UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROPOSTA DE UM MECANISMO DE AVALIAÇÃO DA CONFIANÇA DE ROTAS EM AMBIENTES DE COMPUTAÇÃO UBÍQUA

LUIZ FERNANDO SIROTHEAU SERIQUE JUNIOR

ORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JÚNIOR

TESE DE DOUTORADO EM
ENGENHARIA ELÉTRICA

BRASÍLIA/DF: ABRIL/2013.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROPOSTA DE UM MECANISMO DE AVALIAÇÃO DA CONFIANÇA DE ROTAS EM AMBIENTES DE COMPUTAÇÃO UBÍQUA

LUIZ FERNANDO SIROTHEAU SERIQUE JUNIOR

TESE DE DOUTORADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM ENGENHARIA ELÉTRICA.

APROVADA POR:

Prof. Rafael Timóteo de Sousa Júnior, Dr. (ENE-UnB)
(Orientador)

Prof. Flávio Elias Gomes de Deus, Dr. (ENE-UnB)
(Examinador Interno)

Prof. Georges Daniel Amvame Nze, Dr. (FGA-UnB)
(Examinador Interno)

Prof. Robson de Oliveira Albuquerque, Dr. (ABIN)
(Examinador Externo)

Prof. José Eduardo Malta de Sá Brandão, Dr. (IPEA)
(Examinador Externo)

BRASÍLIA/DF, 8 DE ABRIL DE 2013.

FICHA CATALOGRÁFICA

SERIQUE JUNIOR, LUIZ FERNANDO SIROTHEAU
Proposta de um Mecanismo de Avaliação da Confiança de Rotas em Ambientes de Computação Ubíqua. [Distrito Federal] 2013.
xvii, 148 p., 297 mm (ENE/FT/UnB, Doutor, Engenharia Elétrica, 2013).
Tese de Doutorado - Universidade de Brasília.
Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Redes Ad Hoc	2. Computação Ubíqua
3. Roteamento de Redes	4. Confiança Computacional
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

Serique Junior, L. F. S. S. (2013). Proposta de um Mecanismo de Avaliação da Confiança de Rotas em Ambientes de Computação Ubíqua. Tese de Doutorado em Engenharia Elétrica, Publicação PPGENE.TD - 074/2013, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 148p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Luiz Fernando Sirotheau Serique Junior.

TÍTULO DA TESE DE DOUTORADO: Proposta de um Mecanismo de Avaliação da Confiança de Rotas em Ambientes de Computação Ubíqua.

GRAU / ANO: Doutor / 2013

Luiz Fernando Sirotheau Serique Junior
SHA Quadra 4 Conjunto 5 Chácara 37 Lote 18, Setor Habitacional Arniqueiras
71.995-245 Brasília - DF - Brasil.

Dedico esta obra à Fernanda e ao Lucas.

AGRADECIMENTOS

Agradeço sobretudo à Deus por ter me dado a inspiração necessária, saúde, paciência e perseverança para que fosse possível concluir essa pesquisa, e, ainda, por ter proporcionado o convívio com pessoas especiais em minha vida.

Agradeço ao professor Rafael Timóteo de Sousa Júnior pela oportunidade, pela confiança depositada e pelos conselhos que foram indispensáveis. Ao competente time de professores do ENE/UnB que contribuiu para minha formação acadêmica desde os tempos da graduação. Ao amigo José Carlos, do TJDFT, por todo incentivo que foi essencial para iniciar esta jornada.

Em especial, agradeço à minha esposa, Fernanda, e ao meu filho, Lucas, pelo companheirismo, pela compreensão e pelo carinho, mesmo nos momentos mais difíceis. À minha mãe, Maria de Lourdes, e ao meu pai, Luiz Fernando (*in memorian*), por todo cuidado, dedicação e amor desde meu nascimento. Aos meus queridos irmãos, Marcelo, Ester e Ana, pela admiração, e aos amigos Michael e Vinicius pela sincera amizade. Obrigado por compreenderem a minha ausência em tantas ocasiões!

RESUMO

PROPOSTA DE UM MECANISMO DE AVALIAÇÃO DA CONFIANÇA DE ROTAS EM AMBIENTES DE COMPUTAÇÃO UBÍQUA

Autor: Luiz Fernando Sirotheau Serique Junior

Orientador: Rafael Timóteo de Sousa Júnior

Programa de Pós-graduação em Engenharia Elétrica

Brasília, Abril de 2013

Os ambientes de computação ubíqua oferecem grandes desafios às tecnologias de redes sem fio. Devido à diversidade de dispositivos e ao dinamismo da topologia desses ambientes, a rede está sujeita a diversos problemas, como a falta de cooperação e inoperância dos nós, o rompimento de enlaces e as limitações de energia e de largura de banda. Logo, os protocolos de roteamento devem estar em constante adaptação e, ainda, devem empregar métricas mais sofisticadas para escolha de rotas bem-sucedidas. Esta tese propõe um mecanismo de avaliação da confiança de rotas que visa aprimorar as decisões de roteamento dos protocolos, melhorando, assim, o desempenho da rede. Para isso, são empregadas métricas multidimensionais de roteamento que envolvem parâmetros de mobilidade, atividade, cooperação e distância das rotas. É usada uma abordagem de aprendizagem de máquina indutiva para assimilar os padrões das rotas bem-sucedidas e gerar um conjunto de regras de decisão envolvendo as métricas. As regras são renovadas periodicamente para garantir a adaptação do mecanismo, caso o comportamento da rede se altere. A validação do mecanismo foi feita com o protocolo Dynamic Source Routing (DSR) por meio de simulador de redes. Foram simulados cenários com nós egoístas, nós em modo *sleep* e bastante mudança topológica com o objetivo de causar anomalias na rede. Os resultados demonstraram que o mecanismo se adaptou com o passar do tempo, criando regras mais rigorosas em ambientes hostis, e melhorou a taxa de pacotes transmitidos com sucesso na rede. Sendo assim, o seu emprego pode beneficiar o processo de roteamento, proporcionando maior desempenho aos ambientes de computação ubíqua.

ABSTRACT

PROPOSAL FOR A RELIABILITY EVALUATION MECHANISM OF ROUTING IN UBIQUITOUS COMPUTATIONAL ENVIRONMENTS

Author: Luiz Fernando Sirotheau Serique Junior

Supervisor: Rafael Timóteo de Sousa Júnior

Programa de Pós-graduação em Engenharia Elétrica

Brasília, April of 2013

Ubiquitous computational environments offer great challenges to the wireless network technology. Owing to the diversity of devices and the dynamism of the topology of these environments, the network is subject to diverse problems, such as the lack of cooperation and inoperability of the nodes, broken links, energy and wideband constraints, etc. Ergo, the protocols for routing must be in constant adaptation, and must even employ more sophisticated metrics in choosing successful paths. This thesis proposes a reliability evaluation mechanism that aims to perfect the routing decisions of the protocols, thereby improving network performance. For this purpose, multidimensional routing metrics are employed, which involve parameters of mobility, activity, cooperation and routing distance. An inductive machine learning approach is used to assimilate the high performance routing patterns and to generate a set of decision making rules involving the metrics. The rules are periodically updated to guarantee the adaptation of the mechanism in case the network alters its behavior. The validation of the mechanism was made with the Dynamic Source Routing (DSR) protocol, using a network simulator. Scenarios were simulated with selfish nodes, sleep nodes and many topological changes with the objective of causing anomalies on the network. The result demonstrates the mechanism adapted itself over time, creating more rigid rules in hostile environments, and improved the rate of packets transmitted successfully on the network. Thus, using the mechanism may benefit the routing process, providing higher performance in ubiquitous computational environments.

SUMÁRIO

1 INTRODUÇÃO **1**

1.1 DEFINIÇÃO DO PROBLEMA 3

1.2 OBJETIVOS 5

1.3 VISÃO GERAL DO MECANISMO 6

1.4 CONTRIBUIÇÕES 8

1.5 PUBLICAÇÕES 9

1.6 TRABALHOS RELACIONADOS 9

1.7 ORGANIZAÇÃO DA TESE 11

2 REVISÃO DA LITERATURA **12**

2.1 COMPUTAÇÃO UBÍQUA 12

2.1.1 Histórico da Computação Ubíqua 14

2.1.2 Características da Computação Ubíqua 17

2.1.3 Desafios da Computação Ubíqua 20

2.2 ROTEAMENTO EM REDES AD HOC 21

2.2.1 Breve Histórico das Redes Sem Fio 24

2.2.2 Padrão IEEE 802.11 26

2.2.3 Protocolos de Roteamento em Redes *Ad Hoc* 28

2.2.4 Classificação dos Protocolos de Roteamento de Redes *Ad Hoc* . 31

2.2.4.1 Classificação pelo Modo de Delegação de Tarefas . . . 32

2.2.4.2 Classificação pela Métrica de Roteamento 33

2.2.4.3 Classificação pelo Modo de Avaliação da Topologia, Destinação e Localização 34

2.2.4.4 Classificação pelo Tipo de *Multicast* 34

2.2.4.5 Mapeamento das Classificações dos Principais Protocolos de Redes *Ad Hoc* 35

2.3 MÉTRICAS DE ROTEAMENTO EM REDES AD HOC 35

2.3.1 Objetivos da Otimização do Roteamento 37

2.3.2 Métricas de Enlaces *versus* Métricas Rotas 39

2.3.3 Métodos para o Cálculo de Métricas 40

2.3.4 Métricas de Roteamento em Redes Sem Fio 41
2.3.4.1 Métricas baseadas na Topologia 41
2.3.4.2 Métricas baseadas na Intensidade do Sinal 42
2.3.4.3 Métricas baseadas em Provas Ativas 44
2.3.4.4 Métricas baseadas na Mobilidade 46
2.4 PROTOCOLO DYNAMIC SOURCE ROUTING 47
2.4.1 Premissas do Protocolo Dynamic Source Routing 48
2.4.2 Mecanismos de Descoberta e Manutenção de Rotas 49
2.4.2.1 Funcionamento do Mecanismo de Descoberta de Rotas 50
2.4.2.2 Funcionamento da Manutenção de Rotas 52
2.4.3 Recursos Adicionais do Dynamic Source Routing 53
2.4.3.1 *Caching* das Informações de Roteamento 53
2.4.3.2 Envio de Mensagens *Route Reply* por Nós Intermediários 54
2.4.3.3 Prevenção contra Tempestades de *Route Reply* 54
2.4.3.4 Limitador do Número de Saltos em Mensagens *Route Request* 55
2.4.3.5 Resgate de Pacotes em Rotas Quebradas 56
2.4.3.6 Redução Automática de Rotas 57
2.4.3.7 Aumento da Propagação de Mensagens *Route Error* . 57
2.4.3.8 Armazenamento de Informações Negativas 57
2.4.3.9 Extensão de Estado de Fluxo 58
2.4.4 Formato dos Cabeçalhos do Dynamic Source Routing 58
2.4.4.1 Estrutura do Cabeçalho *DSR Options* 59
2.4.4.2 Estrutura do Cabeçalho *Route Request Option* 59
2.4.4.3 Estrutura do Cabeçalho *Route Reply Option* 60
2.4.4.4 Estrutura do Cabeçalho *Source Route Option* 61
2.4.4.5 Estrutura do Cabeçalho *Route Error Option* 62
2.4.4.6 Estrutura do Cabeçalho *Acknowledgement Option* e *Acknowledgement Request Option* 62
2.5 SEGURANÇA E CONFIANÇA EM REDES AD HOC 63
2.5.1 Confiança em Ambientes de Computação Ubíqua 65
2.5.2 Ataques em Redes *Ad Hoc* 66
2.5.3 Mecanismos de Confiança para Redes *Ad Hoc* 68

2.5.3.1 *Watchdog* e *Pathrater* 69
2.5.3.2 *CONFIDANT: Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks* 69
2.5.3.3 *CORE: A Collaborative Reputation Mechanism for node cooperation in Ad hoc Networks* 71
2.5.3.4 *SORI: A Secure and Objective Reputation-based. Incentive Scheme for Ad-hoc Networks* 72
2.5.3.5 *OCEAN: Observation-based Cooperation Enforcement in Ad hoc Networks* 73
2.6 APRENDIZAGEM INDUTIVA E ÁRVORES DE DECISÃO 74
2.6.1 Árvores de Decisão 75
2.6.2 Entropia da Informação 78
2.6.3 Ganho de Informação 79
2.6.4 Algoritmo ID3 80
2.6.5 Algoritmo C4.5 80

3 PROPOSTA DO MECANISMO DE AVALIAÇÃO DE CONFIANÇA DE ROTAS 83
3.1 DEFINIÇÕES E TERMINOLOGIAS ADOTADAS 83
3.2 PREMISSAS DO MECANISMO 85
3.3 MÉTRICAS DE CONFIANÇA DA ROTA 86
3.3.1 Atividade da Rota - Act_{path} 86
3.3.2 Cooperação da Rota - Cop_{path} 88
3.3.3 Mobilidade da Rota - Mob_{path} 89
3.3.4 Distância da Rota - Dst_{path} 90
3.4 ARQUITETURA DO METRUBI 90
3.4.1 Registrador de Confiança 91
3.4.2 Monitor 92
3.4.3 Indutor DT 94
3.4.4 *Cache* de Rotas 96
3.4.5 Avaliador de Rotas 96
3.5 FUNCIONAMENTO DO METRUBI 97
3.6 COMPARAÇÃO COM OUTROS ESQUEMAS DE CONFIANÇA EM REDES *AD HOC* 98

4 SIMULAÇÕES E RESULTADOS EXPERIMENTAIS 101

4.1 FERRAMENTAS E RECURSOS UTILIZADOS 102

4.1.1 Protocolo Dynamic Source Routing 102

4.1.2 Simulador de Redes Glomosim 103

4.2 EXPERIMENTO A: ESTUDO DOS EFEITOS DA DENSIDADE, DO DINAMISMO, DO EGOÍSMO E DO MODO *SLEEP* NO DESEMPENHO DA REDE 105

4.3 EXPERIMENTO B: VALIDAÇÃO DAS MÉTRICAS DE CONFIANÇA 109

4.4 EXPERIMENTO C: COMPARAÇÃO DO DESEMPENHO COM O MECANISMO DE AVALIAÇÃO DA CONFIANÇA ATIVADO EM AMBIENTE NÃO COOPERATIVO COM MOBILIDADE ALEATÓRIA . 113

4.5 EXPERIMENTO D: COMPARAÇÃO DO DESEMPENHO COM O MECANISMO DE AVALIAÇÃO DA CONFIANÇA ATIVADO EM AMBIENTE NÃO COOPERATIVO COM MOBILIDADE CONTROLADA 117

5 CONCLUSÃO E TRABALHOS FUTUROS 121

REFERÊNCIAS BIBLIOGRÁFICAS 124

APÊNDICES 139

A Script para Automação das Simulações 140

B Rotina de Cálculo das Métricas de Confiança 142

C Rotina de Ordenação do Avaliador de Rotas 144

D Mapeamento das Alterações no Glomosim 146

LISTA DE TABELAS

2.1 Comparação da computação ubíqua com outros estágios da computação 17
2.2 Características da computação ubíqua 18
2.3 Desafios da computação ubíqua 22
2.4 Evolução dos padrões 802.11 e suas características 26
2.5 Mapeamento dos protocolos de acordo com os métodos de classificação 36
2.6 Principais técnicas de ataque em redes *ad hoc* 67
3.1 Tabela de Métricas de Nós 92
3.2 Tabela de Métricas de Rotas 92
4.1 Parâmetros da simulação da Experimento A 105
4.2 Parâmetros da simulação da Experimento B 110
4.3 Parâmetros da simulação da Experimento C 113
4.4 Parâmetros da Simulação do Experimento D 118

