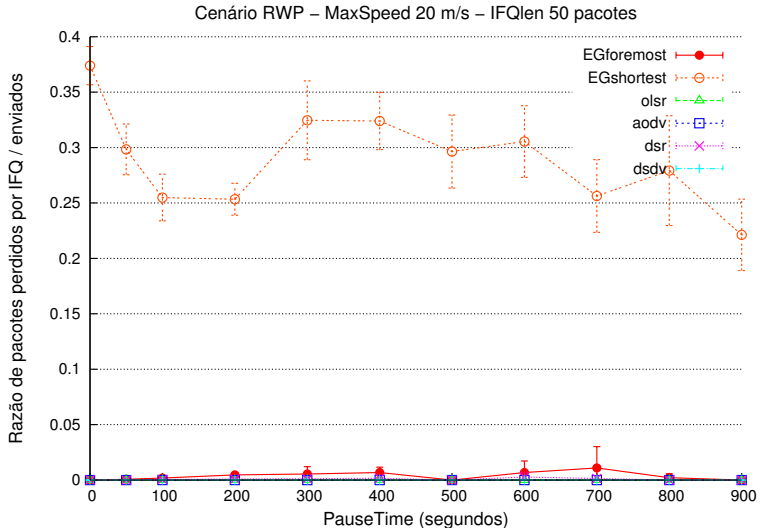


Figura 5.2.3.: Perda de Pacotes por fila IFQ cheia em função PauseTime (mobilidade) no Modelo RWP.

5.2.3. Número de Saltos nos Cenários RWP

O objetivo do algoritmo Jornada Mais Curta é calcular as *jornadas* que chegam ao destino percorrendo o **menor número de saltos** dentro de um certo tempo de simulação. Observando a figura 5.2.4, o protocolo $EG_{shortest}$ obteve os menores valores nesta métrica. Se considerarmos somente os pacotes entregues ao destino, observamos que o $EG_{shortest}$ pode ser considerado um bom candidato a limitante inferior nesta métrica em cenários deste tipo.

5.2.4. Consumo de Energia nos Cenários RWP

Uma outra característica relevante observada no $EG_{shortest}$ é o baixo **consumo de energia** dos nós da redes (veja figura 5.2.5), condizendo com os valores obtidos no número médio de saltos percorridos acima. Nos cenários com alta mobilidade, os valores do OLSR são $3,8$ vezes piores que o $EG_{shortest}$, e no caso de baixa mobilidade, o $EG_{shortest}$ tem um desempenho 40% melhor que o OLSR (que obteve os piores valores nesta métrica – e analisando os resultados, isso se deve à colisões e a um grande número de reenvios de pacotes). Note, novamente, que os protocolos baseados em EG não possuem o sobrecusto da descoberta e manutenção de rotas, portanto, devemos entender estes valores como sendo

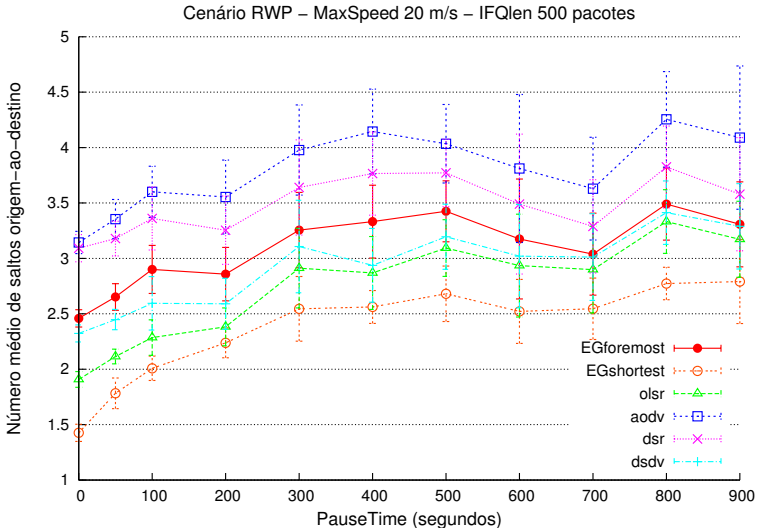


Figura 5.2.4.: Número médio de saltos percorridos para os pacote entregue ao destino no Modelo RWP.

o caso ideal de roteamento, em que toda a topologia é conhecida.

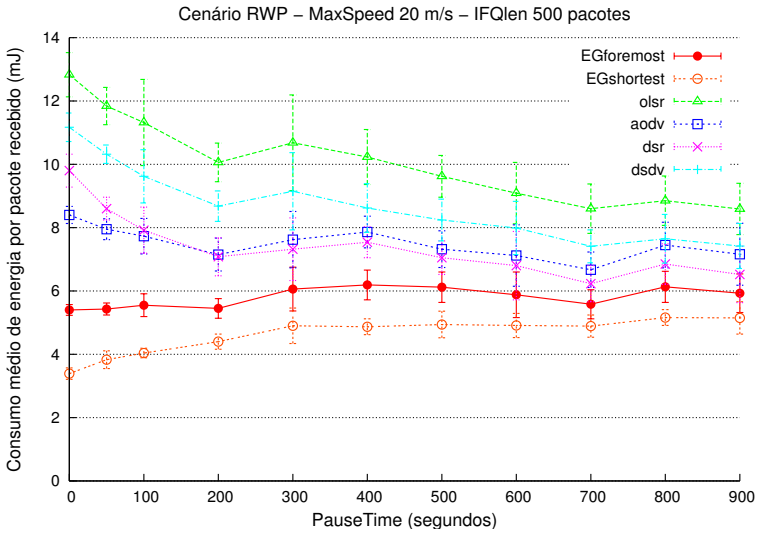


Figura 5.2.5.: Consumo médio de energia normalizado pelo número de pacotes entregues com sucesso no Modelo RWP.

Na figura 5.2.6 estamos mostrando o consumo de energia de cada nó ao final de uma

simulação RWP (cenário de número 1 com PauseTime 0). Nestes experimentos com o consumo de energia, não estamos analisando o tempo de vida da rede ou o mapa de energia [34], que são tópicos interessantes para futura pesquisa na área de grafos evolutivos. Note que o problema de maximar o tempo de vida da bateria dos nós da rede é NP-difícil [53].

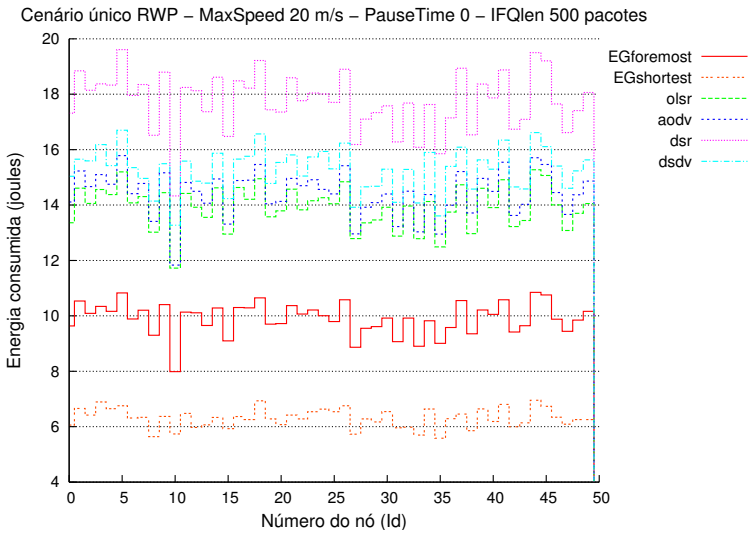


Figura 5.2.6.: Consumo de energia por nó para um cenário com PauseTime 0 no modelo RWP.

5.2.5. Atraso Médio Fim-a-Fim nos Cenários RWP

O objetivo da métrica Jornada que Chega Mais Cedo é calcular as *jornadas* que chegam ao destino no instante mais cedo possível. Mas, o comportamento intrínseco dos algoritmos baseado em EG de "segurar" pacotes por um longo período até que uma aresta exista, faz com o o atraso médio fim-a-fim obtido pelo $EG_{foremost}$ seja muito alto. Note, no entanto, que estes pacotes que permanecem em um nó aguardando por uma aresta ideal são, em muitos casos, jogados fora pelos outros protocolos. Portanto, se levarmos em conta todos os pacotes entregues no destino para calcular o atraso médio, o protocolo $EG_{foremost}$ está em desvantagem, pois no calculo da média os pacotes que permaneceram parados por períodos longos irão prejudicar o valor final da média. Lembre-se, que nos outros protocolos estes pacotes foram jogados fora por não haver rotas naquele instante

e portanto o atraso médio terá valores baixos. Em outras palavras, quando usamos os algoritmos baseados em EG, os pacotes podem demorar muito para chegar ao destino.

Contudo, na tentativa de mensurar o real desempenho do $EG_{foremost}$ em relação aos outros protocolos, calculamos o atraso médio fim-a-fim levando em conta somente os pacotes que são entregues com sucesso em todos os protocolos (i.e. **intersecção dos pacotes entregues**). Os resultados na figura 5.2.7 mostram que o $EG_{foremost}$ serve como um limitante inferior nesta métrica. Mesmo assim, é bom ressaltar que esta métrica não está considerando o fato que o $EG_{foremost}$ entrega praticamente todos os pacotes.

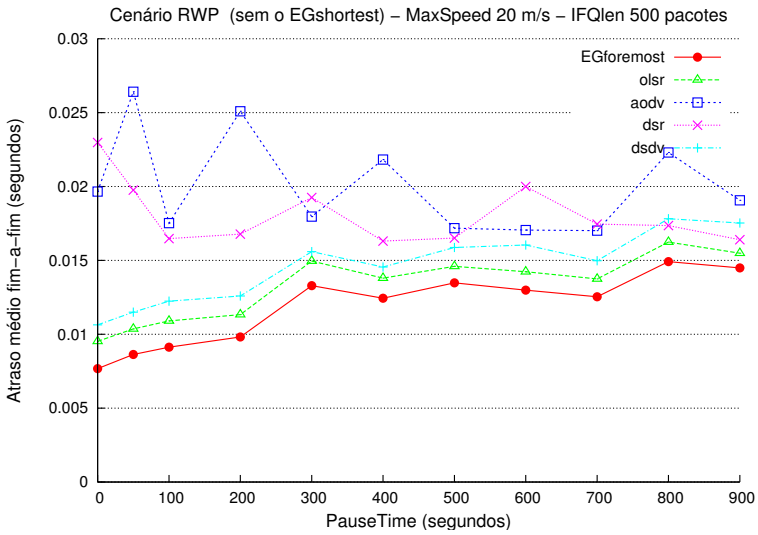


Figura 5.2.7.: Atraso médio fim-a-fim para os pacotes entregues com sucesso no Modelo RWP

O $EG_{shortest}$ foi removido deste gráfico, pois seus valores estão em uma escala bem maior, pois os caminhos mais curtos podem aparecer bem a frente no tempo e portanto a média dos seus valores é alta. Uma discussão particular será feita na próxima seção.