LISTA DE FIGURAS

1.1 Escolha da melhor rota em uma aplicação de computação ubíqua . . . 4
1.2 Mapeamento da área de concentração da pesquisa 5
1.3 Arquitetura do METRUbi 8
2.1 Relação computação pervasiva, ubíqua e móvel 13
2.2 Realidade virtual *versus* computação ubíqua 15
2.3 Transição dos paradigmas da computação 16
2.4 Modos de operação do IEEE 802.11 27
2.5 Probabilidade de entrega de pacotes *versus* médias de S/N 43
2.6 Estimativa da métrica ETX do nó A 46
2.7 Exemplo de descoberta de rotas 51
2.8 Exemplo de manutenção de rotas 53
2.9 Limitação do *caching* de informações de roteamento em enlaces unidirecionais. 54
2.10 Possível duplicação de nós em caminhos concatenados no processo de criação RREP. 54
2.11 Exemplo de tempestade de mensagens *Route Reply*. 55
2.12 Exemplo de redução automática de rota. 57
2.13 Cabeçalho DSR Options 59
2.14 Estrutura do cabeçalho *Route Request Option* 60
2.15 Estrutura do cabeçalho *Route Reply Option* 61
2.16 Estrutura do cabeçalho *Source Route Option* 61
2.17 Estrutura do cabeçalho *Route Error Option* 62
2.18 Estrutura do cabeçalho *Acknowledgement Request Option* 63
2.19 Estrutra do cabeçalho *Acknowledgement Option* 63
2.20 Relacionamento entre confiança, reputação e reciprocidade. 65
2.21 Aproximações de $f(x)$ a partir de pares $(x, f(x))$ 75
2.22 Exemplo de árvore de decisão 76
2.23 Pseudocódigo do algoritmo ID3 80
2.24 Conjunto de exemplos de treinamento para árvores de decisão 81
2.25 Árvore de decisão gerada com o C4.5 81

2.26 Regras *if-then* para uma árvore de decisão 82
3.1 Arquitetura do METRUbi 91
3.2 Esquema de filtragem de informações do DSR pelo componente Monitor 93
3.3 Pseudocódigo do componente Indutor DT 95
3.4 Estruturas de dados do *cache* de rotas do protocolo DSR 96
3.5 Entrada de *cache* modificada para o Avaliador de Rotas 97
4.1 Efeito da densidade na taxa de entrega de pacotes com diferentes proporções de nós egoístas 106
4.2 Efeito da densidade na taxa de entrega de pacotes com diferentes proporções de nós em modo *sleep* 107
4.3 Comparação dos efeitos de nós egoístas e nós em modo *sleep* 108
4.4 Efeito da mobilidade na taxa de entrega de pacotes com diferentes proporções de nós egoístas 109
4.5 Análise da dispersão da atividade e cooperação das rotas 111
4.6 Análise da dispersão da mobilidade e cooperação das rotas 112
4.7 Análise da dispersão do número de saltos e cooperação das rotas 112
4.8 Comparação da taxa de entrega de pacotes do DSR padrão com o DSR auxiliado pelo METRUbi com mobilidade aleatória 114
4.9 Ganho na taxa de entrega proporcionado pelo METRUbi com mobilidade aleatória 115
4.10 Regras obtidas nas simulação sem nós egoístas 115
4.11 Regras obtidas nas simulação com 10 nós egoístas 116
4.12 Regras obtidas nas simulação com 20 nós egoístas 116
4.13 Regras obtidas nas simulação com 30 nós egoístas 117
4.14 Regras obtidas nas simulação com 40 e 50 nós egoístas 117
4.15 Comparação da taxa de entrega de pacotes do DSR padrão com o DSR auxiliado pelo METRUbi com mobilidade controlada 119
4.16 Ganho na taxa de entrega proporcionado pelo METRUbi com mobilidade controlada 120

LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

ABR: Associativity Based Routing.

AMRIS: Ad hoc Multicast Routing utilizing Increasing id numberS.

AMROUTE: Ad Hoc Multicast Routing.

AODV: Ad hoc On-Demand Distance Vector.

CBRP: Cluster Based Routing Protocol.

CEDAR: Core-Extraction Distributed Ad Hoc Routing.

CGSR: Clusterhead Gateway Switch Routing.

CONFIDANT: Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks.

CORE: A Collaborative Reputation Mechanism for node cooperation in Ad hoc Networks.

CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance.

CSMA/CD: Carrier Sense Multiple Access with Collision Detection.

DREAM: Distance Routing Effect Algorithm for Mobility.

DSDV: Destination Sequence Distance Vector.

DSR: Dynamic Source Routing.

DVMRP: Distance Vector Multicast Routing Protocol.

ETSI: European Telecommunication Standards Institute.

ETX: Expected Transmission Count.

FSR: Fisheye State Routing.

GLS: Grid Location Service.

HARP: Hybrid Ad Hoc Routing Protocol.

HSR: Hierarquical State Routing.

IANA: Internet Assigned Numbers Authority.

IBSS: Independent Basic Service Set.

IEEE: Institute of Electrical and Electronic Engineers.

IETF: Internet Engineering Task Force.

IP: Internet Protocol.

IrDA: Infrared Data Association.

LANMAR: Landmark Ad Hoc Routing.

LAR: Location Aided Routing.

MACA: Multiple Access with Collision Detection.

MACAW: Multiple Access with Collision Avoidance for Wireless.

MANET: Mobile Ad hoc Network.

MAODV: Multicast Ad Hoc On-Demand Distance Vector.

MAOV: Multicast Ad Hoc On-demand Distance Vector.

MIMO: Multiple Input, Multiple Output.

MOSPF: Multicast Open Shortest Path First.

MPR: MultiPoint Relays.

MTM: Medium Time Metric.

OCEAN: Observation-based Cooperation Enforcement in Ad hoc Networks.

ODMRP: On-Demand Multicast Routing Protocol.

OLSR: Optimized Link State Routing Protocol.

PIM: Protocol-Independent Multicast.

PRNET: Packet Radio Networks.

QoS: Quality of Service.

RERR: Route Error.

RREP: Route Reply.

RREQ: Route Request.

RTT: Per-hop Round Trip Time.

SORI: A Secure and Objective Reputation-based. Incentive Scheme for Ad-hoc Networks.

SSR: Signal Stability-based Adaptive Routing.

TCP: Transmission Control Procotol.

TORA: Temporally Ordered Routing Algorithm.

UBICOMP: Ubiquitous Computing.

UDP: User Datagram Protocol.

WCETT: Weighted Cumulative Expected Transmission Time.

WLAN: Wireless Local Area Network.

WPAN: Wireless Personal Area Network.

WRP: Wireless Routing Protocol.

WSP: Wireless Routing Protocol.

ZHLS: Zone-based Hierarquical Link State.

ZRP: Zone Routing Protocol.

1 INTRODUÇÃO

Os ambientes de computação ubíqua representam um novo paradigma da Computação no qual dispositivos serão integrados aos objetos que nos cercam para fornecer serviços em qualquer lugar e a qualquer tempo (WEISER, 1996). Com a computação ubíqua as interações com os computadores se tornarão mais naturais e intuitivas, substituindo, por exemplo, teclados e *mouses* por comandos baseados em voz e gestos (HARRISON et al., 1998).

As aplicações serão mais personalizadas e se transportarão por onde o usuário passar. Como resultado, a Computação ficará cada vez mais embarcada e onipresente, suportando melhor as nossas atividades cotidianas. Mark Weiser, considerado o pai da computação ubíqua, afirma em (WEISER, 1994) que o computador ideal é aquele "invisível"perante os usuários, mas focado em suas tarefas.

Gradualmente algumas dessas facilidades estão sendo inseridas em nosso dia a dia, distanciando a computação ubíqua de uma visão meramente utópica. Hoje contamos com uma infinidade de dispositivos portáteis, tais como *smartphones*, PDAs, *pinpads*, *tablets*, e outros com capacidade de comunicação em rede, seja por conexão direta, por células ou por comutação com dispositivos fixos. Novos padrões de redes sem fio surgiram para garantir a interoperabilidade entre a gama de dispositivos, tais como o IEEE 802.11 (MAN; SOCIETY, 2007), o Bluetooth (IEEE WPAN TG1, 2002) e o IrDA (ASSOCIATION, 2012).

Houve um grande aprimoramento das interfaces de comando, principalmente em jogos eletrônicos e simuladores de realidade virtual. A automação predial já está presente há bastante tempo nos empreendimentos inteligentes apoiada por sensores de presença, temperatura, umidade, etc. A Internet, além da sua função de *backbone* global, se transformou em uma grande plataforma de aplicações móveis.

Contudo, os ambientes de computação ubíqua possuem características desafiadoras, principalmente para a confiança na comunicação, visto que são compostos por vários dispositivos, estáticos ou móveis, que se interligam espontaneamente, formando redes altamente dinâmicas (YAU; KARIM, 2004). Essas características interferem no

desempenho da comunicação, pois favorecem a ocorrência de quebras de enlace, particionamentos da rede, retransmissões e perdas de pacotes (GERLA et al., 2005). Devido à baixa convergência da rede e do grande número de dispositivos, pode ocorrer a sobrecarga de mensagens de controle topológico, degradando a largura de banda disponível para as aplicações. Consequentemente, torna-se difícil garantir o sucesso na transmissão de informações.

Nos ambientes de computação ubíqua a comunicação é vital e deve ser garantida pela interoperabilidade de diferentes tecnologias de redes. Devido à infinidade e mobilidade dos dispositivos, as tecnologias de redes sem fio são as mais recomendadas, especialmente as que envolvem redes *ad hoc*. As redes *ad hoc* são redes sem fio autônomas, auto-organizadas e esporádicas, que não requerem uma infraestrutura prévia e tampouco uma entidade de controle central (BASAGNI et al., 2004). Logo, os dispositivos devem cooperar na organização e no roteamento da rede (JOSHI et al., 2008).

Porém alguns nós podem deixar de cooperar em virtude de falhas, sobrecarga, egoísmo ou intenção maliciosa, sendo estes denominados na literatura como nós mal comportados (MARTI et al., 2000). Os nós egoístas visam economizar energia por meio do descarte de pacotes e da desativação das interfaces de rede (modo *sleep*), evitando processamento extra (FEENEY; NILSSON, 2001).

Adnane et al. (2013) afirmam que os mecanismos de descoberta e manutenção de rotas intoduzem problemas de segurança específicos dos protocolos de roteamento e as soluções para assegurar tais protocolos é o uso de algumas unidades de controle centralizado ou terceiros confiáveis que acabam limitando a característica da auto-organização das redes *ad hoc*.

A natureza de *broadcast* das comunicações *ad hoc* também desperta o interesse de atacantes, seja para interceptar pacotes ou para interferir no fluxo da rede. O aprimoramento da tecnologia de redes *ad hoc* é vital para as aplicações de computação ubíqua como afirmado em (SUN, 2001).

Os protocolos de roteamento das redes *ad hoc* têm sido empregados em diversas pesquisas de computação ubíqua, pois foram projetados para atuar em ambientes distribuídos, dinâmicos e limitados em recursos (CORSON, 1999). Uma das funções desses protocolos é a descoberta e manutenção de rotas, possibilitando a comunicação em múltiplos saltos (JOSHI et al., 2008). Mas, para atender plenamente os ambientes de computação

ubíqua, os protocolos devem ser melhorados, principalmente no aspecto da confiança no roteamento. Esse tema tem despertado o interesse de vários pesquisadores.

A confiança é reconhecida como um importante aspecto da tomada de decisão em aplicações auto-organizadas e distribuídas (DE SOUSA JR et al., 2009). Em (MUI; MOHTASHEMI, 2001) a confiança é definida como a expectativa subjetiva que um agente tem sobre o comportamento futuro de outro agente com base no histórico de suas interações. No contexto do roteamento de pacotes em ambientes de computação ubíqua, a confiança pode ser associada ao grau de certeza de que os nós vizinhos encaminhem corretamente os pacotes até o destino com base na observação do histórico de suas ações. Para garantir a confiança no roteamento, os protocolos devem escolher rotas mais estáveis e livres de nós mal comportados, evitando, assim, descartes e retransmissões de pacotes.

1.1 DEFINIÇÃO DO PROBLEMA

Os ambientes de computação ubíqua ampliam os problemas de confiança já existentes nas redes *ad hoc* devido ao maior dinamismo e densidade da rede (SUN, 2001). A confiança no roteamento está associado ao grau de certeza de que os pacotes sejam encaminhados corretamente até o seu destino. O sucesso no roteamento depende de uma série de variáveis, tais como a cooperação dos nós, o dinamismo da rede, as condições locais (ruídos, interferências, atenuações, etc.), as reservas de energia dos dispositivos, dentre outras. Contudo, a maioria dos protocolos projetados para as redes *ad hoc* consideram apenas o número de saltos e o estado dos enlaces, ou seja, caso exista mais de um caminho ativo até o destino, escolhem sempre o mais curto, sem avaliar se os nós intermediários são bons candidatos para o repasse de pacotes.

Para aumentar a confiança no roteamento, os protocolos devem aprimorar suas decisões sobre as rotas por meio da observação de mais variáveis, evitando, assim, que os pacotes passem por caminhos tendentes a falhas. Essa tarefa é complexa em ambientes de computação ubíqua, pois os protocolos devem lidar com uma grande quantidade de informações incompletas e voláteis, o que exige constante adaptação do algoritmo.

Para ilustrar esse problema, a Figura 1.1 exibe uma aplicação de computação ubíqua. O usuário pode contar com diversos dispositivos espalhados pelo ambiente para estabelecer a comunicação com seu par de forma inteiramente *ad hoc*. Alguns problemas de comunicação podem ser observados nesse cenário: (a) o nó 2 está intermitente devido

ao baixo nível de energia de sua bateria; (b) o nó 6 está em uma área com restrições na propagação de rádio, ocasionando várias perdas de pacotes; (c) o nó 5, que é um dispositivo veicular, está em alta velocidade, ficando disponível na rede por um curto período de tempo. O protocolo de roteamento deve ser capaz de perceber esses eventos e escolher a rota mais confiável. Neste exemplo a rota $1-3-4-7-8$ ofereceria melhores condições para a transmissão dos pacotes entre os nós 1 e 8, mesmo sendo o caminho com maior número de saltos.

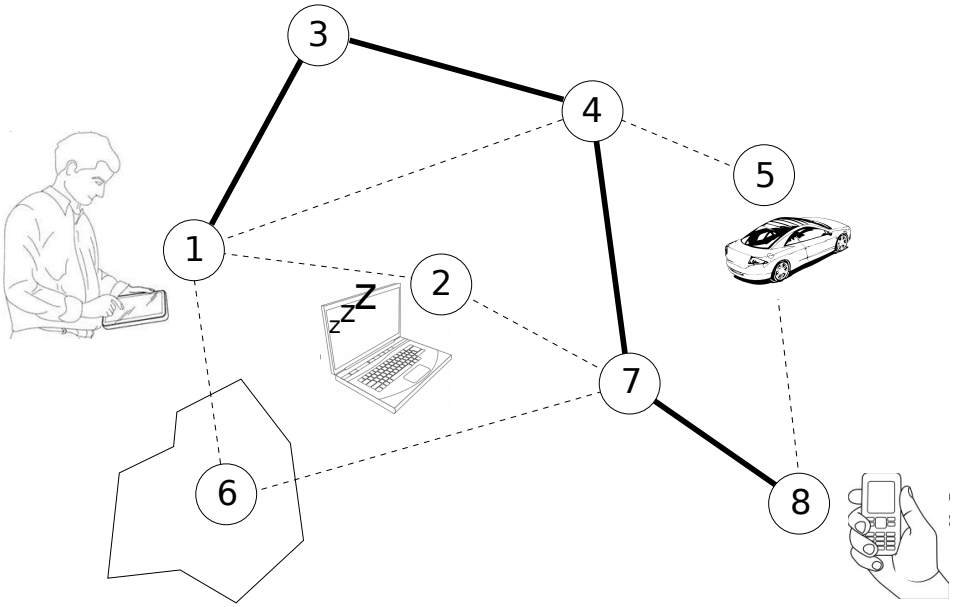

Figura 1.1: Escolha da melhor rota em uma aplicação de computação ubíqua

A presente pesquisa atuou no campo da confiança no roteamento com o objetivo de melhorar o desempenho da comunicação em ambientes de computação ubíqua. A Figura 1.2 aponta a área de concentração explorada bem como as principais técnicas empregadas. Foi dado foco nos ambientes de computação ubíqua formados por redes *ad hoc* puras [1], onde todos os nós compartilham das mesmas funções e não existem entidades de controle de maior hierarquia. Os experimentos foram realizados a partir de uma adaptação do protocolo de roteamento DSR (Dynamic Source Routing) (JOHNSON et al., 2001a), um protocolo do tipo *on-demand*.

O mecanismo de avaliação da confiança de rotas foi incluído no núcleo do algoritmo de seleção de rotas do protocolo DSR. Para aprimorar esse algoritmo, foi utilizada a técnica de aprendizagem indutiva [2] a fim de adquirir regras para escolha das melhores rotas. As regras foram obtidas por meio do treinamento de árvores de decisão com

[1] O termo "pura" se refere à categoria de redes que não conta com entidades especiais ou de maior hierarquia, isto é, os dispositivos possuem os mesmos papéis na rede.

[2] Aprendizagem indutiva é uma técnica de aprendizagem de máquina supervisionada que visa extrair um conhecimento (regras) a partir da observação de eventos passados (MITCHELL, 1997).

o algoritmo C 4.5 (QUINLAN, 1993) usando os valores das métricas de roteamento como base de treino. As métricas de roteamento foram computadas com base em dados das mensagens de roteamento em *broadcast* e da observação direta dos nós vizinhos. Os experimentos foram realizados com uma versão modificada do simulador de redes Glomosim (Global Mobile Information Systems Simulation Library) (GLOMOSIM, 1999) e foram automatizadas por *scripts* em Bash [3], AWK [4] e Gnuplot [5].

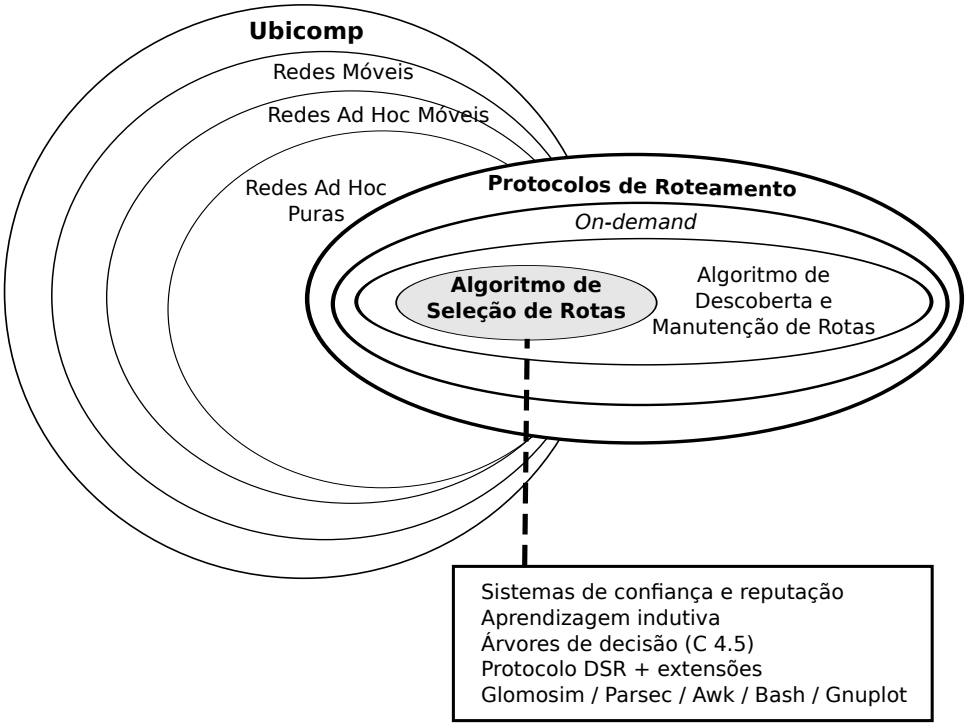

Figura 1.2: Mapeamento da área de concentração da pesquisa

1.2 OBJETIVOS

O objetivo geral da pesquisa é propor e validar um mecanismo de avaliação da confiança de rotas, batizado de METRUbi (*Mechanism for Evaluating the Trust of Routes in Ubiquitous computing environments*), que visa auxiliar os protocolos de comunicação na seleção de rotas em ambientes de computação ubíqua. Esse tipo de mecanismo é vital para que os protocolos das redes *ad hoc* suportem melhor os ambientes de computação ubíqua, pois estão mais sujeitos a problemas derivados da alta densidade, da grande diversidade de dispositivos e do dinamismo da rede.

[3]Bash é uma linguagem de interpretação de comandos dos sistemas operacionais GNU que adere ao padrão IEEE POSIX P1003.2/ISO 9945.2 e oferece recursos de programação e interatividade.

[4]AWK é uma linguagem de programação voltada para extração, transformação e apresentação de dados, criada pela Bell Labs em 1970.

[5]Gnuplot é um utilitário portável para geração de gráficos em diversas plataformas que permite aos estudantes e cientistas a visualização de funções matemáticas e plotagem de dados interativamente.

Para alcançar o objetivo geral, os seguintes objetivos específicos foram estabelecidos:

(a) Caracterizar os cenários de estudo dos ambientes de computação ubíqua;

(b) Implementar o suporte ao modo egoísta e modo *sleep* no simulador de rede (GlomoSim);

(c) Criar *scripts* para automação das simulações;

(d) Estudar os efeitos da alta densidade, do dinamismo, do egoísmo e do modo *sleep* no desempenho da rede por meio de simulações;

(e) Sugerir e validar métricas de confiança das rotas com base no estudo realizado em (d);

(f) Propor o mecanismo de avaliação da confiança de rotas (METRUbi) com base no estudo realizado em (d) e nas métricas estabelecidas em (e);

(g) Incluir as rotinas do METRUbi no protocolo de roteamento DSR;

(h) Comparar os resultados das simulações com o mecanismo de avaliação da confiança ativado.

A presente tese visa apresentar o referencial teórico do tema, relatar as etapas da pesquisa e demonstrar os experimentos realizados e seus resultados. Com a leitura, pretende-se trazer contribuições para outros pesquisadores e viabilizar a continuidade do trabalho.

1.3 VISÃO GERAL DO MECANISMO

A finalidade do METRUbi é auxiliar os protocolos de roteamento na difícil tarefa de selecionar rotas confiáveis em ambientes de computação ubíqua. Para isso, o mecanismo introduz novas métricas de confiança para avaliação das rotas. As métricas são computadas a partir das mensagens de roteamento em *broadcast* e da observação direta dos nós vizinhos.

As informações sobre as rotas e os valores das métricas são armazenadas internamente. Com essas informações é possível gerar regras de decisão por meio de aprendizagem indutiva implementada por árvores de decisão. O treinamento das árvores de decisão

é realizado periodicamente pelo algoritmo C4.5 a fim de evitar o *overtraining* e possibilitar a redenção de nós que apresentaram mal comportamento.

O protocolo de roteamento pode usar as regras geradas para avaliar a confiança das rotas armazenadas no seu *cache*, permitindo, por exemplo, priorizar uma rota em relação a outra existente para um mesmo destino. Todo processo é estocástico, visto que as informações de roteamento e de confiança são bastante voláteis e incompletas nos ambientes de computação ubíqua.

Vale ressaltar que o mecanismo atua em cada nó da rede em uma abordagem puramente *ad hoc*, não existindo entidades especiais que possuem uma abstração maior do comportamento da rede. Dessa forma, cada nó poderá ter uma visão diferente da rede e, por conta disso, as regras serão apropriadas para o seu "pequeno mundo". O resultado esperado é que a junção dessas pequenas melhorias na confiança do roteamento em cada nó possa refletir em um ganho global de desempenho para a rede.

A Figura 1.3 apresenta a arquitetura do METRUbi. Seus componentes são descritos a seguir:

- **Monitor:** captura de pacotes da rede, filtra as informações relevantes e as armazena no Registrador de Confiança. A captura é possível devido à natureza de *broadcast* das redes sem fio e da ativação do modo promíscuo. Também pode descobrir mal comportados de outras vizinhanças usando técnicas como o *Watchdog* (MARTI et al., 2000);
- **Registrador de Confiança:** calcula e armazena as informações sobre a confiança dos nós e das rotas;
- **Indutor DT** [6]**:** faz o treinamento da árvore de decisão com base nas informações presentes no Registrador de Confiança e exporta as regras de decisão para o Avaliador de Rotas;
- ***Cache* de Rotas:** estrutura de dados que armazena as rotas de acordo com cada protocolo de roteamento. Protocolos como o DSR e AODV (LEE et al., 2002) já implementam por padrão o *cache* de rotas.

[6] A sigla "DT"(*Decision Tree*) foi utilizada para fazer referência à técnica de aprendizagem de máquina usando árvores de decisão.

- **Avaliador de Rotas:** faz a avaliação da confiança das rotas com base nas regras de decisão. Sua saída é binária, podendo ser 0 para uma rota avaliada como ruim e 1 para uma rota avaliada como boa. Pode ser integrado aos protocolos de roteamento para que façam melhores escolhas das rotas. Caso haja mais de uma opção boa, pode ser avaliado um segundo critério, como por exemplo, o menor número de saltos.

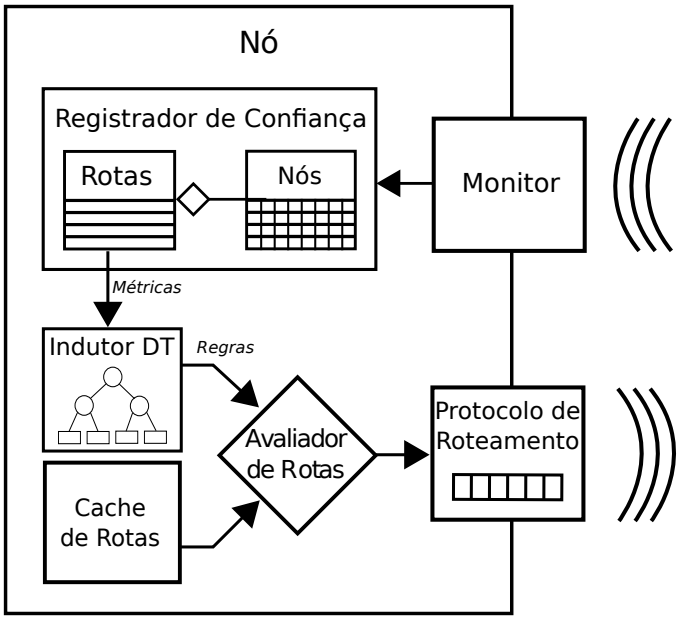

Figura 1.3: Arquitetura do METRUbi

1.4 CONTRIBUIÇÕES

A pesquisa como um todo trouxe as seguintes contribuições:

- Estudo sobre os efeitos da densidade, do dinamismo, do egoísmo e do modo *sleep* no desempenho das redes nos ambientes de computação ubíqua fazendo uso de simulações computacionais. Os resultados podem auxiliar outros pesquisadores na compreensão dos problemas mais relevante, bem como, na proposição de novas soluções.
- Sugestão de métricas multidimensionais para a avaliação da confiança das rotas. Essas métricas envolveram as características de mobilidade, atividade, cooperação e distância das rotas;
- Proposição da arquitetura do METRUbi e dos seus componentes com ênfase para o algoritmo de aprendizagem e geração de regras de decisão sobre as métricas das rotas;

- Adequações no simulador Glomosim, usado nos experimentos, a fim de possibilitar a criação dos cenários de computação ubíqua. Dentre as implementações, foi adicionado o suporte aos nós egoístas e em modo *sleep*. Dessa forma, outros pesquisadores poderão dar continuidade ao trabalho;

- Implementação de ferramentas de automação para as múltiplas simulações. Podem ser úteis para novas pesquisas, nas quais os parâmetros de simulação devem ser variados e os resultados armazenados reiteradamente;

- Prova do funcionamento do METRUbi. Foram realizadas simulações em cenários mistos de computação ubíqua envolvendo nós egoístas, nós em modo *sleep* e mudanças topológicas. Ao final das simulações, os resultados de roteamento e as regras geradas nos diversos nós foram coletadas e analisadas.

1.5 PUBLICAÇÕES

Partes dessa tese foram publicadas por meio de artigos aceitos em congressos e revista de nível internacional, sendo indexados pelos sistemas Capes Qualis, IEEE Xplore, ISI e DOI. Em (SERIQUE; SOUSA, 2005) o problema da confiança em ambientes distribuídos foi primeiramente abordado com sistemas multi-agentes, no qual foi proposto um *framework* para detecção de anomalias. Foi sugerida em (SERIQUE JR et al., 2009) a utilização de algoritmos genéticos como mecanismo de seleção de rotas mais aptas no sentido de minimizar a participação de nós que apresentem comportamentos egoístas.

Mais adiante, foi proposto em (SERIQUE; SOUSA, 2011) um mecanismo de avaliação de rotas que visa mitigar o mal comportamento e outras falhas na rede, usando indução de árvores de decisão. Uma versão estendida desse último artigo encontrada em (SERIQUE; SOUSA, 2012) aprofundou mais a análise dos atributos avaliados para a decisão de roteamento.

1.6 TRABALHOS RELACIONADOS

Marti et al. (2000) abordam duas técnicas que em conjunto podem detectar nós mal comportados e excluí-los das rotas, aumentando o desempenho da rede. A primeira técnica é o *Watchdog* que faz com que os nós monitorem uns aos outros no encaminhamento de pacotes por meio de interfaces de rede em modo promíscuo, permitindo fazer o *acknowledgment* passivo (JUBIN; TORNOW, 1987). A segunda é o *Pathrater*

que elenca as rotas mais confiáveis com base nas informações coletadas dos nós. Esse trabalho contribuiu na definição das premissas do nosso mecanismo.

Buchegger & Le Boudec (2002) analisam a performance do protocolo CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) que se tornou referência em pesquisas nesse campo. O protocolo é formado por quatro componentes: *The Monitor* que é o responsável pelo vigilância dos nós vizinhos; *The Trust Manager* que gerencia o recebimento e envio de mensagens de alarmes relacionadas aos nós maliciosos; *The Cooperation System* que faz a manutenção de *blacklists* e das tabelas de pontuação; e *The Path Manager* que seleciona as rotas com melhores métricas. Esse trabalho também contribuiu na definição das premissas do nosso mecanismo.

Djenouri & Badache (2006) falam de uma abordagem *cross-layer* para detecção de descartadores de pacotes, tanto por motivos de consumo de energia, quanto por ação maliciosa. Neste trabalho exploram as mensagens da camada MAC, chamadas de *two-hop ACK* para divulgação do nós que descartam pacotes. Sundararajan & Shanmugam (2009) propõem outro modelo *cross-layer* para o protocolo AODV chamado de AODV+2ACK que sempre envia, no sentido oposto ao fluxo, mensagens ACK de camada MAC em dois saltos.

Khosravi & Tyson (2006) apresentam extensões para os protocolos de roteamento que permitem a detecção e punição de nós egoístas. Além disso incorpora um sistema de incentivos baseado em uma espécie de moeda chamada de *beans*. Os nós ganham *beans* quando encaminham pacotes de outros, e perdem *beans* quando originam pacotes. Foram feitos experimentos por meio de modificações no código do simulador Glomosim (NUEVO, 2004), sendo bastante proveitosos para a implementação dos experimentos da presente pesquisa.

Seredynski et al. (2008) especificam um mecanismo evolucionário de confiança e incentivo à cooperação por meio de teoria de jogos. O nó de origem é o primeiro jogador, enquanto os nós intermediários caracterizam o segundo jogador. A lógica do jogo está em conquistar o direito ao encaminhamento de pacotes pelos nós intermediários em favor de pacotes encaminhados com sucesso pela origem. A principal contribuição deste trabalho foi a abordagem do parâmetro de atividade e a definição de um conjunto de equações que inspiraram o cálculo da métrica de atividade.

Por fim, Yasser et al. (2007) propõem em a otimização da migração de agentes móveis

combinando técnicas como a classificação de Naive Bayes e as árvores de decisão. A partir de parâmetros coletados pelos agentes, é possível propor melhores rotas para migração. Aqui obtivemos uma visão prática de como aplicar as árvores de decisão em problemas de otimização do roteamento.

1.7 ORGANIZAÇÃO DA TESE

A tese está dividida em cinco capítulos, incluindo o presente capítulo de introdução. O Capítulo 2 faz a revisão da literatura, abordando tópicos como computação ubíqua, roteamento em redes *ad hoc*, métricas de roteamento, confiança em redes *ad hoc* e aprendizagem indutiva.