Em relação aos protocolos reativos, AODV e DSR, estes não tem uma curva suave, atribuímos este comportamento ao atraso induzido pela descoberta de rotas. Notamos também que a variância nos protocolos deste tipo também é relativamente alta.

5.3. Discussão sobre o Atraso Fim-a-Fim

As simulações realizadas no cenário *Random Waypoint* evidenciam uma característica importante do algoritmo JORNADA MAIS CURTA: Os valores de atraso fim-a-fim são relativamente altos (veja figura 5.3.1). O algoritmo $EG_{shortest}$ busca por caminhos mais curtos levando em conta todo o tempo da simulação (i.e. 900 segundos), com isso, pacotes enviados por algum nó no instante 10 segundos, tem chance de chegar ao destino somente no instante 899 segundos, caso a JORNADA MAIS CURTA só apareça nos instantes finais. Este comportamento é, portanto, dependente da duração da simulação. Quanto mais longa esta for, maior será o valor do atraso médio fim-a-fim, pois existe maior chance de um caminho mais curto aparecer bem a frente no tempo. Consequentemente, o tamanho dos caminhos tendem a ser cada vez mais curtos.

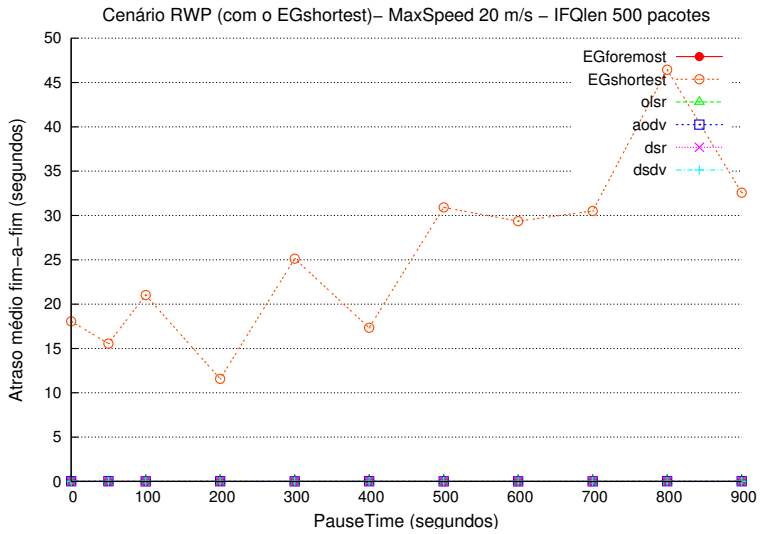


Figura 5.3.1.: Atraso médio fim-a-fim do $EG_{shortest}$ somente, no modelo RWP

Para termos uma melhor noção do comportamento do atraso fim-a-fim, plotamos um gráfico com o **tempo de atraso** para cada pacote entregue com sucesso, figura 5.3.2, sendo que cada círculo aberto representa um pacote. No eixo-x tem-se o horário de envio dos pacotes de 0 a 900 segundos, e no eixo-y tem-se o atraso do pacote no momento do recebimento. Um exemplo: dentre os pacotes enviados próximo aos 300 segundos de simulação, alguns foram entregues com um atraso de 40 segundos e outros com atrasos de 150 segundos Já entre os instantes 450 e 500 segundos muitos pacotes levaram em torno

de 250 a 200 segundos para serem entregues (chegando ao destino provavelmente entre os instantes 650 e 750 segundos). Isso se deve à algum enlace importante que "apareceu" no instante 450 segundos, e muitos pacotes haviam sido agendados para envio nesse momento.

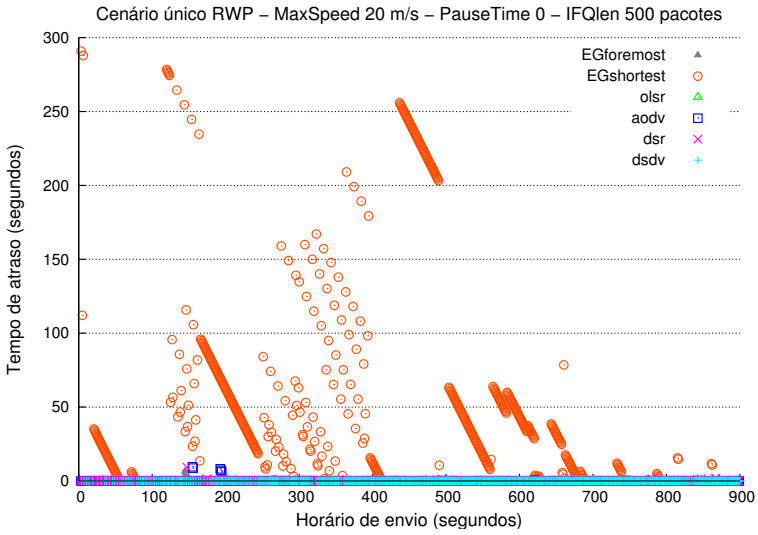


Figura 5.3.2.: Tempo de atraso na entrega de pacotes para um cenário de simulação com PauseTime 0, no modelo RWP

O formato linear destes pontos são decorrentes das taxas de envio constantes (2 pacotes/segundo). Para melhorar a visibilidade do gráficos, plotamos somente um vinte avos da totalidade dos pacotes.

Essa discussão traz a necessidade da inclusão de uma nova variável no algoritmo, denominada: LimiteDePredição, a qual irá limitar o horário de chegada dos pacotes ao calcular a Jornada Mais Curta. Esta alteração para adicionar o LimiteDePredição pode ser feita na Linha 5(a)i (Seção 3.3.2, Página 31) do algoritmo $EG_{shortest}$. O objetivo do LimiteDePredição é, em outras palavras, definir um valor máximo de atraso fim-a-fim no cálculo das *jornadas* de cada pacote.

A discussão sobre esse assunto já havia sido iniciada por A.Ferreira no relatório de pesquisa [32], no qual comenta a existência desta característica.

5.4. Resultados - Modelo Intermitente

O cenário Intermitente se mostra mais realista para o caso de redes FSDN, no sentido que os nós tem posições fixas e os momentos em que estes se ligam/desligam podem ser previsíveis.

Nos próximos testes, durante as simulações alteramos os valores de SleepProb de **0 a 50%**. Alta probabilidade para dormir significa que a rede tem baixa conectividade, isto é, as conexões entre os nós são esparsas, vários nós estão desconectados uns dos outros pois vários estão dormindo. O outro parâmetro, HoldTime de **15 a 180 segundos**, significa dinamicidade, i.e. quão freqüentemente as conexões entre os nós são alteradas, valores baixos de HoldTime significam alta dinamicidade, e vice versa.

Para se ter uma idéia do comportamento destas variáveis, a figura 5.4.1 mostra a dinâmica dos cenários com o modelo de mobilidade Intermitente. Em cenários de baixa conectividade (50% de SleepProb) o número de alterações na topologia da rede (i.e. variações nas possíveis rotas) varia de 7.500 com HoldTime de 15s para 90.000 com HoldTime de 180s. Uma variação de 12 vezes, o que mostra uma certa linearidade entre a dinâmica dos cenários e à variável HoldTime.

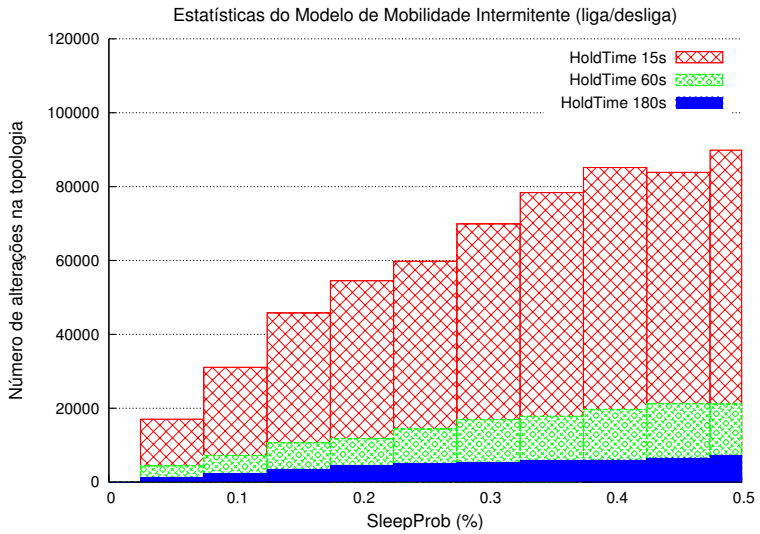
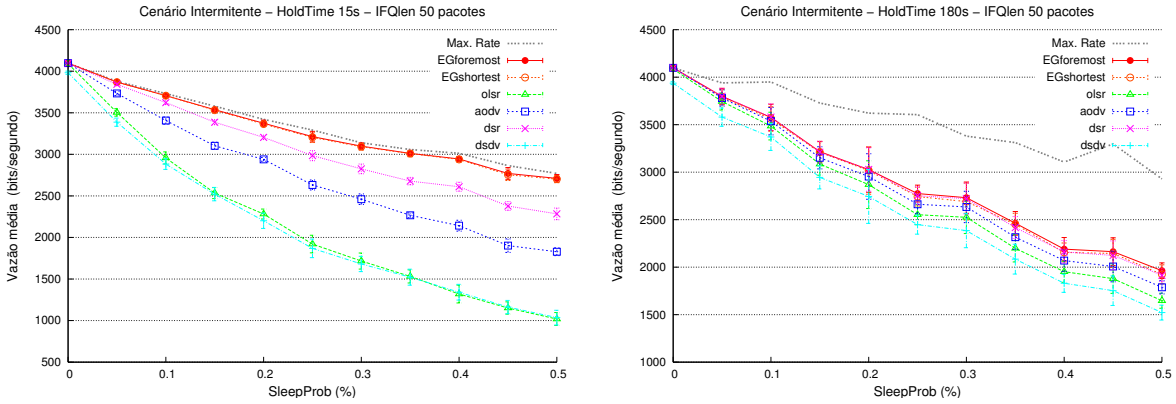


Figura 5.4.1.: Número de alterações na topologia da rede para diferentes valores de SleepProb no Modelo Intermitente.