O Capítulo 3 apresenta a arquitetura e o funcionamento do mecanismo proposto, bem como expõe as premissas para que opere satisfatoriamente. Também apresenta a formulação matemática das métricas usadas na avaliação das rotas.

O Capítulo 4 faz a caracterização dos cenários das simulações e explicita a metodologia e ferramentas empregadas. Ao final, demonstra os resultados alcançados, fazendo uma análise em relação aos objetivos da pesquisa.

Finalmente, o Capítulo 5 traz as conclusões do trabalho, retomando suas contribuições e as perspectivas para a continuidade da pesquisa.

2 REVISÃO DA LITERATURA

Este capítulo tem o objetivo de revisar o referencial teórico utilizado na pesquisa. Se inicia com uma breve discussão sobre a visão da computação ubíqua, suas características e os principais desafios. Em seguida aborda o tema do roteamento em ambientes de redes *ad hoc* móveis com ênfase nos problemas de roteamento e nas possíveis soluções. Alguns trabalhos sobre métricas de roteamento em redes *ad hoc* são então revisados. O protocolo de roteamento *Dynamic Source Route* é estudado com maior profundidade a fim de compreender seu funcionamento e seu emprego nos experimentos da presente pesquisa. Por fim, é estudada a técnica de aprendizagem por indução por meio de treinamento de árvores de decisão, recurso que foi usado no núcleo do algoritmo do METRUbi.

2.1 COMPUTAÇÃO UBÍQUA

A computação ubíqua é um paradigma que combina novas tecnologias de comunicação e sistemas distribuídos a fim de prover uma integração mais natural e mais ampla entre o meio físico e o digital (NIEMELA; LATVAKOSKI, 2004). O termo "computação ubíqua" foi adotado primeiramente por Mark Weiser, chefe do Centro de Pesquisa Xerox PARC, em 1991, no artigo *The Computer for the Twenty-First Century* (WEISER, 1999). Neste trabalho, Mark Weiser descreveu a visão da computação ubíqua do seguinte modo (tradução livre):

> *Por mais de 30 anos muitos projetos de interfaces, e muitos projetos de computadores, têm sido conduzidos pelo caminho da máquina "dramática". Seu mais alto ideal é tornar o computador tão excitante, tão maravilhoso, tão interessante, que nunca gostaríamos de ficar sem ele. Um caminho menos percorrido eu chamo de "invisível"; seu mais elevado ideal é fazer um computador de forma embarcada, tão apropriado, tão natural, que usaríamos sem sequer pensar nele ... Eu acredito que nos próximos 20 anos o segundo caminho virá a dominar. Mas isso não vai ser fácil ...*

A essência deste novo paradigma é a criação de um ambiente computacional no qual o foco é o ser humano e a tarefa que ele deseja realizar, possibilitando assim que as pes-

soas se dediquem às questões de maior importância. A administração desse complexo sistema ficaria a cargo de especialistas. Assim, os computadores se tornarão mais disponíveis e integrados aos ambientes, tornando-se imperceptíveis para os usuários. Weiser (1994) deixa ainda mais claro o foco almejado pelas novas tecnologias, quando faz a seguinte afirmação (tradução livre):

> *Uma boa ferramenta é uma ferramenta invisível. Sendo invisível, quero dizer que a ferramenta não se intromete na sua consciência, você se concentra na tarefa, não na ferramenta. Os óculos são uma boa ferramenta - você olha para o mundo, não para os óculos .*

Vale ressaltar que essa visão vem recebendo diferentes denominações nas pesquisas e na literatura da área. Termos como "computação pervasiva", "tecnologia calma", "internet de objetos" (*internet of things*), "computação invisível", "computação sensível a contexto" (*context-aware computing*), "tecnologias embarcadas", "realidade aumentada", dentre outros, se referem ao mesmo tema, em diferentes disciplinas.

Alguns autores tratam os conceitos de computação ubíqua e computação pervasiva como sendo funções complementares de uma mesma visão. A ubiquidade seria o campo referente aos diversos computadores disponíveis no ambiente para atenderem às necessidades diárias do usuário, independente de sua localização. Já a "pervasividade" cuidaria da integração e comunicação desses computadores no ambiente. Nesse sentido, Araujo (2003) coloca que é possível entender a computação ubíqua como uma junção das características da computação móvel com a computação pervasiva, conforme mostrado na Figura 2.1.

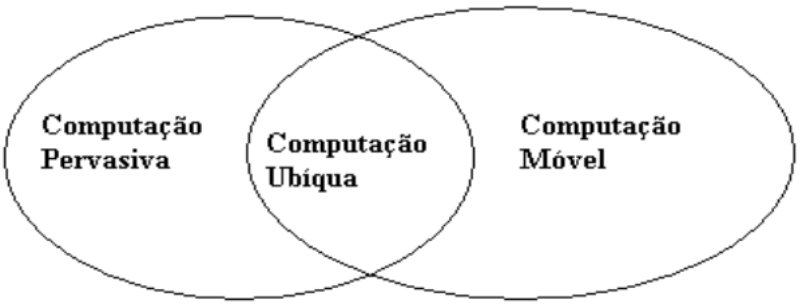

Figura 2.1: Relação computação pervasiva, ubíqua e móvel
(ARAUJO, 2003)

Augustin et al. (2008) expõem que: "computação móvel é um sistema distribuído - que envolve *softwares*, dados, *hardware* e usuário - cuja localização se altera no curso

da execução". A computação móvel pode ser caracterizada por três propriedades que introduzem restrições no ambiente: portabilidade, mobilidade e conectividade. Para ser portátil, um computador deve ser pequeno, leve e incluir pequenas fontes de energia, proporcionando restrições no tamanho de memória, na capacidade de armazenamento, no consumo de energia e na interface do usuário.

Quando em movimento, o dispositivo móvel pode alterar sua localização e seu ponto de contato com a rede. Essa natureza dinâmica do deslocamento introduz questões relativas ao endereçamento dos nós, localização do usuário e informações dependentes da localização. Adicionalmente a aplicação com seu código, dados e estado também pode se deslocar entre os nós da rede. A conexão por meio de rede sem fio gera outros problemas: comunicação intermitente, largura de banda variável e limitada, alta latência e alta taxa de erros.

Weiser (1996) afirma também que a noção de computação ubíqua é oposta à da realidade virtual. Na realidade virtual os usuários são transportados para um ambiente computacional que imita o mundo real, já na computação ubíqua, são os computadores que se transportam para os ambientes reais habitados pelos usuários. Azuma (1997) acrescenta ainda o conceito de realidade aumentada que seria a incorporação de objetos virtuais nos ambientes físicos reais. A Figura 2.2 denota a diferença de ênfase em cada conceito.

2.1.1 Histórico da Computação Ubíqua

Para Weiser & Brown (1997), a computação ubíqua se trata de uma nova fase da computação conforme exposto em trecho do seu artigo *The Coming Age of Calm Technology* (tradução livre):

> *(...) é o nome dado à terceira onda na computação, que está iniciando. Primeiro eram mainframes, servindo várias pessoas. Agora estamos na era da computação pessoal, pessoa e máquina interagindo frente a frente através do desktop. A próxima a vir é a Computação Ubíqua, a era da "tecnologia calma" (calm technology), na qual a tecnologia atuará silenciosamente em nossas vidas. Alan Kay, da Apple, chama isso de "terceiro paradigma" da computação.*

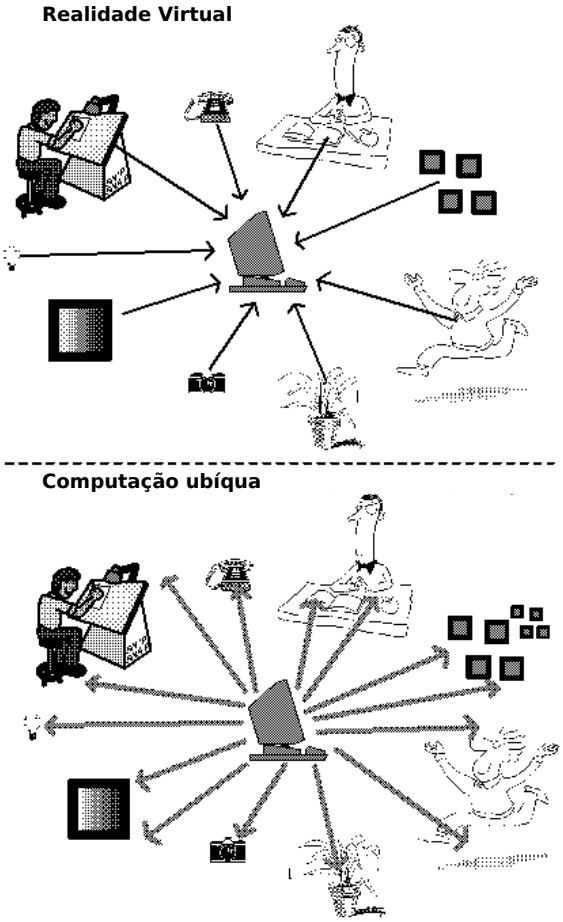

Figura 2.2: Realidade virtual *versus* computação ubíqua
(WEISER; BROWN, 1997)

A primeira fase foi a do *mainframe*, na qual vários usuários compartilhavam um único computador. A segunda fase foi a da computação pessoal, na qual cada usuário operava seu próprio computador. A terceira fase será a da computação ubíqua, na qual vários computadores integrados aos ambientes estão à disposição de cada um de nós. A Figura 2.3 denota as transições entre as três fases.

Apesar do entusiasmo de Mark Weiser quanto à ideia de uma nova geração de computadores, os pesquisadores mais recentes preferem enquadrá-la como um campo de pesquisa interdisciplinar que utiliza tecnologias pervasivas, redes sem fio, tecnologias embarcadas, *wearable computing* [7] e tecnologias móveis para redução da lacuna entre o mundo digital e o mundo real.

[7]Computadores e dispositivos em forma de vestuário (óculos, roupas, pulseiras, relógios, etc.), a exemplo do projeto *Glass*, incubado pela Google, que promete disponibilizar interfaces de navegação e interação por meio de óculos eletrônicos.

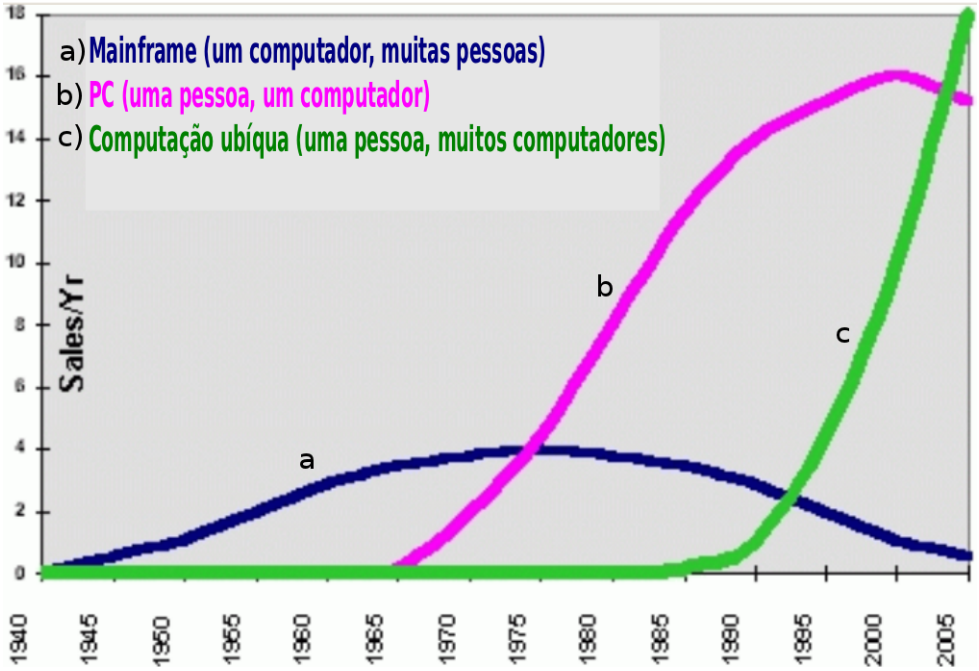

Figura 2.3: Transição dos paradigmas da computação
(WEISER; BROWN, 1997)

A Tabela 2.1 faz um comparativo da computação ubíqua com outros paradigmas da computação com base nos estudos propostos em (HENRICKSEN et al., 2001), (SAHA; MUKHERJEE, 2003), (SATYANARAYANAN, 2001) e (BOHN, 2002).

Tabela 2.1: Comparação da computação ubíqua com outros estágios da computação

	Computação Distribuída	**Computação Móvel**	**Computação Ubíqua**
Topologia	Fixa	Moderadamente dinâmica	Altamente dinâmica
Tecnologia de rede	Redes fixas e autorizadas	Redes sem fio	Redes *ad hoc* de curto alcance
Densidade da rede	Baixa	Média	Alta
Diversidade	Dispositivos homogêneos	Clientes heterogêneos	Dispositivos heterogêneos
Mobilidade	Dispositivos estáticos	Clientes móveis	Dispositivos móveis
Arquitetura	Cliente-servidor	Cliente nômade e servidor	Dispositivo é cliente e também servidor
Conectividade	Permanente	Intermitente	Intermitente
Descentralização	Baixa	Média	Alta
Fonte de energia	Abundante e constante	Limitada e constante	Escassa e variável
Tamanho do dispositivo	Metros	Decímetros	Centímetros
Posicionamento	Absoluto (fixo)	Absoluto (variável)	Relativo (proximidade)
Relação com o usuário	Um dispositivo para muitos usuários	Um dispositivo por usuário	Vários dispositivos por tarefa do usuário
Tempo das relações	Longo	Curto	Muito curto
Intervenção do usuário	Alta	Média	Baixa

2.1.2 Características da Computação Ubíqua

As aplicações de forma geral têm sido motivadas a adotarem recursos da computação ubíqua. Isso se deve às características desses recursos, tais como a grande disponibilidade e conectividade entre dispositivos, sem a necessidade de uma infraestrutura preexistente, e a baixa intervenção do usuário (BOUKERCHE; REN, 2008). A Tabela 2.2 relaciona as principais características da computação ubíqua com base na revisão

da literatura apontada.

Tabela 2.2: Características da computação ubíqua

Característica	**Descrição**	**Exemplo**
Onipresença do Serviço	Garante que o sistema se desloque juntamente com o usuário, permitindo seu acesso independente da sua localização. Com isso, o usuário tem a impressão que o sistema está em toda parte. Esse efeito é alcançado por meio da interconexão de dispositivos e sensores espalhados pelo ambiente (GEER, 2006).	Anúncios de várias lojas de um *shopping* que se alteram automaticamente com base nas preferências do cliente.
Invisibilidade	Oferece meios naturais de interação, de forma que o usuário o utilize sem perceber, por exemplo, por meio de gestos e vozes (NIEMELA; LATVAKOSKI, 2004).	O cliente apenas retira o produto da loja e, em seguida, o sistema de reconhecimento facial e sensor RFID ordena o débito na conta bancária.
Sensibilidade ao Contexto	O sistema altera seu comportamento conforme captura informações do ambiente em que está inserido (SAKAMURA, 2006).	Ao perceber que um ambiente do *shopping* está vazio a bastante tempo, o sistema de iluminação é colocado em modo econômico.
Comportamento Adaptável	O sistema se adapta ao ambiente para prover uma funcionalidade (SAKAMURA, 2006).	Um aplicativo de vídeo-chamada reduz a qualidade para evitar atrasos quando a largura de banda é limitada.
Captura da Experiência	O sistema aprende a rotina de seus usuários para que futuramente possa apoiar suas atividades (KOGURE et al., 2003).	Um veículo aprende as rotas utilizadas pelo usuário para ir ao *shopping* e já alerta o mesmo em relação aos possíveis congestionamentos.
Descoberta e Composição de Serviços	O sistema é capaz de descobrir e usar os serviços disponíveis no ambiente autonomamente (FRIDAY et al., 2004). Provê funções complexas com a junção de funções básicas disponíveis no ambiente (SAKAMURA, 2006).	O usuário entra na loja e seu celular mostra os produtos em oferta e a quantidade em estoque.
Interoperabilidade Espontânea	O sistema permite que os dispositivos se comuniquem sem intervenção do usuário (NIEMELA; LATVAKOSKI, 2004).	Telefones celulares se alinham com os computadores das lojas e trocam informações de contatos.
Heterogeneidade de Dispositivos	O sistema pode ser utilizado em dispositivos de natureza diferente (NIEMELA; LATVAKOSKI, 2004).	Para emitir o bilhete do cinema, o usuário tem a opção de usar o terminal ou seu celular.
Usabilidade e Conectividade Universal	A interface adere a padrões de usabilidade prevendo a necessidade de diferentes grupos de usuários (WANT; PERING, 2005).	Os usuários, dispositivos e objetos de qualquer região do globo terrestre estão continuamente conectados.

Uma importante característica presente na computação ubíqua é a sensibilidade ao contexto. Existem muitos estudos e definições sobre o que é contexto. Dey (2001) define o contexto do seguinte modo (tradução livre):

> *Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade pode ser uma pessoa, lugar ou*

> *objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação (tradução livre).*
> Anind K. Dey

Schilit & Adams (1994) expõe que a computação sensível se refere a "um software que se adapta de acordo com a localização de uso, o conjunto de pessoas, servidores e dispositivos nas proximidades, bem como as mudanças dessas coisas com o passar do tempo". Dey (2001) ainda complementa que o sistema é sensível ao contexto se ele usa o contexto para prover informações e serviços relevantes, no qual a relevância depende das tarefas do usuário.

Brooks (2003) considera que a sensibilidade ao contexto possui cinco aspectos:

- **Quem:** é a sensibilidade à identidade do usuário. Esse aspecto gerencia os perfis de usuário e como o contexto o diferencia para obter as permissões necessárias;
- **Onde:** é a sensibilidade à localização. Esse é o conhecimento da localização geográfica dos usuários e dos objetos que serão envolvidos em uma tarefa;
- **Quando:** é a sensibilidade ao tempo. Esse aspecto se refere à aquisição e manutenção de informações sobre o horário e a data, além de calendários estáticos e o dinâmicos de acordo com o usuário;
- **O Que:** é a sensibilidade à tarefa. Esse aspecto foca no que o usuário está fazendo e os objetivos que deseja alcançar;
- **Como:** é a sensibilidade ao meio de comunicação. Esse aspecto se refere às diversas interfaces de computador que permitem uma comunicação fácil e integrada ao ambiente.

Existem também características de desempenho vitais para os ambientes de computação ubíqua. De acordo com Satyanarayanan (2001) e Krukow et al. (2008), essas características são as seguintes:

- **Gerenciamento de Energia:** o sistema deve controlar automaticamente o uso da energia visando maximizar o tempo de vida das baterias, especialmente quanto em estado ocioso;

- **Tolerância a falhas:** o sistema se adapta diante de falhas no ambiente e continua a prover as funcionalidades essenciais;
- **Confiança e privacidade:** o sistema mantém sob sigilo as operações executadas por um usuário e garante que este não seja burlado no contexto do sistema;
- **Escalabilidade:** indica a habilidade do sistema de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer;
- **Qualidade de serviço:** indica a capacidade do sistema em manter o funcionamento de seus serviços em níveis satisfatórios durante sua execução;
- **Mobilidade:** o usuário do sistema pode se mover para qualquer localidade sem interrupção nas aplicações;
- **Autonomia:** se refere à capacidade dos agentes de tomar decisões e executar procedimentos sem intervenção humana para garantir a estabilidade do ambiente.

2.1.3 Desafios da Computação Ubíqua

À medida que os dispositivos computacionais se tornam mais embarcados e móveis, a natureza das interações entre os usuários e os computadores também se transforma. As aplicações devem se tornar cada vez mais autônomas e "invisíveis", contando com maior conhecimento do contexto e com menor intervenção dos usuários. Além disso, as aplicações devem lidar com ambientes altamente dinâmicos e cujos recursos, como a conectividade de rede e serviços de *software*, variam a todo instante. Essas características por si só já proporcionam desafios complexos para os pesquisadores.

É crescente o número de dispositivos que incorporam microprocessadores e recursos de conectividade: eletrodomésticos, televisores, *smartphones*, veículos, relógios, *notebooks*, *tablets*, etc. Fica claro que a tendência é o aumento da heterogeneidade e da quantidade de dispositivos, acarretando em incompatibilidades de *software* e *hardware*. Paralelamente, os dispositivos estão se tornando cada vez mais portáteis e móveis. Essas propriedades requerem o conhecimento das diferentes configurações e uma infraestrutura que garanta a escalabilidade, a conectividade e a interoperabilidade dos dispositivos.

Além da mobilidade dos dispositivos, os componentes de *software* também devem ser móveis para se instalarem nos diferentes ambientes. Para isso, a infraestrutura deve suportar a migração do código e dos dados sem grande esforço dos programadores,

mantendo uma sincronização e coordenação dos componentes de forma transparente. Plataformas como as máquinas virtuais Java podem se encarregar dessa tarefa. Os usuários dos ambientes de computação ubíqua podem mover-se livremente sem o encerramento da sessão estabelecida com os diferentes dispositivos. O papel da infraestrutura é manter todas as informações do contexto e gerenciar as ações referentes à mobilidade.

Outro grande obstáculo é o emprego das redes sem fio para garantir a mobilidade dos usuários e das aplicações. Segundo (DEUS, 2006), como as pesquisas recentes buscam expandir a capacidade de transmissão de dados das redes sem fio para fornecer serviços multimídia, haverá um aumento na demanda dos usuários pela dependência de serviços móveis de dados. Deus (2006) coloca ainda que para fornecer um acesso a Internet com boa qualidade de serviço, será requerida uma rede projetada para fornecer largura de banda suficiente para uma area com um grande número usuários que exigem taxas de dados elevadas, porém as abordagens atuais de planejamento das redes sem fio não projetam redes para taxas de dados específicas.

A invisibilidade das aplicações é alcançada com a redução das entradas dos usuários e aumento do conhecimento do contexto. Logo, os componentes de *software* terão que explorar um grande número de informações do ambiente (tarefas cotidianas, proximidade com outros dispositivos, localização, condições climáticas, etc.) e terão que se adaptar na ocorrência de mudanças.

As interfaces de usuário também devem ser aprimoradas nos ambientes de computação ubíqua. Primeiramente para garantir a universalidade e adaptabilidade das interfaces, permitindo novos meios e formatos de interação. Outra questão é a da usabilidade que visa oferecer maior ergonomia, maior foco na tarefa do usuário e maior intuitividade.

A fim de experimentar uma tentativa de taxonomia para os desafios, Henricksen et al. (2001) fazem uma divisão dos problemas em quatro grandes áreas - suporte aos dispositivos, suporte aos softwares, suporte aos usuários e suporte às interfaces de usuário -, conforme mostrado na Tabela 2.3.

2.2 ROTEAMENTO EM REDES AD HOC

Devido ao avanço das tecnologias de comunicação sem fio, diversos dispositivos móveis têm utilizado amplamente interfaces de rede sem fio de baixo custo, possibilitando a

Tabela 2.3: Desafios da computação ubíqua

Desafio	**Problema**
Suporte aos Dispositivos	• Heterogeneidade • Mobilidade
Suporte aos Softwares	• Mobilidade e distribuição das aplicações • Sensibilidade ao contexto • Adaptação • Interoperabilidade dinâmica • Descoberta de componentes • Implantação e desenvolvimento ágeis • Escalabilidade
Suporte aos Usuários	• Gerenciamento do contexto • Mobilidade do usuário
Suporte às Interfaces de Usuário	• Interfaces universais • Adaptação das interfaces • Usabilidade

criação de redes de dispositivos móveis ou, simplesmente, redes móveis. Nesse contexto, as redes móveis têm atraído atenção nos últimos anos devido à sua comprovada flexibilidade e custos reduzidos. Sun (2001) afirma que essa tecnologia de redes é vital para a Computação Ubíqua.

Comparadas às redes cabeadas, as redes móveis possuem características únicas. Por exemplo, a movimentação dos nós pode causar rápidas mudanças topológicas. A capacidade do enlace também varia continuamente devido às oscilações no nível de sinal, ruídos, atenuações e interferências. Adicionalmente, essas redes são restringidas por altas taxas de erros, largura de banda limitada, bem como pela escassez de energia dos nós. As redes móveis podem ter dependência ou não de estruturas fixas (GROUP, 2000). Em uma rede sem fio estruturada, nós móveis possuem pontos de acessos cabeados (ou estações base) dentro de sua área de cobertura, sendo que os pontos de acesso formam o *backbone* da rede.

Em contraste, redes *ad hoc* são redes sem fio autônomas e auto-organizadas, sem o suporte de uma infraestrutura e sem uma entidade de controle. Basagni et al. (2004) define uma rede *ad hoc* como "uma rede composta por nós iguais que se comunicam por enlaces sem fio, sem qualquer tipo de controle central". Nestas redes, os nós se movem arbitrariamente causando mudanças rápidas e imprevisíveis na topologia. Adicionalmente, devido a limitação do alcance de rádio, alguns nós não podem se comunicar diretamente o que ocasiona a necessidade de todos os nós atuarem como roteadores, resultando em uma rede de saltos múltiplos (*multi-hop*). Esse tipo de rede

é comumente referenciada como rede *ad hoc* móvel ou, simplesmente, MANET (*Mobile Ad hoc Network*).

Amvame-nze (2006) coloca que a demanda por conectividade móvel em ambiente sem fio fez surgir o conceito das redes *ad hoc*, de características temporárias e autoconfiguráveis, por meio das quais qualquer nó dentro do alcance de rádio pode se comunicar diretamente, sem a dependência de uma infraestrutura preestabelecida. Logo, as conexões por saltos simples (*single-hop*), típicas das redes sem fio infraestuturadas, nas quais cada nó se comunica com uma estação-base central evoluem para conexões de saltos múltiplos, na quais os nós adjacentes são utilizados para realizar o roteamento desde o nó de origem até o nó de destino, quando estes não possuem conectividade direta. Essas redes, de saltos múltiplos, têm grande flexibilidade e rapidez na sua implantação, sendo adequadas para muitas aplicações, como a comunicação em ambientes hostis, como os encontrados em campos de batalha ou situações de calamidade.

As redes *ad hoc* incorporam importantes avanços às redes sem fio, contudo sua tecnologia ainda está em fase de maturação. Além de apresentarem todos os problemas tradicionais das comunicações sem fio, as redes *ad hoc* adicionam novos problemas devido à natureza de multi-saltos e à ausência de infraestrutura fixa, tais como o controle topológico, a descoberta e manutenção de rotas e o roteamento autônomo.

Redes *ad hoc* oferecem a vantagem da fácil e rápida implantação, melhorando a flexibilidade e reduzindo os custos, mesmo em ambientes sem uma infraestrutura preestabelecida. Dessa forma são mais apropriadas para aplicações que requerem mobilidade, nas quais nenhuma infraestrutura foi prevista e os custos devem ser baixos. Esse tipo de aplicação está ganhando cada vez mais importância em organizações não militares e no terceiro setor de forma geral.

Muitas abordagens e protocolos têm sido propostos e existem diversas iniciativas de padronização pelo IEEE (*Institute of Electrical and Electronic Engineers*) e pelo IETF (*Internet Engineering Task Force*), bem como projetos acadêmicos e industriais. As pesquisas são direcionadas principalmente para os campos de controle do meio de acesso, roteamento, gestão de recursos, gerenciamento de energia e segurança. Devido à importância do roteamento para as redes *ad hoc* móveis, muitos protocolos de comunicação foram propostos nos últimos anos.

Projetar protocolos de roteamento para redes *ad hoc* móveis é uma tarefa complexa.

Primeiro devido à mobilidade dos nós que acarretam em constantes mudanças de topologia. Segundo, porque a capacidade dos enlaces sem fio são variáveis, limitadas e imprevisíveis, ocasionado perdas de pacotes. Vale citar ainda os clássicos problemas de terminal escondido e terminal exposto como consequência da difusão no meio sem fio (JAYASURIYA et al., 2004). Adicionalmente, os nós móveis possuem limitação de energia, processamento e largura de banda, fazendo com que os protocolos tenham que lidar com frequentes inoperâncias.

Como são fortes candidatas a substituírem as redes convencionais no futuro, principalmente em virtude dos ambientes de Computação Ubíqua, as redes *ad hoc* tem atraído cada vez mais a atenção dos pesquisadores. Esse capítulo fará a revisão de aspectos importantes observados na presente pesquisa: uma visão geral das tecnologias que habilitam as redes *ad hoc*, o roteamento em redes *ad hoc* e métricas de roteamento.

2.2.1 Breve Histórico das Redes Sem Fio

A ideia das redes sem fio, bem como as redes *ad hoc*, surgiu por volta de 1970 com o sistema ALOHA. O ALOHA era utilizado para conectar as sub-redes universitárias, presente em quatro ilhas, ao centro de computação da Universidade do Havaí. Para isso usava oito transceptores de rádio FM com um protocolo de comunicação bastante simples. Quando um terminal tinha dados a enviar, ele simplesmente fazia a transmissão. Se a central recebesse os dados corretamente, enviava uma mensagem de confirmação para o terminal. Se o terminal não recebesse a confirmação dentro de um intervalo de tempo predefinido, era feita a retransmissão dos dados.

Por volta de 1972, o Departamento de Defesa americano (DoD) fez uso de uma tecnologia de acesso ao meio similar ao protocolo ALOHA para implementar a PRNET (*Packet Radio Networks*) - uma rede militar que poderia sobreviver mesmos após as falhas ou a destruição dos sistemas de rádio. A PRNET foi considerada a primeira rede *ad hoc* baseada em tecnologias de redes sem fio em nível MAC e físico. Inicialmente a principal motivação era a necessidade de atender aplicações militares que requeriam tolerância a falhas, independência de infraestrutura e sigilo, porém em pouco tempo chamou a atenção de universidades e empresas interessadas na comercialização da tecnologia.

Mais tarde a PRNET foi substituída pela SURAN (*Survivable Adaptive Radio Networks*) que aprimorou as características físicas e o protocolo de roteamento da rede. Apesar do crescente desenvolvimento das redes *ad hoc* desde às iniciativas do DoD, só houve

uma real evolução por volta dos anos 90, quando essa funcionalidade foi inserida em diversos produtos, como laptops, PDAs (personal digital assistants) e telefones celulares. Além disso a evolução do *hardware* e *software*, em especial o *software* livre, viabilizou a interconexão dos computadores à grande rede da Internet.

Rapidamente as indústrias começaram a se preocupar com o desenvolvimento de padrões, tecnologias e produtos que habilitassem os dispositivos móveis sem fio. Antigos problemas das redes sem fio, já detectados na época da PRNET, começaram a ser revistos, encadeando várias iniciativas de pesquisa que contribuíram para a base das redes *ad hoc*. O grupo de trabalho do IEEE 802, responsável por padrões de redes de computadores, estabeleceram um subcomitê, o IEEE 802.11, para uniformizar as técnicas e tecnologias de redes sem fio.

O subcomitê, composto por membros de universidades e empresas, verificou a necessidade de padronizar tanto as tecnologias de comunicações baseadas em infraestrutura, quanto sem infraestrutura (redes *ad hoc*). Em 1997 o 802.11 aprovou o primeiro padrão de redes sem fio, definindo a camada física, a camada MAC e a camada LLC para comunicações com infraestrutura e *ad hoc*.

A primeira geração das redes sem fio operavam na faixa de 900 MHz, com uma taxa de até 500 Kbps, mas só era usada em sistemas proprietários e militares. A segunda geração se iniciou por volta de 1997 motivada por grandes esforços de padronização do IEEE 802.11 para usar a faixa de 2,4 GHz com taxas de até 2 Mbps. Pouco tempo depois, surgiu a terceira geração com a padronização do grupo IEEE 802.11b e a grande oferta de produtos mais baratos. Com o IEEE 802.11b a faixa de 2,4 GHz proporcionava taxas de até 11 Mbps. Paralelamente surgiu o grupo 802.11a que operava na faixa dos 5 GHz com taxas de até 54 Mbps, que mais tarde resultou no padrão IEEE 802.11g.