5.4.1. Vazão Média nos Cenários do Modelo Intermitente

Na figura 5.4.2 pode-se observar os valores de **vazão** em função de SleepProb para dois cenários distintos do Modelo Intermitente: o gráfico da esquerda com os valores de HoldTime igual a 15s (maior dinamicidade) e o da direita com HoldTime de 180s (menor dinamicidade).

A linha cinza tracejada no topo é o limite máximo teórico de vazão e corresponde à taxa em que os pacotes são enviados pela origem na rede. Esta taxa decai inversamente com os valores de SleepProb, pois os nós transmissores também podem ser escolhidos para dormir por certos períodos de tempo, e ao dormir estes nós não geram pacotes. Na figura 5.4.2(a), observa-se que os valores de *MaxRate* variam de 4096 bits/s para 2728 bits/s em cenários com SleepProb igual a 50%, um decaimento de 33,4% no fluxo de pacotes na rede.



(a) HoldTime 15s

(b) HoldTime 180s

Figura 5.4.2.: Vazão média em função de SleepProb (conectividade) no Modelo Intermitente.

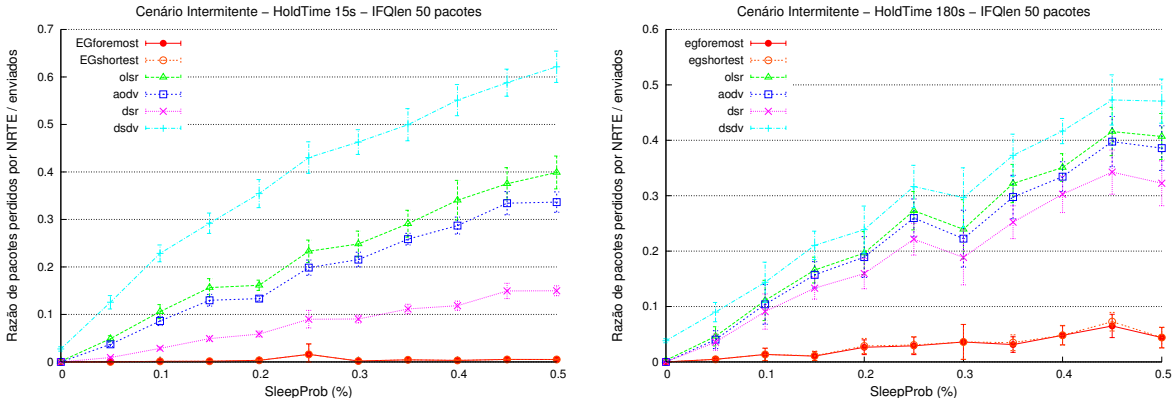
Em ambos os gráficos, os protocolos $EG_{foremost}$ e $EG_{shortest}$ obtiveram os melhores desempenhos na métrica de vazão média. Mas, nota-se que no cenário de maior dinamicidade a vazão de ambos protocolos baseados em EG obtiveram valores bem próximos ao máximo teórico (as curvas estão bem próximas do *MaxRate*), veja figura 5.4.2(b). Por outro lado, no gráfico da figura 5.4.2(b) os valores de ambos protocolos baseados em EG descolam do máximo e mostram valores de vazão próximos aos dos outros protocolos (uma perda de 37% se comparado ao *MaxRate*). Este menor desempenho será explicado através das

perdas de pacotes, na próxima subseção.

5.4.2. Perda de Pacotes nos Cenários do Modelo Intermitente

A figura 5.4.3, mostra a perda de pacotes na camada de roteamento por inexistência de rotas (NRTE), nestes casos o protocolo não encontrou uma rota válida origem-ao-destino. Assim como na vazão média, mostramos os resultados para dois valores de HoldTime.

O resultados com HoldTime 15s (figura 5.4.3(a)) mostram uma perda de pacotes mínima, condizendo com os bons resultados de vazão média na seção anterior. Já o gráfico da direita (figura 5.4.3(b)) mostra que a perda de pacotes por NRTE é muito pequena (média de 3,4%) em relação ao decaimento bem pronunciado (37%) de vazão média mostrado na figura 5.4.2(a).



(a) HoldTime 15s

(b) HoldTime 180s

Figura 5.4.3.: Razão de pacotes perdidos por NRTE em função de SleepProb (conectividade) no Modelo Intermitente.

Os valores obtidos com os protocolos $EG_{foremost}$ e $EG_{shortest}$ na figura 5.4.3 nos parecem ser os limitantes inferiores para essa métrica, i.e. se os protocolos baseados em EG jogam fora um pacote por NRTE, significa que este caminho não existe e não existirá em nenhum instante durante a simulação. Note, porém, que existe uma pequena variação nos valores obtidos nesses dois protocolos. Atribuímos essa diferença aos casos em que os atrasos provocados por longos momentos de desconexão ocasionam filas cheias, e que posteriormente induzem a uma perda de pacotes por decorrência de atrasos extras. Nestes casos, o atraso extra faz com que alguns pacotes deixem de utilizar um enlace que estaria disponível caso

o atraso não tivesse ocorrido.

Perda de Pacotes por IFQ

A razão para o fraco desempenho na métrica vazão em cenários de baixa dinamicidade não está relacionado a inexistência de rotas, mas sim ao transbordo na fila do enlace (IFQ).

O gráfico da figura 5.4.4 mostra o número de pacotes perdidos por transbordo da IFQ para o cenário de baixa dinamicidade (HoldTime 180s), somente o $EG_{foremost}$ e $EG_{shortest}$ perdem uma grande quantidade de pacotes por IFQ (chegando a 27,5% de perdas por esta razão).

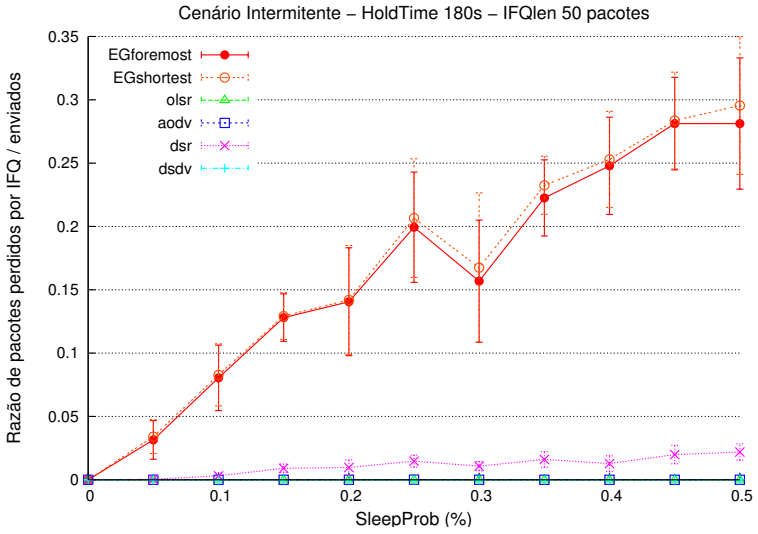


Figura 5.4.4.: Número de pacotes perdidos por IFQ com HoldTime 180s no Modelo Intermitente.

O comportamento intrínseco dos protocolos baseados em EG de agendar pacotes para serem enviados somente quando o nó vizinho acorda, trás a tona o problema dos ***gargalos***, em que vários nós agendam o envio de pacotes para mesmo exato momento, que é o primeiro instante posterior ao que o nó em questão é ativado, assim a fila de pacotes no enlace (IFQ) não tem capacidade para receber rajada de dados e vários pacotes são jogados fora.

Nas simulações realizadas nos cenários RWP, o protocolo $EG_{foremost}$ não sofreu perdas significativas por IFQ, atribuímos esse fato a alta mobilidade da topologia naqueles cenários, fazendo com o que o $EG_{foremost}$ encontrasse rapidamente uma *jornada* que che-

gasse mais cedo ao destino. Em contra-partida, devido a maior mobilidade e na busca pelo caminho mais curto, o $EG_{shortest}$ tende a agendar o envio dos pacotes para um horário mais distante, e portanto, ocorre o transbordamento das filas.

É importante ressaltar que os resultados do $EG_{foremost}$ e $EG_{shortest}$ no modelo Intermitente foram muito próximos, em alguns casos, inclusive, compartilham da mesma curva no gráfico (um em cima do outro). Diferentemente do cenário RWP, em que somente o algoritmo JORNADA MAIS CURTA mostrou um comportamento particular na perda de pacotes, os experimentos nesta seção mostram que ambos os protocolos tem um comportamento parecido e ambos evidenciam o problema dos *gargalos*.

Histograma de Perda de Pacotes no tempo

Para ilustrar o problema dos *gargalos*, a figura 5.4.5 contém um histograma com o número de pacotes perdidos por instante de tempo em um cenário com SLEEPPROB de 50% e HOLDTIME de 180s. Nele podemos observar que os pacotes são perdidos em forma de rajada (e.g. 800 pacotes são perdidos entre os instantes 310 e 320s), e novamente, devido a algum ou alguns nós que dormem por longos períodos de tempo, e muitos pacotes são agendados para serem enviados no instante em que estes nós acordam. Logo, estes pacotes são jogados fora de uma só vez pelo enchimento da fila na interface de rede (IFQ).

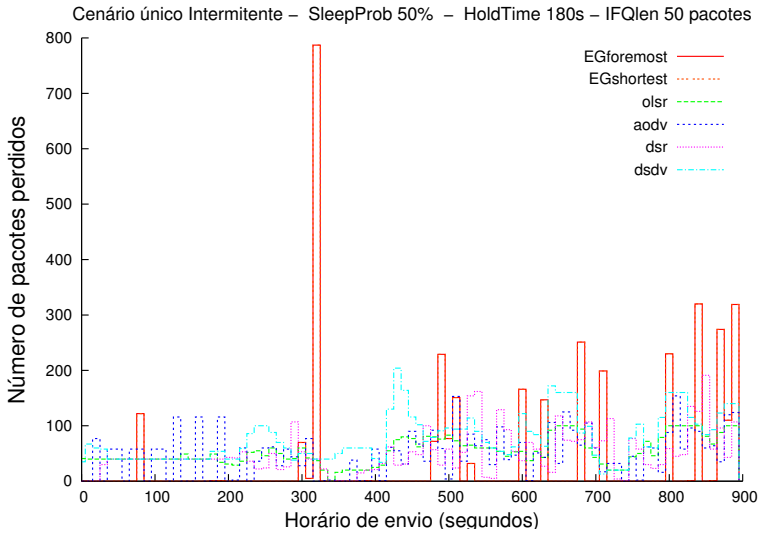


Figura 5.4.5.: Histograma com o número de pacotes perdidos por instante de tempo para um cenário (SLEEPPROB 50% e HOLDTIME 180s) no Modelo Intermitente.