A quarta geração se deu com o padrão 802.11n que adotou antenas do tipo MIMO (*Multiple Input, Multiple Output*) para possibilitar vários canais simultâneos, oferecendo taxas de até 600 Mbps. Os padrões de quinta geração, 802.11ac e 802.11ad, são os mais recentes e visam taxas da ordem dos Gigabits. A evolução dos padrões 802.11 foi viabilizada por novos esquemas de modulação, aumento da largura de banda e melhoria da tecnologia e quantidade das antenas conforme o sumário da Tabela 2.4.

Tabela 2.4: Evolução dos padrões 802.11 e suas características

Padrão	**Ano**	**Frequência**	**Larg. Banda**	**Modulação**	**Antena**	**Taxa Max.**
802.11	1997	2,4 GHz	20 MHz	DSSS	Comum	2 Mbps
802.11b	1999	2,4 GHz	20 MHz	CCK	Comum	11 Mbps
802.11a	1999	5 GHz	20 MHz	OFDM	Comum	54 Mbps
802.11g	2003	2,4 GHz	20 MHz	CCK/OFDM	Comum	54 Mbps
802.11n	2009	2,4/5 GHz	20/40 MHz	OFDM(64-QAM)	4 x MIMO	600 Mbps
802.11ac	2013	5 GHz	40/80/160 MHz	OFDM(256-QAM)	8 x MIMO	6.93 Gbps

Enquanto o grupo IEEE 802.11 trabalhava na padronização de redes sem fio, o ETSI (*European Telecommunication Standards Institute*) trabalhava em outro padrão chamado de HiperLAN (High Performance Radio LAN). O padrão HiperLAN foi lançado no mesmo período do IEEE 802.11b, contudo não recebeu a mesma atenção devido aos rigorosos requisitos de fabricação. O HiperLAN operava na faixa de 5 GHz com taxas de até 20 Mbps e no ano de 2000 recebeu sua segunda versão, o HiperLAN 2, com taxas similares as do IEEE 802.11a e suporte à integração com redes sem fio 3G (UMTS) e garantia de qualidade de serviço (QoS).

Basagni et al. (2004) expõem que, apesar de existirem outras tecnologias para redes sem fio que poderiam ser usadas em redes *ad hoc*, o IEEE 802.11 se tornou o padrão de fato para as redes locais sem fio (WLAN [8]), enquanto que a tecnologia Bluetooth ocupou seu espaço nas pequenas redes pessoais sem fio (WPAN [9]).

2.2.2 Padrão IEEE 802.11

O IEEE 802.11 é um padrão para transmissão de dados digitais na faixa de frequência de 2.4 GHz que tem o objetivo de prover a interconexão de computadores via interfaces de rede sem fio e a comutação com uma infraestrutura de redes cabeadas. O padrão define por completo a camada física (PHY) e a camada de acesso ao meio (MAC) do modelo de referência OSI. O método de acesso em nível MAC é baseado no protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*).

Conforme mostrado na Figura 2.4, o padrão 802.11 suporta dois tipos de arquiteturas: redes sem fio com infraestrutura ou redes sem fio em modo *ad hoc*. Na arquitetura

[8]WLAN (*Wireless Local Area Network*) é uma rede sem fio de médio porte que opera, por exemplo, dentro dos limites de uma organização.

[9]WPAN (*Wireless Personal Area Network*) é uma rede de curto alcance formada por dispositivos de usuários de uma pequena área, por exemplo, de um escritório ou uma sala de aula. O subcomitê IEEE 802.15 é responsável pela especificação dos seus padrões.

de redes sem fio com infraestrutura, chamada de BSS (*Basic Service Set*), existe um controlador central para cada célula, chamado de AP (*Access Point*). Cada célula é identificada por um BSSID baseado no endereço MAC do AP. Geralmente o AP é conectado à rede cabeada para prover o acesso à Internet para os dispositivos móveis. Todo tráfego passa pelo AP, mesmo entre os dispositivos de uma mesma célula.

As células vizinhas podem usar canais de frequência ligeiramente diferentes a fim de evitar interferências. Todas as células são conectadas para compor um único meio de *broadcast* em nível de LLC. Um sistema de distribuição, não especificado pelo padrão, deve tratar do encaminhamento de pacotes para fora das células até a rede cabeada. É possível interligar via enlace sem fio diferentes APs, permitindo uma arquitetura multi-salto com até dois níveis.

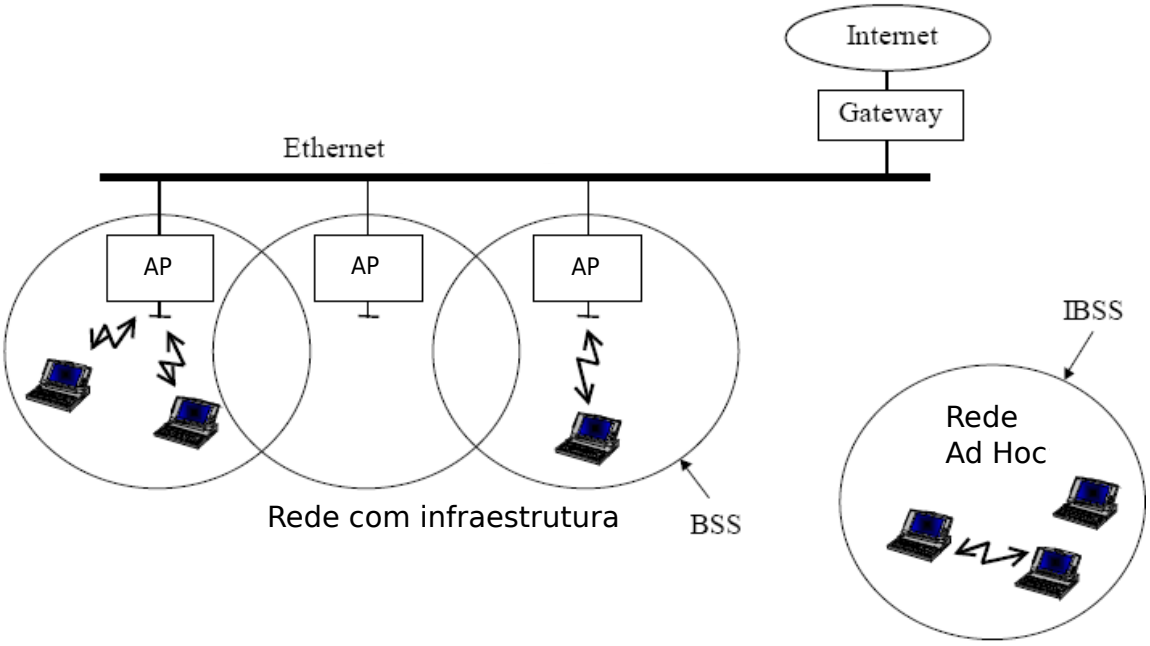

Figura 2.4: Modos de operação do IEEE 802.11
(GIORDANO, 2002)

Na arquitetura *ad hoc*, cada dispositivo da célula, chamado de IBSS (*Independent Basic Service Set*), pode se comunicar diretamente com qualquer outro dispositivo, sem a intervenção de uma entidade central ou a necessidade de uma infraestrutura. Em uma célula *ad hoc*, identificada por um IBSSID localmente atribuído, todos dispositivos devem usar uma frequência pré-definida.

Para que um dispositivo conecte um IBSS é necessário ocorrer uma varredura prévia de sincronização a fim de possibilitar o pleno funcionamento do CSMA/CA. Caso a varredura não encontre algum IBSS, o dispositivo pode iniciar um novo. A manutenção

da sincronização é implementada por um algoritmo distribuído baseado na transmissão de quadros de *beacons* periodicamente.

Devido aos uso de um algoritmo de *backoff*, o número de dispositivos que transmitem em um *slot* aumenta com o número de redes ativas, aumentando a probabilidade de colisões. Adicionalmente, devido às limitações de energia dos dispositivos, o 802.11 também disponibiliza políticas de economia de energia.

Nas redes 802.11, cada ponto de acesso tem uma área de cobertura, ou seja, uma faixa de operação limitada entre 20 a 300 metros em ambientes abertos. As estações móveis que operam dentro da área de cobertura de um ponto de acesso são capazes de receber sinal desse ponto de acesso. Múltiplos pontos de acesso são tipicamente instalados a fim de fornecer, sem interrupção, conectividade contínua às estações móveis assim que elas se movam de uma posição para outra (DEUS, 2006).

Apesar do padrão 802.11 ser uma plataforma muito interessante para experimentos de redes *ad hoc*, não conta com o pleno suporte a redes multi-salto em nível físico e MAC. Consequentemente é necessário empregar protocolos adicionais para esta tarefa. As subseções seguintes discorrem sobre protocolos de roteamento em redes *ad hoc* que habilitam o suporte a redes multi-saltos.

2.2.3 Protocolos de Roteamento em Redes *Ad Hoc*

Giordano (2002) expõe que existem dois serviços básicos necessários à formação das redes *ad hoc*: roteamento e autoconfiguração. O roteamento está relacionado com a natureza multi-salto da rede que deve levar em consideração as constantes mudanças na topologia da rede em função da mobilidade dos nós. Já a autoconfiguração se refere à associação dos nós à rede, permitindo uma implantação rápida e com pouca intervenção dos usuários.

A RFC 2501 (CORSON, 1999) elenca algumas características qualitativas desejáveis para os protocolos de roteamento em redes ad-hoc móveis: operação distribuída, eliminação de *loops*, operação sob demanda, operação proativa, segurança, suporte ao modo *sleep* e suporte a enlaces unidirecionais. Elenca ainda características quantitativas dos protocolos: vazão e atraso na transmissão de dados fim a fim, tempo para aquisição de rotas, percentual de *overhead* pacotes de controle de roteamento e taxa de entrega bem-sucedida de pacotes de dados e controle. Por fim, enfatiza a necessidade de contextualizar o ambiente de operação do protocolo, principalmente nos aspectos

de tamanho e conectividade da rede, taxa de mudança topológica, capacidade dos enlaces, proporção de enlace unidirecionais, padrões de tráfego, mobilidade e proporção e frequência de nós em modo *sleeping*.

Do ponto de vista da confiança, os protococos de roteamento das redes *ad hoc* devem ainda possuir as propriedades apontadas em (LUNDBERG, 2000):

- **Garantia de descoberta de rotas:** se existir uma rota entre dois pontos na rede, deve ser possível encontrá-la. Além disso, o nó que requisita a rota deve ter garantias de que se destina ao nó correto.
- **Isolamento:** o protocolo de roteamento deve ter condições de identificar os nós mal comportados e isolá-los de forma que não interfiram no roteamento. Também deve ser projetado para ser imune aos nós maliciosos.
- **Processamento leve:** muitos dos dispositivos que se conectam à rede *ad hoc* são alimentados por baterias e possuem baixo poder computacional. Por isso não são capazes de fazer processamentos pesados. Se operações como criptografia de chaves públicas e algoritmos de menor caminho são realmente necessárias, devem ser feitas com um grupo menor de nós. Isso evitaria ataques triviais de negação de serviço na rede.
- **Autoestabilização:** garante que, em caso de falhas ou ataques, o protocolo de roteamento consiga restaurar a situação normal da rede, sem intervenção humana. Se existe autoestabilização, um atacante não conseguirá trazer danos permanentes a rede após enviar um conjunto de pacotes corrompidos. Para fazer isso, o atacante deverá enviar continuamente pacotes maliciosos, o que torna mais fácil sua localização e bloqueio.
- **Robustez contra ataques bizantinos:** os ataques bizantinos ocorrem quando nós participam normalmente no encaminhamento de pacotes da rede, porém fraudam os seus conteúdos, visando a degradação da rede. Os nós da vizinhança acreditam que o nó bizantino é bem comportado. Por conta da autoestabilização, os protocolos de roteamento devem continuar operando, mesmo com a existência de alguns nós bizantinos.

Em redes cabeadas, os algoritmos de roteamento baseados em vetor distância e estado de enlace desempenham bem seu papel devido as propriedades previsíveis da rede, como

a qualidade estática do enlace e a topologia da rede. No entanto, as características dinâmicas das redes *ad hoc* deterioram essa eficiência.

O uso de protocolos de roteamento baseados em vetor de distância ou estado de enlace projetados para redes cabeadas ocasionaria um crescimento inaceitável de *overhead* de mensagens de controle em redes *ad hoc*, o que seria incompatível com as restrições de largura de banda nesses ambientes. Adicionalmente, algoritmos de roteamento de vetor de distância e estado de enlace geram inconsistências nas rotas quando usado em redes móveis (KIRUTHIKA; UMARANI, 2010).

A comunicação em *multicast* é utilizada por aplicações nas quais grupos de nós compartilham interesses comuns por informações específicas. Nesses cenários, o *multicast* supera esquemas baseados em *unicast* devido à economia de banda e processamento em redes cabeadas, pois são evitadas transmissões múltiplas da mesma mensagem para receptores dentro do mesmo grupo. Muitos esquemas de roteamento *multicast* foram propostos para redes cabeadas.

Protocolos de roteamento *multicast*, como *Distance Vector Multicast Routing Protocol* (DVMRP) (WAITZMAN; PARTRIDGE, 1988), *Multicast Open Shortest Path First* (MOSPF) (MOY, 1994), *Protocol-Independent Multicast* (PIM) (FARINACCI et al., 1998) foram amplamente utilizadas em redes cabeadas. Na maioria deles, árvores de distribuição são construídas a partir do emissor para os receptores pertencentes ao mesmo grupo.

Como nas redes cabeadas, a utilização de protocolos de roteamento *multicast* também é atraente em redes *ad hoc*, pois pode economizar banda em canais sem fio. Adicionalmente, a propriedade inerente de difusão dos canais sem fio podem ser exploradas para melhorar o desempenho de *multicast* em redes *ad hoc*. Comparado a esquemas de roteamento *unicast*, projetar protocolos de roteamento *multicast* é mais complexo.

A mobilidade faz a manutenção do rastreamento dos membros do grupo *multicast* ainda mais complicada e cara do que em redes cabeadas. Também, as árvores de distribuição sofrem de frequentes reconstruções devido ao movimento dos nós. Assim, protocolos de roteamento *multicast* para redes *ad hoc* devem incluir mecanismos para lidar com as dificuldades ocasionadas pela mobilidade dos nós e consequente mudanças na topologia.

2.2.4 Classificação dos Protocolos de Roteamento de Redes *Ad Hoc*

De forma geral, os protocolos de roteamento para redes *ad hoc* podem ser classificados em reativos e proativos (ROYER; TOH, 1999). No roteamento proativo, também chamado de *table driven routing*, os nós propagam continuamente informações sobre a topologia, independentemente da existência de tráfego na rede, o que gera certo *overhead*. Os protocolos reativos, ou *on-demand*, visam economizar recursos fazendo a descoberta de rotas apenas quando necessário.

Com protocolos proativos, para manter rotas consistentes e atualizadas entre cada par origem-destino, é necessária a propagação de um grande número de informações de roteamento, mesmo que não sejam usadas. Consequentemente, as rotas estão sempre disponíveis, contudo os protocolos podem não funcionar corretamente quando a mobilidade é muito alta ou quando a quantidade de nós é muito grande. Giordano (2002) ainda acrescenta que a sobrecarga de pacotes de controle, em termos de consumo de tráfego e de energia, é uma séria limitação nas redes *ad hoc*, nas quais a largura de banda e a energia são escassas. Contudo, em redes mais estáticas, tendem a oferecer uma comunicação com menor latência.

Os protocolos reativos criam e mantém rotas entre os pares origem-destino apenas quando necessário, geralmente quando solicitado pelas origens (abordagem *on-demand*). Dessa forma a sobrecarga de mensagens de controle é reduzida drasticamente, mas, em contrapartida, adicionam maior latência devido ao processo de descoberta de rotas. Segundo Giordano (2002), a abordagem *on-demand* se baseia nas seguintes assunções: (a) em topologias dinâmicas as informações de rotas expiram a todo instante; e (b) nem todas as rotas são utilizadas ao mesmo tempo.