5.4.3. Consumo Médio de Energia nos Cenários do Modelo Intermitente

Na figura 5.4.6 mostramos o consumo de energia por pacote entregue, calculado da mesma forma que no cenário RWP. Neste, ambos os protocolos baseados em EG obtiveram os melhores resultados. O interessante neste cenário, é que os valores obtidos nos dois protocolos baseados em EG foram praticamente os mesmos. Atribuímos esse comportamento à menor dinamicidade da rede nos cenários do Modelo Intermitente.

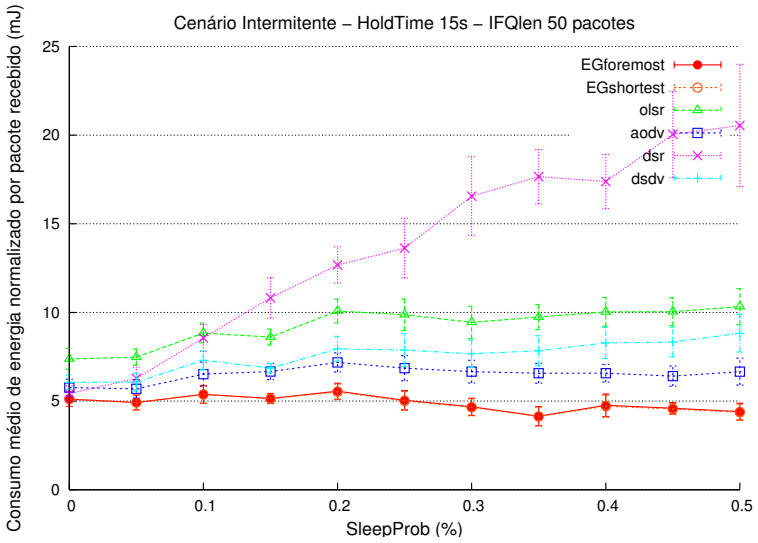


Figura 5.4.6.: Consumo médio de energia normalizado pelo número de pacotes entregues com sucesso (HoldTime 15s), no Modelo Intermitente.

5.4.4. Atraso Fim-a-Fim nos Cenários do Modelo Intermitente

Em relação ao atraso médio fim-a-fim (figura 5.4.7), no cenário Intermitente os valores do $EG_{foremost}$ foram os menores em praticamente todas as simulações, mas houve alguns casos em que o OLSR obteve resultados melhores, o que em primeira instância invalidaria nossa argumentação de que ele é um limitante inferior para esta métrica. Após uma análise mais detalhada, reparamos que os valores de $EG_{foremost}$ não eram os melhores nos cenários em que ocorria congestão da fila IFQ. Constatamos, que o atraso no algoritmo baseado em EG estava sendo aumentado devido a uma grande quantidade de pacotes esperando na fila para serem enviados, ocasionando atrasos extras.

Também foi observado que um enlace em congestão ocasiona uma sobrecarga no meio

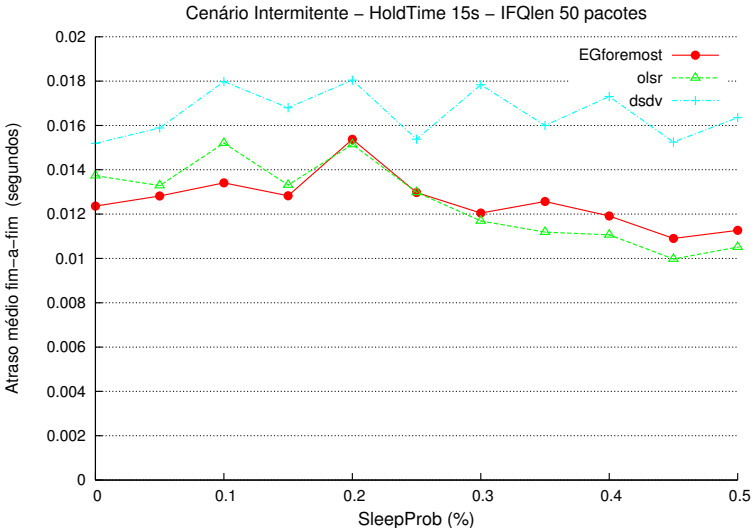


Figura 5.4.7.: Atraso médio fim-a-fim para os pacotes entregues com sucesso em todos os protocolos no Modelo Intermitente. Observe que para valores de baixa conectividade, os $EG_{foremost}$ não obteve os melhores resultados.

físico, e nós na região de alcance do enlace congestionado também sofrem um atraso extra pois não conseguem acesso ao meio físico rapidamente.

Os outros protocolos, não sofrem desse problema pois como não trabalham com políticas de reenvio com tempos longos, as filas IFQ não trabalham tão cheias quanto nos protocolos baseados em EG. Portanto, os pacotes ou estão sendo jogados fora ou estão sendo transmitidos diretamente, sem permanecer em filas.

5.5. Discussão sobre Gargalos, Congestão e Fluxos

As perdas de pacotes observadas nas Seções 5.2.2 e 5.4.2 dos protocolos baseados em EG são, como já mencionado, relativas ao problema dos *gargalos*.

Um exemplo de caso intuitivo (mas não o único) da existência destes gargalos pode ser visto na figura 5.5.1. Neste caso, diversos fluxos provenientes de diferentes origens no lado esquerdo do grafo **A,B,C,D** estão enviando dados para alguns nós no lado direito **G,H**, e todos eles precisam passar por um enlace (aresta) na região central (E-F). Caso este enlace fique indisponível por um longo tempo, muitas pacotes permanecerão em filas aguardando o primeiro momento de existência da aresta (neste caso, o instante 7), para enviar os dados.

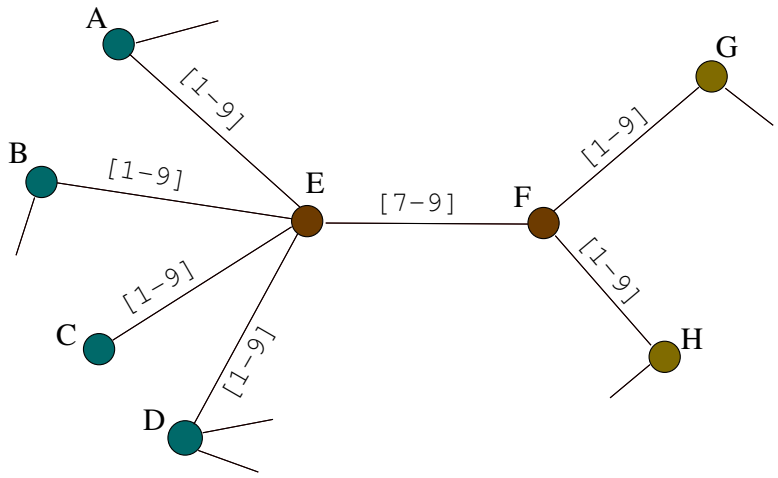


Figura 5.5.1.: Exemplo de possível cenário para existência de *gargalos*. No qual os nós **A,B,C,D** transmitem dados ininterruptamente para **G,H**, mas a aresta (E,F) só existe nos instantes 7 a 9, ocasionando perdas se as filas não tiverem um tamanho suficiente.

No nosso exemplo, caso o fluxo em cada origem seja de 10 pacotes/segundo, no instante 7, o nó **E** terá recebido em torno de 70 pacotes de cada origem, totalizando 240 pacotes para serem enviados de uma só vez para o nó **F**. Se a fila para envio no enlace não tiver capacidade para suportar tal quantidade, diversos pacotes podem ser perdidos.

A existência dos gargalos está relacionada à falta de mecanismos na teoria de EG para balancear os fluxos de dados no tempo de maneira ótima, porém para existir tal mecanismo seria necessário que o fluxo de dados fosse conhecido de antemão pelos nós da rede, assim como ocorre na topologia. Existem diversos trabalhos teóricos relacionados a este tema, mas poucos incluem a questão do tempo. Discussões sobre o assunto podem ser encontradas em [33, 41, 48, 77, 80]. Köeler e Skutella em [51], mostram que o problema de maximizar o fluxo no tempo é um problema NP-difícil. A seguir, experimentamos algumas técnicas para diminuir as perdas nos *gargalos*, mas sem um embasamento teórico mais formal.

Experimentos

Experimentamos três técnicas empíricas para tentar diminuir as perdas por IFQ. Elas estão descritas a seguir e ilustradas na tabela 5.1:

1. **Jitter**: Adicionar um atraso induzido (*jitter*) no horário de envio de cada pacote, os

menores valores de atraso induzindo que melhoraram substancialmente o desempenho das perdas foram de meio segundo (0,5 segundo), e portanto utilizaremos esse valor nos experimentos a seguir;

2. **SmartJitter**: Esta segunda técnica, proposta por nós, visa adicionar um atraso induzido no envio, como em (1), mas de maneira mais elaborada. Se o número de pacotes programados para transmissão em um enlace for menor que o tamanho da fila IFQ, nenhum *jitter* é adicionado. O atraso será adicionado somente aos pacotes que estão sendo agendados para transmissão quando um enlace naquele instante já está congestionado (IFQ cheio), isto é, ao invés de escolher um atraso aleatório, cada pacote terá o seu horário de envio "reagendado" de acordo com quantos pacotes já estão programados para serem enviados naquele instante para determinado nó.

   O cálculo do reagendamento do horário de envio (SmartJitter) de um pacote sendo enviado pelo nó u para o nó v no instante t_v pode ser descrito por este esquema:

   if $npkts[u, v, t_v] \leq IFQlen$ **then**
   $smartJitter = 0$
   else
   $smartJitter = \delta * (npkts[u, v, t_v] - IFQlen)$
   end if
   $npkts[u, v, t_v]$++
   Reagenda envio para o instante: $t_v + smartJitter$

   Sendo que $npkts[u, v, t_v]$ é o número de pacotes já programados para envio usando a aresta (u,v) no instante t_v. $IFQlen$ é o tamanho máximo da fila na interface de rede e δ é o tempo do *jitter* adicionado a cada pacote reagendado, no nosso caso estamos utilizando o tempo de 3,5 ms, que é o tempo médio de percurso entre dois nós, como estimado na Seção 4.6 (página 48) para pacotes de tamanho 256 bytes a uma taxa de 2 pacotes/segundo;

3. **Aumentar o tamanho da IFQ**: Incrementar o tamanho do fila de 50 para 500 pacotes.

Tabela 5.1.: Esquemas de agendamento para diminuir perdas por IFQ

Pacote	IFQlen 500	Jitter 0,5s	SmartJitter
p_1	t	t + rnd(0,5)	t
p_2	t	t + rnd(0,5)	t
...	t	t + rnd(0,5)	t
p_{50}	t	t + rnd(0,5)	t
p_{51}	t	t + rnd(0,5)	$t + \delta$
p_{52}	t	t + rnd(0,5)	$t + 2 * \delta$
p_n	t	t + rnd(0,5)	$t + (n - IFQlen) * \delta$

Resultados

Os resultados obtidos utilizando essas três alternativas podem ser observados nas figuras 5.5.2 e 5.5.3. A curva $EG_{foremost}$ é a curva original obtida nas simulações com os cenários do Modelo Intermitente, mostrada na figura 5.4.4 (página 64) e é usada como referência.

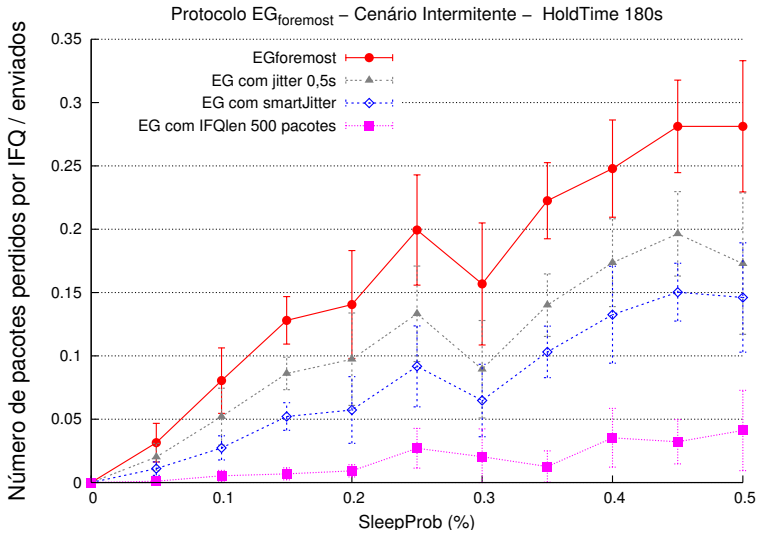


Figura 5.5.2.: Diferentes alternativas para diminuir as perdas por IFQ (*jitter* induzido, SmartJitter e aumentar o tamanho da fila para 500 pacotes).

Podemos observar nestes resultados, que a técnica *jitter* já mostra uma melhora de 30% em relação aos pacotes perdidos, porém, por induzir um atraso no envio de cada pacote, a métrica de atraso fim-a-fim é prejudicada, na figura 5.5.3 vemos que o atraso médio subiu

de 14 para 23 segundos (os valores do $EG_{foremost}$ são 40% menores).

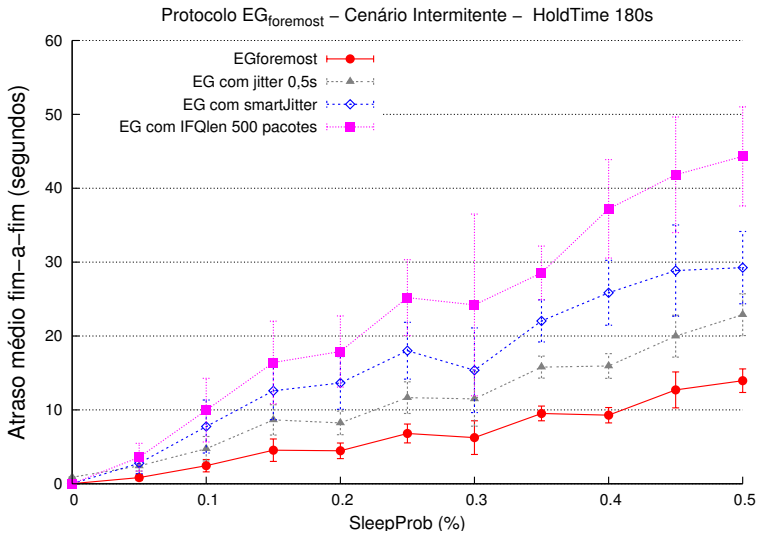


Figura 5.5.3.: Atraso médio fim-a-fim nas diferentes alternativas para diminuir as perdas por IFQ.

Usando o SmartJitter, o número de pacotes perdidos melhorou 47% se comparado à curva original $EG_{foremost}$ nos cenários de baixa conectividade (SleepProb 50%). Mostrando que esta solução simples provê um bom desempenho, mas, assim como o *jitter* sofre de um razoável aumento no atraso médio.

Os resultados obtidos usando-se uma fila IFQ com tamanho aumentado de 50 para 500 pacotes são os melhores em relação ao número de pacotes perdidos, mas, com relação ao atraso médio, os seus valores são bem ruins, com um aumento de até 3 vezes em relação à curva de original. Este atraso é proveniente do longo tempo que um pacote fica na fila esperando para ser enviado, e devido à colisões no meio físico, esses pacotes podem potencialmente ser reenviados diversas vezes. Por fim, podemos dizer que estas técnicas apresentadas (*jitter*, SmartJitter e aumento de IFQlen) melhoram o desempenho em relação à quantidade de pacotes perdidos por IFQ mas têm um compromisso com atraso médio.

5.6. Considerações sobre os Resultados

Nos cenários de mobilidade RWP, o protocolo $EG_{foremost}$ obteve bons resultados, como esperado. Em diversas métricas ele obteve um desempenho próximo ao limite teórico. Em

relação aos pacotes perdidos por não existir rota, apontamos que os valores obtidos com os protocolos $\text{EG}_{foremost}$ e $\text{EG}_{shortest}$ podem ser usados como referência.

Esta primeira implementação de protocolos de roteamento baseado em grafos evolutivos destacou alguns pontos a serem estudados com mais profundidade. Um deles, o problema dos *gargalos*, que ocorre quando um nó importante fica indisponível por muito tempo. Esta ausência causa sobrecarga na rede no momento de seu retorno, quando pacotes podem ser jogados fora devido a colisões e filas cheias.

Este comportamento foi mais visível nas simulações com o modelo de Mobilidade Intermitente, no qual os protocolos baseados em grafos evolutivos obtiveram resultados muito parecidos e ambos perderam muitos pacotes nos cenários de baixa dinamicidade. Embora, a existência de gargalos já havia sido identificada nos cenários RWP com a grande quantidade de pacotes perdidos pelo $\text{EG}_{shortest}$.

É importante realçar que os valores altos de atraso médio origem-ao-destino é uma característica intrínseca das redes de comunicação dinâmicas. No caso dos protocolos baseados em *EG* utilizando a métrica Jornada que Chega Mais Cedo, o atraso médio obtido é o menor possível para cada pacote. Se atrasos longos não são desejáveis, então uma política de jogar fora pacotes velhos, que estão esperando a muito tempo, pode ser uma alternativa.

Resumo

Neste capítulo, foram mostrados os resultados experimentais e análises de diversas métricas de desempenho nas simulações de dois cenários de mobilidade. Os problemas relacionados a fluxos de dados ficaram evidentes, e como não há base teórica difundida, propomos e experimentamos algumas técnicas alternativas para diminuir perdas por congestão no enlace. A seguir, as considerações finais deste trabalho.

6. Conclusão

O conceito de *grafos evolutivos*, introduzido recentemente, foi proposto como uma abstração formal para redes dinâmicas, no qual a noção de *jornadas* é adicionada à teoria de grafos tradicional, trazendo a noção do tempo.

Os algoritmos e idéias obtidos com este modelo são teoricamente muito eficientes, mas, no entanto, antes deste trabalho não existiam estudos do uso deste modelo em situações práticas, como por exemplo, na construção de protocolos de roteamento para redes dinâmicas.

Neste trabalho, estudamos diversos algoritmos da teoria de *grafos evolutivos*, e dentre eles escolhemos alguns que pudessem ser utilizados na construção de um protocolo de roteamento. Assim, o nosso objetivo foi analisar a aplicabilidade desta teoria na construção de protocolos de roteamento eficientes em cenários realistas. Optamos, no entanto, por implementar um protocolo de roteamento da maneira mais simples possível, sem nos preocuparmos com a troca de mensagens de controle ou com um funcionamento distribuído. O objetivo desta abordagem foi elaborar protocolos que expressassem os limites teóricos de capacidade de uma rede com a utilização prática, sem que esforços fossem despendidos em tarefas "administrativas".

Utilizamos um simulador de redes para realizar esta implementação. Dentre os diversos simuladores existentes, optamos por utilizar o tradicional NS2, que é utilizado por grande parte dos estudos relacionados na área. Mas, apesar de ser amplamente utilizado, a sua curva de aprendizado é acentuada.

Um fator importante na elaboração de simulações é fazer uma boa escolha dos diferentes parâmetros e cenários que serão utilizados durante as simulações. Aqui, novamente, optamos por escolher um cenário tradicional, que é o *Random Waypoint*. No decorrer dos experimentos foi necessário desenvolver um novo cenário de mobilidade para os nós, o Modelo Intermitente, para que conseguíssemos captar com maior precisão as características

de redes com comportamento previsível.

Um dos grandes desafios no início dos trabalhos foi a escolha dos cenários e das métricas de desempenho que deveriam ser utilizados para avaliar os protocolos. Com isso, para cada experimento analisamos diversas variáveis até que alguma expusesse valores interessantes.

Por exemplo, em um primeiro momento os resultados obtidos na métrica de atraso médio fim-a-fim não foram animadores, devido aos valores muito altos obtidos pelos protocolos baseados em *grafos evolutivos*. Com o decorrer do trabalho aprimoramos essa métrica e analisamos somente os pacotes que eram entregues com sucesso por todos os algoritmos, e dessa forma conseguimos mensurar o desempenho do algoritmo Jornada que Chega Mais Cedo.