Apesar dos protocolos proativos provocarem o desperdício de recursos, principalmente quando rotas são inconsistentes por conta do dinamismo da rede, essa assertiva nem sempre é verdadeira em todos os projetos de protocolos. Macker et al. (2001) explicam que a validade nas decisões de roteamento dependem, em parte, da distribuição do tráfego e do grau de dinamismo da topologia.

Alguns dos protocolos das redes *ad hoc* não são unicamente proativos ou reativos, mas possuem algoritmos híbridos. O objetivo é reduzir o tempo de resposta e a sobrecarga de controle em certos cenários. Evidentemente, a abordagem ideal seria aquela que mantém a atualização das rotas mais usadas, e cria as demais rotas sobre demanda. Macker et al. (2001) acrescentam que para encontrar o equilíbrio correto entre

a operação proativa e reativa em uma abordagem híbrida é requerido algum conhecimento prévio do ambiente de rede e mecanismos adicionais para controlar o modo de operação de forma adaptativa.

O OLSR (*Optimized Link State Routing*) (JAQUET et al., 2001) é um exemplo de protocolo proativo e não uniforme, pois as rotas são atualizadas dinamicamente e existem nós com atribuições especiais chamados de MPRs (*MultiPoint Relays*). O *Wireless Routing Protocol* (WRP) (MURTHY; GARCIA-LUNA-ACEVES, 1996) e o *Destination Sequence Distance Vector* (DSDV) (PERKINS; BHAGWAT, 1994) são exemplos de protocolos proativos. O *Zone Routing Protocol* (ZRP) (BEIJAR, 2002) e o *Hybrid Ad Hoc Routing Protocol* (HARP) (NIKAEIN; BONNET, 2001) apresentam características reativas e proativas.

2.2.4.1 Classificação pelo Modo de Delegação de Tarefas

Liu & Kaiser (2003) propõe outro método de classificação que se baseia nos papéis que os nós desempenham no esquema de roteamento. No caso de um protocolo de roteamento uniforme, todos os nós possuem o mesmo papel, importância e funcionalidade. O *Wireless Routing Protocol* (WSP) (MURTHY; GARCIA-LUNA-ACEVES, 1996), o *Dynamic Source Routing* (DSR) (JOHNSON et al., 2001a) e o *Ad hoc On-Demand Distance Vector* (AODV) (LEE et al., 2002) são exemplos de protocolos de roteamento uniforme. Os protocolos de roteamento uniforme produzem redes planas, ou seja, sem níveis hierárquicos.

Em protocolos de roteamento não uniformes, alguns nós são escolhidos para desempenhar funções especiais de gerenciamento e roteamento. Os protocolos não uniformes podem ser divididos de acordo com a organização dos nós e com as funções delegadas. Sendo assim, os protocolos não uniformes podem ser divididos do seguinte modo: baseados em zona (*zone-based*), em cluster (*cluster-based*) e em nós centrais (*core-node based*).

Os protocolos de roteamento baseados em zona usam diferentes algoritmos de delimitação de regiões, por exemplo, a partir de informações geográficas. Esses protocolos reduzem drasticamente o *overhead* causado pelas informações de roteamento, já que os nós só precisam saber como alcançar outros nós de sua mesma zona, dispensando a compreensão da topologia como um todo. Em alguns casos, nós especiais atuam como roteadores entre as zonas. O *Zone Routing Protocol* (ZRP) e o *Zone-based Hierarquical Link State* (ZHLS) (JOA-NG, 1999) são exemplos de protocolos de roteamento

baseados em zonas.

Os protocolos baseados em cluster usam algoritmos específicos para eleição dos cluster-heads, que são os nós que atuam como pontos focais, gerenciando o roteamento e cadastramento dos nós do cluster. O *Clusterhead Gateway Switch Routing* (CGSR) (CHIANG et al., 1997) e o *Hierarquical State Routing* (HSR) (IWATA et al., 1999) são exemplos de protocolos de roteamento baseado em cluster.

Os protocolos baseados em nós centrais selecionam dinamicamente nós que farão parte do *backbone* da rede. Os nós que integram o *backbone* possuem atribuições especiais, tais como a construção das tabelas de roteamento e a propagação dos pacotes da rede. O *Optimized Link State Routing Protocol* (OLSR) (JACQUET, 2003) e o *Core-Extraction Distributed Ad Hoc Routing* (CEDAR) (SIVAKUMAR et al., 1999) são exemplos de protocolos baseados em nós centrais.

2.2.4.2 Classificação pela Métrica de Roteamento

Não há como desconsiderar as métricas usadas para construção de rotas como uma forma de classificar protocolos de roteamento. A maioria dos protocolos de roteamento para redes *ad hoc* utiliza o número de saltos como métrica. Assim, na multiplicidade de rotas disponíveis é considerado o melhor caminho aquele que passar pelo menor número de nós. Assumindo que todos os enlaces tem a mesma taxa de erro, rotas mais curtas são mais estáveis e podem reduzir o tráfego de *overhead* e a colisão de pacotes.

Entretanto, não se pode assumir a constância das taxas de erros em redes *ad hoc*. Assim, a estabilidade de um enlace deve ser considerada na construção de uma rota. Como exemplo pode-se citar abordagens como *Associativity Based Routing* (ABR) (TOH, 1997a) e *Signal Stability based Routing* (SSR) (DUBE et al., 1997) que são propostas que usam estabilidade de enlace ou intensidade de sinal como métrica de roteamento.

Métricas de QoS [10] também podem se mostrar necessárias para suportar determinadas aplicações móveis. Assim, métricas para QoS apropriadas devem ser usadas para roteamento e encaminhamento de pacotes em redes *ad hoc*. Como nas redes cabeadas, protocolos QoS de roteamento para redes *ad hoc* se utilizam de métricas como largura

[10] QoS (*Quality of Service*) é uma especificação qualitativa e quantitativa dos requisitos de uma aplicação que um sistema multimídia deveria satisfazer a fim de obter a qualidade desejada (SHI, 2001).

de banda, atraso, *jitter*, taxa de perda de pacotes e custo. Por exemplo, largura de banda e estabilidade de enlace são usados como métricas de roteamento no protocolo CEDAR (SIVAKUMAR et al., 1999).

2.2.4.3 Classificação pelo Modo de Avaliação da Topologia, Destinação e Localização

Em um protocolo de roteamento para redes *ad hoc* baseado na topologia, decisões de roteamento são tomadas a partir de informações de topologia coletadas pelos nós. Além de protocolos baseados em topologia, existem protocolos de roteamento que se baseiam no destino. Neste tipo de protocolo, um nó precisa saber apenas o próximo salto no caminho da rota quando encaminha o pacote para o destino. Por exemplo, o DSR é um protocolo baseado na topologia enquanto AODV e DSDV são baseados no destino.

A disponibilidade de GPS ou outro sistema de localização similar permite nós móveis obterem facilmente informações geográficas. Em protocolos de roteamento baseados na localização, a relação entre a posição de um nó de encaminhamento de pacotes e o destino, juntamente com a mobilidade dos nós pode ser utilizado tanto na descoberta de rotas quanto no encaminhamento de pacotes.

Abordagens existentes de roteamento baseado em localização para redes *ad hoc* dividem-se em duas classes. Na primeira, nós enviam pacotes baseados somente em informações de localização. A outra classe usa tanto informações de localização quanto de topologia. *Location Aided Routing* (LAR) (KO; VAIDYA, 2000) e *Distance Routing Effect Algorithm for Mobility* (DREAM) (BASAGNI et al., 1998) são dois típicos protocolos baseados em localização propostos para redes *ad hoc*.

2.2.4.4 Classificação pelo Tipo de *Multicast*

Os métodos de classificação discutidos até agora se aplicam também aos protocolos de roteamento *multicast*. Por exemplo, o *Ad Hoc Multicast Routing* (AMroute) (XIE et al., 2002) e o *Ad hoc Multicast Routing utilizing Increasing id numberS* (AMRIS) (WU; TAY, 1999) são enquadrados como protocolos de roteamento proativos, enquanto que o *On-Demand Multicast Routing Protocol* (ODMRP) (GERLA et al., 1998) e o *Multicast Ad Hoc On-demand Distance Vector* (MAOV) (ROYER, 2000) são categorizados como reativos.

Existe uma classificação específica para os protocolos de roteamento *multicast* em redes *ad hoc*. De acordo com a abordagem de construção dos enlaces, os protocolos de

roteamento *multicast* podem ser baseados em árvore, em *mesh*, em nós centrais ou em grupos de encaminhamento. Os protocolos de roteamento *multicast* baseados em árvore podem ser divididos em *source-rooted* e *core-rooted*. Em árvores do tipo *source-rooted*, os nós de origem são as bases da árvore, exigindo o conhecimento de toda topologia que envolve os destinatários do grupo *multicast*.

2.2.4.5 Mapeamento das Classificações dos Principais Protocolos de Redes *Ad Hoc*

A Tabela 2.5 mostra o mapeamento dos protocolos de roteamento de redes *ad hoc* com base na revisão da literatura sobre as diferentes métodos de classificação.

2.3 MÉTRICAS DE ROTEAMENTO EM REDES AD HOC

Nas últimas decadas, o roteamento de redes sem fio tem despertado bastante interesse nas pesquisas. Apesar dos objetivos dos protocolos de roteamento serem os mesmos em qualquer tecnologia, as redes sem fio oferecem novos desafios que tornam o problema mais complexo e, ao mesmo tempo, mais interessante. Apesar das tecnologias das redes cabeadas terem influenciado os primeiros passos das tecnologias de redes sem fio, são necessárias novas soluções para garantir o bom desempenho da rede.

Como as redes cabeadas são muito estáveis, a maioria dos protocolos de roteamento dessas redes operam proativamente, ou seja, procuram manter rotas atualizadas em todos os nós por meio da propagação continua das informações de topologia. Já as redes sem fio possuem topologias altamente dinâmicas devido à mobilidade dos nós e às constantes rupturas dos enlaces por conta dos fenômenos de propagação.

Essas características fazem com que os protocolos proativos e o respectivo *overhead* associado à manutenção das rotas sejam pouco atrativos para as redes sem fio. Seguem os principais obstáculos para os protocolos de roteamento de redes sem fio de acordo com (PARISSIDIS; KARALIOPOULOS, 2009):

- **Mobilidade dos nós:** os dispositivos sem fio podem se mover livremente. Logo os enlaces podem ser rompidos e a topologia pode ser alterada a todo instante. Como consequência, a manutenção das rotas é muito mais complexa nessas redes;
- **Fenômenos de propagação:** em ambientes de redes sem fio, as transmissões se dão por *broadcast* no meio físico, estando sujeitas aos fenômenos de propagação de radio, tais como: sobreposições, atenuações, ruídos, etc. Mesmo os enlaces mais

Tabela 2.5: Mapeamento dos protocolos de acordo com os métodos de classificação

Uniformidade	Classe	Protocolo	Subclasse
Uniformes	**Proativos**	Wireless Routing Protocol (WRP)	
		Destination Sequence Distance Vector (DSDV)	
		Fisheye State Routing (FSR)	
		Distance Routing Effect Algorithm for Mobility (DREAM)	**Localização**
		Ad Hoc Multicast Routing (AMRoute)	**Multicast**
		Ad Hoc Multicast Routing utilizing Increasing id numberS (AMRIS)	**Multicast**
	Reativos	Dynamic Source Routing (DSR)	
		Temporally Ordered Routing Algorithm (TORA)	
		Ad Hoc On-Demand Distance Vector (AODV)	
		Location Aided Routing (LAR)	**Localização**
		Associativity Based Routing (ABR)	**Link State**
		Signal Stability-based Adaptive Routing (SSR)	**Link State**
		On-Demand Multicast Routing Protocol (ODMRP)	**Multicast**
		Multicast Ad Hoc On-Demand Distance Vector (MAODV)	**Multicast**
Não uniformes	**Por Zona**	Zone Routing Protocol (ZRP)	**Híbrido**
		Hybrid Ad Hoc Routing Protocol (HARP)	**Híbrido**
		Zone-based Hierarquical Link State (ZHLS)	**Híbrido**
		Grid Location Service (GLS)	**Localização**
	Cluster	Clusterhead Gateway Switch Routing (CGSR)	
		Hierarquical State Routing (HSR)	
		Cluster Based Routing Protocol (CBRP)	
	Nó Central	Landmark Ad Hoc Routing (LANMAR)	**Proativo**
		Core-Extraction Distributed Ad Hoc Routing (CEDAR)	**Reativo**
		Optimized Link State Routing (OLSR)	**Proativo**

estáveis sofrem constantes flutuações na qualidade, possibilitando, por exemplo, que a rede conte com enlaces de baixa qualidade que ainda estejam ativos. Essas situações forçam uma revalidação frequente das rotas;

- **Limitações de energia:** a economia de energia dos dispositivos portáteis muitas vezes se torna um objetivo primário nas redes sem fio. Infelizmente os avanços nas tecnologias de baterias são mais lentos do que os avanços na nanotecnologia e na

engenharia eletrônica. Os protocolos devem cuidar para que não ocorra desgaste dos recursos em decorrência de atualizações e computações de roteamento. Além disso, a escassez de energia faz com alguns nós deixem de cooperar no roteamento para resguardar seus recursos;

- **Falta de controle centralizado:** a auto-organização é uma das funcionalidades mais atrativas das redes sem fio. Algumas funções, como o acesso ao meio e o roteamento, são completamente distribuídas e exigem pouca intervenção humana. Logo, a rede não requer qualquer entidade de gerenciamento centralizado, como ocorre nas redes cabeadas. Em contrapartida muitas decisões são tomadas individualmente pelos nós que somente possuem conhecimento de seu ambiente local. Isso inviabiliza otimizações que exigem um conhecimento global do estado da rede. Outra questão importante é que as operações da rede assumem que haverá cooperação de todos os nós, deixando a rede vulnerável ao mal comportamento.

A necessidade de uma visão diferente para o roteamento fez surgir uma diversidade de métricas para melhoria da eficiência na entrega de pacotes. Além da métrica de número de saltos (*shortest path first*), amplamente adotada na maioria dos protocolos de roteamento, a literatura apresenta outras métricas que adicionam maior inteligência para garantir melhores taxas de entrega de pacotes e outros aspectos de desempenho.

Dentre os objetivos das novas métricas, destacam-se o balanceamento de carga, a tolerância a falhas, a redução do *jitter* e o aumento da largura de banda, que são usados como ponto de partida para a melhoria das decisões de roteamento (APOSTOLOPOULOS; GUÉRIN, 1998).

As subseções seguintes fazem uma visão geral das diferentes métricas de roteamento, com ênfase nas métricas propostas para as redes sem fio, e investiga suas categorias, similaridades, vantagens e desvantagens. Cabe ressaltar que se trata de um resumo dos estudos encontrados em (PARISSIDIS; KARALIOPOULOS, 2009) e (BAUMANN et al., 2007).

2.3.1 Objetivos da Otimização do Roteamento

Uma métrica de roteamento é essencialmente um valor atribuído a cada rota que é usado pelo algoritmo de roteamento para selecionar uma rota ou um subconjunto de rotas que pertencem ao conjunto de rotas descoberto pelo protocolo de roteamento. Esses valores representam o custo de uma rota em particular frente a algum objetivo de

otimização. Parissidis & Karaliopoulos (2009) afirmam que os objetivos de otimização buscam:

- **Minimização de atrasos:** esse é objetivo mais comum das funções de roteamento. Os caminhos de rede que oferecem menor tempo de atraso são sempre escolhidos, independentemente das suas confiabilidades. Geralmente as métricas para minimização de atraso beneficiam as aplicações de tempo real;
- **Maximização da probabilidade de entrega dos pacotes na rota:** para aplicações que não são de tempo real, o principal objetivo é garantir uma baixa taxa de perda de pacotes, mesmo quando isso acarreta em maior atraso. É equivalente dizer que deve ser minimizada a probabilidade de perda de dados no caminho fim-a-fim;
- **Maximização da vazão da rota:** o objetivo aqui é escolher um caminho fim-a-fim que agregue enlaces de maior capacidade;
- **Maximização da vazão da rede:** os três primeiros objetivos visam melhorar o desempenho de uma aplicação específica, mas neste caso o objetivo é do sistema como um todo. Também pode ser entendido como a maximização do fluxo global de dados na rede ou, implicitamente, como a minimização global de interferências e retransmissões;
- **Minimização do consumo de energia:** o consumo de energia não é uma preocupação na redes cabeadas, porém em redes de sensores e redes *ad hoc* móveis esse objetivo é essencial para a autonomia dos nós da rede;
- **Balanceamento do tráfego nos enlaces:** esse objetivo também é mais geral e visa garantir que não existem nós ou enlaces com uso desproporcional da rede. Pode ser entendido também como a minimização da diferença entre as cargas máximas e mínimas de tráfego nos enlaces da rede. O balanceamento de carga interfere indiretamente em outros objetivos como o tempo de vida da bateria, vazão dos enlaces usados pelos nós, etc.