Com todos os resultados experimentais obtidos e com as diversas soluções específicas propostas, acreditamos ter dado um grande passo na aplicabilidade de grafos evolutivos.

6.1. Contribuições

Neste trabalho, implementamos dois protocolos de roteamento para redes móveis ad hoc baseados na teoria de *grafos evolutivos*, são eles: Jornada que Chega Mais Cedo e Jornada Mais Curta. Realizamos extensivos experimentos utilizando o simulador de redes NS2, sendo que diversas métricas de desempenho foram analisadas e os resultados foram comparados com outros quatro protocolos conhecidos: AODV, DSR, OLSR e DSDV.

A contribuição do nosso trabalho foi mostrar que a teoria de *grafos evolutivos* como um todo é uma ferramenta poderosa para o desenvolvimento e aprimoramento de protocolos de roteamento para redes dinâmicas. Os protocolos baseados EG tiveram um desempenho próximo ao ótimo teórico em diversas métricas. Podendo, portanto, serem utilizados também como referencial no desenvolvimento de outros algoritmos para esta área de pesquisa, i.e. como *benchmarking* para outros protocolos.

Ficou claro que a existência dos *gargalos* está relacionada à falta de mecanismos na teoria de *grafos evolutivos* para balancear os fluxos de dados no tempo de maneira conveniente, porém para existir tal mecanismo seria interessante que o fluxo de dados fosse conhecido de antemão pelos nós da rede, assim como ocorre com a topologia.

Porém, propusemos três alternativas práticas para diminuir o número de pacotes perdidos por fila cheia no enlace. A primeira, adicionar um atraso induzido no envio de cada pacote, surtiu efeito mas valores muito altos de atraso foram necessários para melhorar o desempenho. Introduzimos a técnica de SmartJitter, como opção mais elaborada para

diminuir a congestão e os resultados foram animadores. A terceira opção, mais simples, é aumentar o tamanho da fila. Mostramos que existe um compromisso nestas soluções entre a redução do número de pacotes perdidos e o aumento do atraso médio fim-a-fim.

O desenvolvimento de algoritmos adaptativos para abordar este problema talvez seja a melhor solução, antecipando a congestão de nós e encontrando rotas alternativas. No entanto, não encontramos nenhum trabalho que maximize o fluxo em *grafos evolutivos*, o que é um tópico interessante de pesquisa.

Os atrasos fim-a-fim nas simulações com o protocolo $\text{EG}_{shortest}$ são ainda maiores, e discutimos o fato de que: quanto mais longa a simulação, maiores serão os valores de atraso na métrica Jornada Mais Curta. A noção de LimiteDePredição é uma alternativa sugerida para diminuir o atraso médio fim-a-fim neste protocolo, e em contra-partida, diminuir o número de perdas por congestão.

No desenvolver deste trabalho foram publicados dois artigos em conferências internacionais [58, 59] e um em simpósio nacional [60].

6.2. Trabalhos Futuros

Esta primeira implementação de *grafos evolutivos* abriu as portas para diversos pontos a serem pesquisados com mais profundidade. Além da implementação da outra métrica: Jornada Mais Rápida, almeja-se estudar um protocolo adaptativo para lidar com o problema dos *gargalos*. O estudo de fluxos em *grafos evolutivos* é um problema em aberto, e inclui o estudo de uma rede em que, além da topologia, o padrão de comunicação entre os nós também é conhecido.

Outra extensão natural para este trabalho está relacionada aos desvios (diferenças) na rede dinâmica prevista, que é o caso em que o EG não representa mais a realidade da topologia. Este problema motiva a utilização de teorias de comportamento estocástico para representar uma rede dinâmica de forma mais realista.

A. Exemplo de Simulação no NS2

NS2 - Arquivo de Configuração

Segue exemplo de configuração para a realização de uma simulação no NS2, composta por uma rede sem fio com 5 nós móveis, conectados através de um enlace 802.11. A simulação tem duração de 100 segundos. Os arquivos contendo o cenário de mobilidade (`cenario.tcl`) e o padrão de comunicação (`comunicacao.tcl`) são mostrados mais adiante. O protocolo de roteamento escolhido foi o $\text{EG}_{foremost}$, sendo que o grafo evolutivo será lido do arquivo `eg.txt`, e é mostrado no final deste apêndice.

Os traços da simulação serão gravados em um arquivo de nome `saida.tr` para posterior análise, a qual é feita utilizando programas auxiliares (geralmente em linguagens Awk ou Perl).

```
# ----------------------------------------------------------------------
# Opcoes de execucao
# ----------------------------------------------------------------------

set val(chan)    Channel/WirelessChannel  ;# tipo do canal
set val(prop)    Propagation/TwoRayGround ;# modelo de propagacao
set val(netif)   Phy/WirelessPhy          ;# interface de rede
set val(mac)     Mac/802_11               ;# tipo MAC
set val(ifq)     Queue/DropTail/PriQueue  ;# tipo de fila de rede
set val(ll)      LL                       ;# tipo de enlace
set val(ant)     Antenna/OmniAntenna      ;# modelo de antena
set val(ifqlen)  50                       ;# tamanho da fila IFQ
set val(nn)      2                        ;# numero de nos
set val(rp)      Egraph                   ;# protocolo de roteamento

Agent/Egraph set eg_file          "eg.txt" ;# arquivo com EG
Agent/Egraph set edge_delay_      0.035    ;# tempo de percurso
Agent/Egraph set routing_metric_  1        ;# Algoritmo: EGforemost
```

```
#---------------------------------------------------------------------
# Programa principal
#---------------------------------------------------------------------

set ns_     [new Simulator]
set topo    [new Topography]

create-god $val(nn)
$topo load_flatgrid 600 600; # Dimensoes do terreno

set tracefd [open saida.tr w]
$ns_ trace-all $tracefd

$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON

for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns_ node]
}

#---------------------------------------------------------------------
# Importa arquivo com cenario de mobilidade
#---------------------------------------------------------------------
source "cenario.tcl"

#---------------------------------------------------------------------
# Importa arquivo com padrão de comunicacao
#---------------------------------------------------------------------
```

```
source "comunicacao.tcl"

#----------------------------------------------------------------------
# Programa o termino da simulacao para o instante 100s
#----------------------------------------------------------------------
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 100.0 "$node_($i) reset";
}
$ns_ at 100.01 "$ns_ flush-trace"
$ns_ at 100.02 "close $tracefd"
$ns_ at 100.03 "$ns_ halt"

puts "Iniciando Simulacao..."
$ns_ run
```

Esse *script* é gravado em um arquivo para ser executado pelo NS2 através da linha de comando.

NS2 - Arquivo com Modelo de Mobilidade

Os cenários de mobilidade no NS2 são compostos, geralmente, por seqüencias de comandos `setdest`, e são utilizados da seguinte maneira: `$ns_ at T "$node_(N) setdest POSX POSY VEL"`. Sendo `T` o instante para execução do comando, `POSX` e `POSY` as coordenadas de destino e `VEL` a velocidade durante o trajeto. Além deste também estamos usando os comandos liga/desliga (*on/off*) nos cenários criados neste trabalho. Também é preciso definir a posição inicial (x,y) de cada nó da rede. A seguir, um arquivo contendo cenário de mobilidade misto (contendo comandos 'setdest', 'on' e 'off'). Nele, são definidas as posições iniciais bem como movimentações de 5 nós.

```
#----------------------------------------------------------------------
# Posicao inicial (X,Y)
#----------------------------------------------------------------------

$node_(0) set X_   0.0
$node_(0) set Y_ 100.0

$node_(1) set X_  60.0
$node_(1) set Y_ 200.0
```

```
$node_(2) set X_ 280.0
$node_(2) set Y_ 200.0

$node_(3) set X_ 400.0
$node_(3) set Y_ 100.0

$node_(4) set X_  50.0
$node_(4) set Y_  50.0


# ------------------------------------------------------------------
# Movimentacoes dos nodes (comandos 'setdest', 'on' e 'off')
# ------------------------------------------------------------------
$ns_ at   0.0 "$node_(1) off"
$ns_ at  10.0 "$node_(1) on"

$ns_ at 20.00 "$node_(2) off"
$ns_ at 35.00 "$node_(2) on"

$ns_ at 40.0 "$node_(4) setdest 200.0 100.0 15.0"
$ns_ at 45.0 "$node_(4) off"
$ns_ at 55.0 "$node_(4) on"

$ns_ at 65.0 "$node_(1) setdest 10.0  120.0 15.0"
$ns_ at 75.0 "$node_(2) setdest 350.0 120.0 15.0"
$ns_ at 80.0 "$node_(4) setdest 200.0 500.0 15.0"
```

NS2 - Arquivo com Cenário de Tráfego

Segue arquivo `cenario.tcl` contendo dois fluxos de comunicação CBR/UDP, com 256 bytes por pacote a uma taxa de 10 pacotes/segundo, para ser utilizado em uma simulação do NS2. O primeiro fluxo, do nó 0 para o nó 3, tem início no instante 5 e término no instante 90 segundos. Já o segundo fluxo, do nó 1 para o nó 2, tem início no instante 10 segundos e permanece até o fim da simulação.

```
# ------------------------------------------------------------------
set udp_(0) [new Agent/UDP]
```

```
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/LossMonitor]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.1
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 5.0 "$cbr_(0) start"
$ns_ at 90.0 "$cbr_(0) stop"

#----------------------------------------------------------------------
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(1)
set null_(1) [new Agent/LossMonitor]
$ns_ attach-agent $node_(2) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 256
$cbr_(1) set interval_ 0.1
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 10.0 "$cbr_(1) start"
```

NS2 - Arquivo com Grafo Evolutivo

O formato do arquivo que contém o grafo evolutivo é muito simples: Cada par de nós é representado em uma linha, e é seguido por uma seqüência de intervalos de tempo separados por espaço. Cada linha: '`u v` t_1`,`t_2 `...` t_{n-1}`,`t_n', significa que o enlace (u,v) existe nos instantes definidos pela lista de intervalos t_1,t_2 até t_{n-1} a t_n.

Este arquivo é gerado pelo do programa `calcEG`, através da seguinte linha de comando:

```
calceg -i cenario.tcl -o cenario.out -g eg.txt -n 5 -t 100
```

Sendo que os parâmetros utilizados correspondem a:

- `-i`: arquivo de entrada com o modelo de mobilidade do NS2
- `-o`: arquivo de saída com estatísticas do cenário
- `-g`: arquivo texto contendo o grafo evolutivo

- `-n:` número de nós
- `-t:` tempo de simulação

A seguir, apresentamos o arquivo `eg.txt` contendo o grafo evolutivo para o cenário definido no início do apêndice.

```
# EG file scen-apendiceC.tcl num_nodes: 5 sim_time: 100.00 range: 250m
0 1 10.000001,100.00000
0 4  0.000000,34.999999 45.000001,59.999999
1 2 10.000001,19.999999 25.000001,46.682137
1 4 10.000001,34.999999 45.000001,62.165384
2 3  0.000000,19.999999 25.000001,100.00000
2 4 31.700425,34.999999 45.000001,64.666666
3 4 45.000001,59.999999
```

NS2 - Arquivo de Log

Um arquivo de traços do NS2 [57] pode ser gerado de duas maneiras distintas, uma usando o formato antigo (mais compacto, e mostrado aqui), e a outra usando o estilo mais novo (`$ns_ use-newtrace`), que é mais completo e foi usada nas simulações deste trabalho.

A seguir, exemplo de envio de um pacote do nó 0 para o nó 3. O pacote é enviado no instante 1,1 segundo e é recebido pelo nó 3 no instante 1,119 segundo. O nó 1 e 2 servem de roteadores, repassando o pacote adiante.

```
s 1.100000000 _0_ AGT  --- 1 cbr 256 [0 0 0 0] ---- [0:0 6:0 32 0]

r 1.100000000 _0_ RTR  --- 1 cbr 256 [0 0 0 0] ---- [0:0 6:0 32 0]
s 1.100003496 _0_ RTR  --- 1 cbr 276 [0 0 0 0] ---- [0:0 6:0 31 1]
s 1.100118496 _0_ MAC  --- 0 RTS 44 [cbe 1 0 0]
r 1.100470662 _1_ MAC  --- 0 RTS 44 [cbe 1 0 0]
```

```
s 1.100480662 _1_ MAC  --- 0 CTS 38 [b84 0 0 0]
r 1.100784829 _0_ MAC  --- 0 CTS 38 [b84 0 0 0]
s 1.100794829 _0_ MAC  --- 1 cbr 328 [13a 1 0 800] ---- [0:0 6:0 31 1]
r 1.103418996 _1_ MAC  --- 1 cbr 276 [13a 1 0 800] ---- [0:0 6:0 31 1]
s 1.103428996 _1_ MAC  --- 0 ACK 38 [0 0 0 0]
r 1.103443996 _1_ RTR  --- 1 cbr 276 [13a 1 0 800] ---- [0:0 6:0 31 1]

f 1.103586142 _1_ RTR  --- 1 cbr 276 [13a 1 0 800] ---- [0:0 6:0 30 2]
r 1.103733162 _0_ MAC  --- 0 ACK 38 [0 0 0 0]
s 1.104082996 _1_ MAC  --- 0 RTS 44 [cbe 2 2 0]
r 1.104435829 _2_ MAC  --- 0 RTS 44 [cbe 2 2 0]
s 1.104445829 _2_ MAC  --- 0 CTS 38 [b84 1 0 0]
r 1.104750662 _1_ MAC  --- 0 CTS 38 [b84 1 0 0]
s 1.104760662 _2_ MAC  --- 1 cbr 328 [13a 2 1 800] ---- [0:0 6:0 30 2]
r 1.107385496 _2_ MAC  --- 1 cbr 276 [13a 2 1 800] ---- [0:0 6:0 30 2]
s 1.107395496 _2_ MAC  --- 0 ACK 38 [0 2 0 0]
r 1.107410496 _2_ RTR  --- 1 cbr 276 [13a 2 1 800] ---- [0:0 6:0 30 2]
r 1.107700329 _2_ MAC  --- 0 ACK 38 [0 2 0 0]

f 1.115402150 _2_ RTR  --- 1 cbr 276 [13a 2 2 800] ---- [0:0 6:0 29 3]
s 1.115677150 _2_ MAC  --- 0 RTS 44 [cbe 3 2 0]
r 1.116029751 _3_ MAC  --- 0 RTS 44 [cbe 3 2 0]
s 1.116039751 _3_ MAC  --- 0 CTS 38 [b84 2 0 0]
r 1.116344352 _2_ MAC  --- 0 CTS 38 [b84 2 0 0]
s 1.116354352 _2_ MAC  --- 1 cbr 328 [13a 3 2 800] ---- [0:0 6:0 29 3]
r 1.118978953 _3_ MAC  --- 1 cbr 276 [13a 3 2 800] ---- [0:0 6:0 29 3]
s 1.118988953 _3_ MAC  --- 0 ACK 38 [0 2 0 0]

r 1.119003953 _3_ AGT  --- 1 cbr 276 [13a 3 2 800] ---- [0:0 6:0 29 3]
```

Parte do log foi removido para se encaixar melhor no tamanho da página

Referências Bibliográficas

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. **Wireless sensor networks: a survey**. *Computer Networks (Elsevier) Journal*, 38(4):393–422, Mar 2002.

[2] Michael Bahr. **Proposed routing for IEEE 802.11s WLAN mesh networks**. In *Proceedings of the 2nd annual international workshop on Wireless internet WICON '06*, page 5, New York, NY, USA, 2006. ACM Press.

[3] Dimitri P. Bertsekas and Robert G. Gallager. **Distributed Asynchronous Bellman-Ford Algorithm**. In *Data Networks*, chapter 5.2.4, pages 325–333. Prentice Hall, Englewood Cliffs, 1987.

[4] Sandeep Bhadra and Afonso Ferreira. **Computing multicast trees in dynamic networks using evolving graphs**. Research Report RR-4531, INRIA, Ago 2002.

[5] Sandeep Bhadra and Afonso Ferreira. **Complexity of Connected Components in Evolving Graphs and the Computation of Multicast Trees in Dynamic Networks**. In *Proceedings of Adhoc-Now'03*, volume 2865 of *Lecture Notes in Computer Science*, pages 259–270. Springer Verlag, Oct 2003.

[6] **Bluetooth Specification**. http://www.bluetooth.org/spec/.

[7] Jean-Yves Le Boudec. **Understanding the simulation of mobility models with Palm calculus**. *Performance Evaluation*, 64(2):126–147, 2007.

[8] Jean-Yves Le Boudec and Milan Vojnovic. **Perfect simulation and stationarity of a class of mobility models**. 4:2743–2754, Mar 2005.

[9] Azzedine Boukerche. **Performance evaluation of routing protocols for ad hoc wireless networks**. *ACM Mobile Network Applications (MONET)*, 9(4):333–342, 2004.

[10] 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 the 4th ACM annual international Conference on Mobile Computing and Networking (MobiCom'98)*, pages 85–97, Dallas, TX, USA, 1998. ACM Press.

[11] Binh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. **Computing Shortest, Fastest, and Foremost Journeys in Dynamic Networks**. *International Journal of Foundations of Computer Science*, 14(2):267–285, Apr 2003.

[12] Binh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. **Evolving graphs and least cost journeys in dynamic networks**. In *Proceedings of Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt'03)*, pages 141–150. INRIA Press, March 2003.

[13] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, and Howard Weiss. **Delay-tolerant networking: an approach to interplanetary Internet**. *IEEE Communications Magazine*, 41(6):128–136, 2003.

[14] Tracy Camp, Jeff Boleng, and Vanessa Davies. **A Survey of Mobility Models for Ad Hoc Network Research**. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[15] Casey Carter, Seung Yi, and Robin Kravets. **ARP Considered Harmful: Manycast Transactions in Ad Hoc Networks**. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 03)*, New Orleans, LA, Mar 2003.

[16] Vint Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Bob Durst, Keith Scott, Eric Travis, and Howard Weiss. **InterPlaNetary Internet (IPN): architectural definition**. Internet draft, IRTF, May 2001.

[17] Chao Chen and Eylem Ekici. **A Routing Protocol for Hierarchical LEO/MEO Satellite IP Networks.** *ACM Wireless Networks (WiNet)*, 11(4):507–521, 2005.

[18] Chee-Yee Chong and S.P. Kumar. **Sensor networks: evolution, opportunities, and challenges**. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[19] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. **Introduction to Algorithms**. The MIT press, Cambridge, MA, USA, 1990.

[20] Scott Corson and Joseph Macker. **Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations**. RFC 2501, IETF, January 1999.

[21] David Culler, Deborah Estrin, , and Mani Srivastava. **Guest Editors' Introduction: Overview of Sensor Networks**. *IEEE Computer*, 37(8):41–49, Aug 2004.

[22] Samir Das, Robert Castaneda, and Jiangtao Yan. **Simulation based performance evaluation of mobile, ad hoc network routing protocols**. In *ACM Mobile Networks and Applications (MONET) Journal*, pages 179–189. Kluwer Academic Publishers, Jul 2000.

[23] Christian de Waal. **BonnMotion – A mobility scenario generation and analysis tool**. http://www.cs.uni-bonn.de/IV/BonnMotion/, Acesso em Jun 2007.

[24] Deborah Estrin, Lewis Girod, Greg Pottie, and Mani Srivastava. **Instrumenting the world with wireless sensor networks**. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*, Salt Lake City, UT, USA, May 2001.

[25] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. **Next Century Challenges: Scalable Coordination in Sensor Networks**. In *Proceedings of the 5th ACM annual international Conference on Mobile Computing and Networking (MobiCom'99)*, pages 263–270, Seattle, WA, USA, August 1999. ACM Press.

[26] Kevin Fall. **A delay-tolerant network architecture for challenged internets**. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for computer communications (SIGCOMM'03)*, pages 27–34, New York, NY, USA, 2003. ACM Press.

[27] Andres Farago and Violet R. Syrotiuk. **MERIT: a scalable approach for protocol assessment**. *ACM Mobile Networks and Applications (MONET) journal*, 8(5):567–577, 2003.

[28] Afonso Ferreira. **Building a Reference Combinatorial Model for Dynamic Networks: Initial Results in Evolving Graphs**. Research Report RR-5041, INRIA, Dec 2003.

[29] Afonso Ferreira. **Building a reference combinatorial model for MANETs**. *IEEE Network*, 18(5):24–29, Set 2004.

[30] Afonso Ferreira, Jérôme Galtier, and Paolo Penna. **Topological design, routing and hand-over in satellite networks**. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, pages 473–493. John Wiley and Sons, New York, NY, USA, 2002.

[31] Afonso Ferreira and Aubin Jarry. **Complexity of Minimum Spanning Tree in Evolving Graphs and the Minimum-Energy Broadcast Routing Problem**. In *Proceedings of Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt'04)*, pages 55–61, Cambridge, UK, Mar 2004.

[32] Afonso Ferreira and Laurent Viennot. **A Note on Models, Algorithms, and Data Structures for Dynamic Communication Networks**. Research Report RR-4403, INRIA, Mar 2002.

[33] Lester R. Ford and Delbert R. Fulkerson. **Constructing Maximal Dynamic Flow from Static Flows**. *The Journal of the Operations Research Society of America*, 6:419–433, 1958.

[34] Olga Goussevskaia, Max Machado, Raquel Mini, and Jose Nogueira Antonio Loureiro, Geraldo Mateus. **Data dissemination based on the energy map**. volume 43, pages 134–143, Jul 2005.

[35] Khaled A. Harras, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. **Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding Schemes in Sparse Mobile Networks**. In *IFIP Netwoking*, Waterloo, Canada, May 2005.

[36] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. **Energy-Efficient Communication Protocol for Wireless Microsensor Networks**. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00)*, volume 8, pages 3005–3014. IEEE Computer Society, Jan 2000.

[37] **IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications**. http://grouper.ieee.org/groups/802/11/main.html, Acesso em Mar 2007.

[38] Free Software Foundation Inc. GNU General Public License. Disponível em: `http://www.gnu.org/licenses/gpl.html`, June 1991. Acesso em: 05/06/2007.

[39] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. **Directed diffusion: a scalable and robust communication paradigm for sensor networks**. In *Proceedings of the 6th ACM annual international Conference on Mobile Computing and Networking (MobiCom'00)*, pages 56–67, Boston, MA, USA, Aug 2000. ACM Press.

[40] Philippe Jacquet, Paul Muhlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. **Optimized Link State Routing Protocol**. In *Proceedings of 5th IEEE INMIC'01*, pages 62–68, Lahore, Pakistan, Dec 2001.

[41] Sushant Jain, Kevin Fall, and Rabin Patra. **Routing in a Delay Tolerant Network**. *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for computer communications (SIGCOMM'04)*, 34(4):145–158, October 2004.

[42] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. **Towards realistic mobility models for mobile ad hoc networks**. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 217–229, New York, NY, USA, 2003. ACM Press.

[43] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. **Scenario-based performance analysis of routing protocols for mobile ad-hoc networks**. In *Proceedings of the 5th annual ACM international conference on Mobile computing and networking (MobiCom'99)*, pages 195–206, Seattle, WA, USA, 1999. ACM Press.

[44] David B. Johnson and David A. Maltz. **Dynamic Source Routing in Ad Hoc Wireless Networks**. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[45] Evan P. C. Jones, Lily Li, and Paul A. S. Ward. **Practical routing in delay-tolerant networks**. In *Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN'05)*, pages 237–243, New York, NY, USA, 2005. ACM Press.

[46] Joe M. Kahn, Randy H. Katz, and Kris S. J. Pister. **Next Century Challenges: Mobile Networking for "Smart Dust"**. In *Proceedings of the 5th ACM annual international conference on Mobile Computing and networking (MobiCom'99)*, pages 271–278, Seattle, WA, USA, Aug 1999. ACM Press.

[47] Vijay Kakadia and Wei Ye. **Energy Model Update in NS-2**. http://www.isi.edu/ilense/software/smac/ns2_energy.html, Acesso em Dec 2005.

[48] Balázs Kotnyek. **An annotated overview of dynamic network flows**. Research Report RR-4936, INRIA, Set 2003.

[49] Bhaskar Krishnamachari. **Networking Wireless Sensors**. Cambridge University Press, New York, NY, USA, 2005.

[50] Ekkehard Köhler, Katharina Langkau, and Martin Skutella. **Time-Expanded Graphs for Flow-Dependent Transit Times**. In *Proceedings of the 10th annual European Symposium on Algorithms (ESA'02)*, pages 599–611, London, UK, 2002. Springer-Verlag.

[51] Ekkehard Köhler and Martin Skutella. **Flows over time with load-dependent transit times**. In *Proceedings of the 13th annual ACM-SIAM symposium on Discrete algorithms (SODA'02)*, pages 174–183, San Francisco, CA, USA, 2002. Society for Industrial and Applied Mathematics.

[52] Daniel Lang. **A comprehensive overview about selected Ad Hoc Networking Routing Protocols**. Technical report, Technische Universitaet Munchen, Department of Computer Science, Mar 2003.

[53] Weifa Liang and Yang Yuansheng. **Maximizing Battery Life Routing in Wireless Ad Hoc Networks**. 09:90295, 2004.

[54] Antonio A.F. Loureiro, Jose M. Nogueira, Linnyer Ruiz, Eduardo Nakamura, Carlos Serodio, and Raquel Mini. **Redes de Sensores sem Fio**. Chapter 4 of short-courses book in the 21th Brazilian Symposium on Computer Networks (SBRC'03), 2003.

[55] Marina Mahesh. **AODV code for CMU Wireless and Mobility Extensions to NS2**. http://www.cs.sunysb.edu/ mahesh/aodv/, Acesso em Nov 2005.

[56] B. S. Manoj, K. Jayanth Kumar, Christo Frank, and C. Siva Ram Murthy. **On the Use of Multiple Hops in Next Generation Wireless Systems.** *ACM Wireless Networks (WiNet)*, 12(2):199–221, 2006.

[57] Marc Greis. **NS2 Tutorial**. http://www.isi.edu/nsnam/ns/tutorial/index.html, Acesso em Jun 2007.

[58] Julian Monteiro, Alfredo Goldman, and Afonso Ferreira. **Performance Evaluation of Dynamic Networks using an Evolving Graph Combinatorial Model**. In *Proceedings of the 2nd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'06)*, pages 173–180, Montreal, CA, Jun 2006. Best Student Paper Award.

[59] Julian Monteiro, Alfredo Goldman, and Afonso Ferreira. **On the Evaluation of Shortest Journeys in Dynamic Networks**. In *Proceedings of the 6th IEEE Intl. Symposium on Networking Computing and Applications (NCA'07)*, Cambridge, USA, Jul 2007. to appear.

[60] Julian Monteiro, Alfredo Goldman, and Afonso Ferreira. **Using Evolving Graphs Foremost Journey to Evaluate Ad-Hoc Routing Protocols**. In *Proceedings of 25th Brazilian Symposium on Computer Networks (SBRC'07)*, pages 17–30, Belém, Brazil, Jun 2007.

[61] NS2. **The Network Simulator – NS2**. http://nsnam.isi.edu/nsnam/, Acesso em Mar 2007.

[62] NS3. **The Network Simulator – NS3**. http://www.nsnam.org/, Acesso em Mar 2007.

[63] Carina T. Oliveira and Otto Carlos M.B. Duarte. **Uma Análise da Probabilidade de Entrega de Mensagens em Redes Tolerantes a Atrasos e Desconexões** . In *Proceedings of 25th Brazilian Symposium on Computer Networks (SBRC'07)*, pages 293–305, Belém, Brazil, Jun 2007.

[64] Carina T. Oliveira, Marcelo D.D. Moreira, Marcelo G. Rubinstein, Luis Henrique M. K. Costa, and Otto Carlos M. B. Duarte. **Redes Tolerantes a Atrasos e Desconexões**. Chapter 5 of short-courses book in the 25th Brazilian Symposium on Computer Networks (SBRC'07), Belém, PA, Brazil, 2007.

[65] Charles Perkins and Pravin Bhagwat. **Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers**. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications (ACM SIGCOMM'94)*, pages 234–244, Sep 1994.

[66] Charles E. Perkins and Elizabeth M. Royer. **Ad-hoc On-Demand Distance Vector Routing**. In *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, Feb 1999.

[67] Charles E. Perkins, Elizabeth M. Royer, Samir Ranjan Das, and Mahesh K. Marina. **Performance Comparison of two on-demand routing protocols for ad hoc networks**. In *IEEE Personal Communications*, volume 8, pages 16–28. IEEE Communications Society, feb 2001.

[68] Gregory J. Pottie and William J. Kaiser. **Wireless Integrated Network Sensors**. *Communications of the ACM*, 43(5):51–58, May 2000.

[69] Rice University Monarch Project. **The CMU Monarch Wireless and Mobility Extensions to NS2**. http://www.monarch.cs.rice.edu/, Acesso em Mar 2007.

[70] Francisco J. Ros. **UM-OLSR implementation (version 0.8.8) for NS2**. http://masimum.dif.um.es/?Software:UM-OLSR, Acesso em Mar 2007.

[71] Elizabeth M. Royer and C-K Toh. **A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks**. *IEEE Personal Communications Magazine*, pages 46–55, Apr 1999.

[72] Isabela G. Siqueira, Linnyer Beatrys Ruiz, Antonio A. F. Loureiro, and Jose Marcos Nogueira. **Coverage area management for wireless sensor networks**. *International Journal of Network Management*, 17(1):17–31, 2007.

[73] Ivan Stojmenovic, Amiya Nayak, and Johnson Kuruvila. **Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer**. *IEEE Communications Magazine (Ad Hoc and Sensor Networks Series)*, 43(3):101–106, March 2005.

[74] Ivan Stojmenovic(ed.). **Handbook of Wireless Networks and Mobile Computing**. John Wiley and Sons, Feb 2002.

[75] Ivan Stojmenovic(ed.). **Handbook of Sensor Networks: Algorithms and Architectures**. John Wiley and Sons, Oct 2005.

[76] VINT project. **Virtual InterNetwork Testbed**. http://www.isi.edu/nsnam/vint/, Acesso em Jun 2007.

[77] Markus Werner and Gérard Maral. **Traffic flows and dynamic routing in LEO intersatellite link networks**. In *International Mobile Satellite Conference (IMSC'97)*, pages 283–288, Jun 1997.

[78] Jie Wu. **Handbook On Theoretical And Algorithmic Aspects Of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks**. Auerbach Publications, Boston, MA, USA, 2005.

[79] Jungkeun Yoon, Mingyan Liu, and Brian Noble. **Random Waypoint Considered Harmful**. In *The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, volume 2, pages 1312–1321, 2003.

[80] Hongqiang Zhai and Yuguang Fang. **Distributed Flow Control and Medium Access in Multihop Ad Hoc Networks**. *IEEE Transactions on Mobile Computing*, 05(11):1503–1514, 2006.

[81] Zhensheng Zhang. **Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges**. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.