Pode-se observar que os três primeiros objetivos se voltam para o desempenho de uma aplicação em particular, enquanto que os três últimos se voltam para o desempenho do sistema como um todo. Cabe ainda mencionar que as métricas de roteamento podem considerar mais de um objetivo ao mesmo tempo. Neste caso as métricas são

chamadas de multidimensionais e combinam diferentes medidas ponderadas de acordo com a prioridade dos objetivos.

2.3.2 Métricas de Enlaces *versus* Métricas Rotas

As decisões de roteamento são feitas com base no valor final da métricas da rota completa (fim-a-fim). Porém, o valor final da métrica deve ser calculado como uma função dos valores estimados das métricas individuais de cada enlace da rota. A função a ser utilizada depende fortemente do tipo de métrica adotada. Parissidis & Karaliopoulos (2009) afirmam que as funções mais usadas para o calculo das métricas são:

- **Somatório:** os valores de métricas dos enlaces são simplesmente somados ao valor final da métrica da rota. Atrasos e número de retransmissões dos enlaces são exemplos de métricas que geralmente se dão por somatório;
- **Produtório:** os valores estimados das métricas dos enlaces são multiplicados para resultar na métrica da rota. A probabilidade de sucesso da entrega de pacotes é um exemplo de métrica multiplicativa;
- **Funções estatísticas:** a métrica da rota poderá ser, por exemplo, o mínimo, a média ou o máximo dos valores encontrados nos enlaces. Por exemplo, a vazão de uma rota é calculada em função do valor mínimo de vazão dentre os enlaces.

Baumann et al. (2007) colocam que as métricas de roteamento podem ser classificadas segundo critérios matemáticos do seguinte modo:

- **Operador de combinação de enlace:** as métricas dos enlaces são agregadas e concatenadas de acordo com certas regras para resultar na métrica da rota completa. Apesar do operador de concatenação ser claramente vinculado à natureza da métrica, a definição do operador de agregação não é muito óbvia;
- **Dinâmicas *versus* estáticas:** uma métrica é dinâmica se seu valor se altera a todo instante, sendo o caso da maioria das métricas. Algumas métricas podem ser estáticas como o número de interfaces de rede e a capacidade máxima de energia de um nó;
- **Simétricas *versus* assimétricas:** sendo m_{ij} o valor da métrica do enlace que liga o nó i ao nó j e m_{ji} o valor da métrica do enlace no sentido inverso, uma

métrica só é simétrica se $m_{ij} = m_{ji}$ para todos os enlaces em qualquer ponto da rede;

- **Dimensão única *versus* multidimensional:** métricas multidimensionais não possuem um valor do tipo real, mas uma coleção de valores ou vetores. Wang & Crowcroft (apud BAUMANN et al., 2007) usam o atraso e a largura de banda para o projeto de uma métrica de duas dimensões e provam que a busca de uma rota mais curta por meio de métricas multidimensionais é um problema NP-completo, tanto para métricas resultantes de produtórios quanto de somatórios.

2.3.3 Métodos para o Cálculo de Métricas

Os nós da rede podem coletar as informações necessárias para o cálculo das métricas de diferentes formas:

- **Reuso de informações locais:** os nós podem usar as informações disponíveis localmente, como, por exemplo, os resultados das operações do protocolo de roteamento. O número de interfaces de rede, o número de nós vizinhos e o tamanho das filas de entrada e saída são outros exemplos de informações locais;
- **Monitoramento passivo:** as informações são coletadas por meio da observação direta do tráfego de rede que chega e parte do nó. Em combinação com outras medidas, pode, por exemplo, estimar a largura de banda disponível nos enlaces;
- **Prova ativa:** pacotes especiais são gerados para medir as propriedades dos enlaces e rotas. Esse método é o que inclui maior *overhead* na rede em proporção à frequência das provas;
- **Prova por sobrecarga:** informações de prova são embutidas no tráfego normal de dados ou nos pacotes de roteamento, dispensando pacotes adicionais e reduzindo o *overhead* para o cálculo das métricas. A medida de atrasos é comumente realizada com este método.

Os dados brutos coletados dos enlaces, seja por monitoramento passivo ou ativo, requerem certo processamento antes de serem usados para construção de métricas eficientes e confiáveis. O parâmetros das medidas, tais como atrasos ou taxa de perda de pacotes dos enlaces, frequentemente estão sujeitos à grande variação. Por isso é desejável que variações de curto prazo não influenciem o valor da métrica, visando

evitar o mesmo fenômeno da auto-interferência de métricas da Arpanet descrito em (KHANNA; ZINKY, 1989 apud PARISSIDIS; KARALIOPOULOS, 2009).

Além disso, as medidas das métricas são submetidas a filtragens no decorrer do tempo. Dependendo do tipo de métrica, as filtragens podem fazer uso das seguintes técnicas:

- **Janelas dinâmicas de histórico:** média calculada sobre o número de exemplos de medidas que varia de acordo com a taxa de transmissão corrente;
- **Intervalos fixos de histórico:** média calculada sobre um número fixo de exemplos de medidas;
- **Média móvel exponencial:** os exemplos de medidas são ponderados de forma que os valores decaem exponencialmente de acordo com sua antiguidade, privilegiando as últimas tomadas de medidas. Sempre que um novo exemplo $m_{exemplo}$ é obtido, o valor de sua métrica é atualizado de forma que $m_{novo} = \alpha \times m_{antigo} + (1 - \alpha) \times m_{exemplo}$ com $\alpha \in [0, 1]$, onde α é o fator de ponderação e m_{antigo} é o valor corrente da métrica.

2.3.4 Métricas de Roteamento em Redes Sem Fio

Nessa subseção são descritas as principais métricas projetadas para as redes sem fio. Inicialmente são discutidas as métricas baseadas em topologia e demonstrada a desvantagem das métricas baseadas em número de saltos na redes sem fio. Em seguida são apresentadas métricas mais robustas em relação aos desafios das redes sem fio, agrupando-as em quatro categorias: baseadas no nível de sinal; baseadas em provas ativas e baseadas na mobilidade.

2.3.4.1 Métricas baseadas na Topologia

A maior vantagem das métricas de roteamento baseadas na topologia é sua simplicidade. Uma informação topológica pode ser, por exemplo, o número de nós vizinhos de cada nó ou o número de saltos até um destino. Geralmente as métricas baseadas na topologia não requerem qualquer monitoramento ativo ou passivo, mas usa apenas dados locais do protocolo de roteamento.

Porém, definir uma topologia em redes sem fio não é trivial, visto que os enlaces são instáveis e dependem essencialmente da potência de transmissão dos nós de origem, sensibilidade na recepção dos nós de destino e das características do meio de propagação,

que são parâmetros que são altamente variáveis. Outro problema é a existência de enlaces assimétricos, pois mesmo que um nó X receba pacotes com sucesso de um nó Y, o mesmo pode ser inválido no sentido contrário.

Apesar de as métricas baseadas em topologia não tratarem uma série de variáveis que têm influência no desempenho das aplicações e da rede como um todo, são bastante usadas devido a sua simplicidade. De fato, um dos tipos de métrica baseada na topologia, a métrica por quantidade de saltos, é sem dúvida a mais popular dentre as métricas de roteamento, tanto em redes cabeadas, quanto em redes sem fio.

O conceito de métricas por quantidade de saltos é simples, bastando contar cada enlace como uma unidade, independente da sua qualidade e de outras características. Essa simplicidade fez com que a maioria dos protocolos de roteamento para as redes cabeadas a adotassem e ainda está presente em protocolos populares das redes *ad hoc* sem fio, tais como o OLSR, o DSR, o DSDV e o AODV. A ideia de minimizar o número de saltos implica na redução de atrasos e economia de recursos na rede, porém isso só é verdade para enlaces livre de erros.

2.3.4.2 Métricas baseadas na Intensidade do Sinal

A intensidade do sinal tem sido usada como métrica de qualidade dos enlaces em vários protocolos de roteamento para redes sem fio. A hipótese é que se um pacote é recebido quando a intensidade do sinal excede certo limiar, essa intensidade deve ser tratada como um bom indicador de qualidade do enlace. Parissidis & Karaliopoulos (2009) esclarecem que o valor da intensidade do sinal pode ser usado de duas maneiras no roteamento:

- Como um parâmetro de controle para excluir rotas com enlaces de baixa qualidade no processo de seleção de rotas;
- Como uma métrica de roteamento cujo nível de sinal dos enlaces é considerado na função de custo da rota.

Goff & Abu-Ghazaleh (2003) propõem um esquema de roteamento preemptivo em redes *ad hoc*. Após a aferição da intensidade do sinal na recepção de pacote, uma região preemptiva é definida em torno dos nós. Enlaces de limiar inferior ao da região preemptiva são considerados ruins e serão desconsiderados no roteamento. O limiar P_{limiar}

é calculado como mostrado na Equação 2.1, onde P_O é uma constante dependente do ganho da antena para cada par transmissor-receptor e $r_{preemptiva}$ é o raio da região preemptiva.

$$P_{limiar} = P_O / r^4_{preemptiva} \tag{2.1}$$

Dube & Rais (1997) projetaram o protocolo SSAR - *Signal Stability-based Adaptive Routing* que faz uso de *beacons* periódicos de camada MAC para averiguar a qualidade dos enlaces. Em seguida o mecanismo de descoberta de rotas operará apenas com enlaces que apresentaram uma intensidade de sinal mínima.

Punnoose & Nikitin (1999) fazem um estudo sobre a relação da intensidade do sinal com a probabilidade de entrega de pacotes. Um dos resultados obtidos é mostrado na Figura 2.5, onde é possível observar que o aumento da intensidade do sinal afeta positivamente a probabilidade de entrega de pacotes, contudo a redução da intensidade do sinal não implica necessariamente em baixas probabilidades de entrega de pacotes. Zhao & Govindan (2003) faz um estudo similar em redes de sensores densas, chegando às mesmas conclusões.

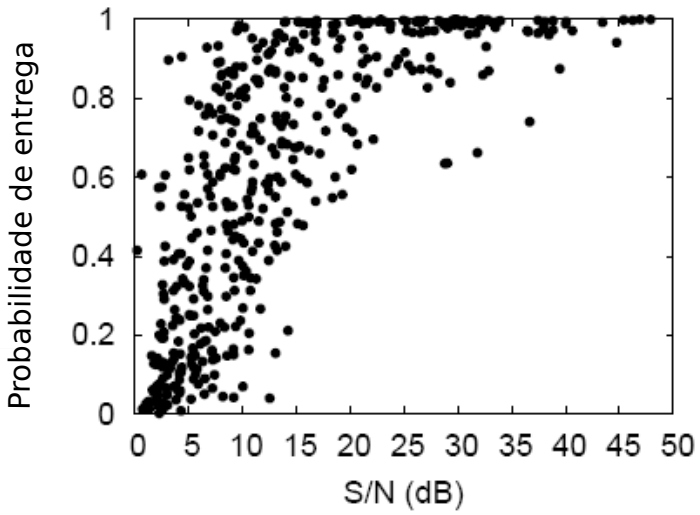

Figura 2.5: Probabilidade de entrega de pacotes *versus* médias de S/N
(PUNNOOSE; NIKITIN, 1999)

Mais adiante, Punnoose & Nikitin (1999) propõem um modelo preditivo de propagação em tempo real a fim de otimizar os protocolos de roteamento. Pesos são atribuídos aos enlaces de acordo com suas intensidades de sinal. Em uma rota formada por M saltos, o fator de qualidade L da rota é estimado pela Equação 2.2, onde $Q(x)$ é a função Q

padrão (*Q-function*), P_{pred_i} é a potência recebida pelo i^n nó a partir do $(i-1)^n$ nó, P_n é o limiar de recepção e σ é a variança dos sinais, que é assumida como uma distribuição normal.

$$L = \prod_{s=1}^{M}(1 - Q((P_{pred_i} - P_n)/\sigma)) \tag{2.2}$$

2.3.4.3 Métricas baseadas em Provas Ativas

Visto que o uso da intensidade do sinal para calcular a probabilidade de entrega de pacotes nem sempre é satisfatório, pode ser usada uma abordagem alternativa por meio de mensurações ativas e pacotes de prova para estimar diretamente essas probabilidades. Segundo Punnoose & Nikitin (1999), as provas ativas introduzem alguns obstáculos:

- Caso sejam usados pacotes de prova isolados, devem ter o mesmo padrão dos pacotes do tráfego normal para evitar que sejam priorizados ou tratados de modo diferente pela rede;
- No caso de pacotes de prova embutidos no tráfego normal (técnica chamada de mensuração intrusiva ou *in-band*), pode haver influência no volume do tráfego;
- O período ideal das aferições também é um dilema entre a acurácia da medida e o *overhead* de sinalização.

Os trabalhos envolvendo métricas baseadas em provas ativas em redes *ad hoc* são bastante promissores. As provas são feitas diretamente ou indiretamente, via inferência a partir da observação do tráfego local, em uma quantidade apropriada, e não dependem de assunções meramente analíticas. A principal métrica de prova ativa é a Expected Transmission Count (ETX). A ETX foi uma das primeiras métricas projetadas especificamente para as redes *ad hoc* e serviu de base para uma gama de outras métricas, como a Expected Transmission Time (ETT), a Medium Time Metric (MTM) e a Weighted Cumulative Expected Transmission Time (WCETT).

Após observar que minimização do número de saltos não é uma técnica otimizada para as redes *ad hoc*, Couto et al. (2005) propuseram a métrica ETX que leva em conta as taxas de perdas de pacotes de forma bidirecional. A ETX estima o número de

transmissões (e retransmissões) necessárias para enviar um pacote por cada enlace. Após isso tenta minimizar o número de transmissões para otimizar a vazão total da rede, o que também minimiza o consumo de energia e as interferências.

O número esperado de transmissões ETX é calculado como mostrado na Equação 2.3, onde d_f é a taxa esperada (ou probabilidade) de entrega de pacotes com sucesso no sentido direto do enlace e d_r é a taxa esperada no sentido reverso, ou seja, a probabilidade de receber uma confirmação de recebimento pacote, considerando, ainda, que cada transmissão não interfere na outra (como uma tentativa de Bernoulli). As taxas de entrega são medidas usando pacotes de prova em *broadcast* em nível MAC. Cada nó propaga seu pacote de prova a cada segundo, incluindo no pacote o número de pacotes de prova recebido de cada nó vizinho nos últimos w segundos (por padrão, $w = 10$). Consequentemente, cada nó vizinho de A poderá calcular o d_r de A toda vez que receber um pacote de prova de B.

$$ETX = 1/(d_r \times d_f) \tag{2.3}$$

A Figura 2.6 sumariza o processo de cálculo da ETX. O nó B relata, por meio do envio de pacote de prova em *broadcast*, o número x de pacotes recebidos de A no último período w. Assim o nó A pode estimar a probabilidade de um pacote de dados ser transmitido com sucesso em uma única tentativa que será $d_f = x/w$. Também pode calcular a probabilidade no sentido reverso, que será $d_r = y/w$, observando o número y de pacotes de prova recebido de A no mesmo período w. Com isso métricas ETX_A pode ser estimada.

Experimentos demonstraram que um protocolo de roteamento baseado na minimização do número de saltos, poderia dobrar a vazão da rede ao adotar a métrica ETX. A ETX é uma das únicas métricas não baseadas em número de saltos que foi incorporada na prática em redes *ad hoc*, por exemplo, como parte integrante da implementação OLSRD (OLSR *protocol daemon*) em diversas plataformas.

Outra métrica importante é a Per-hop Round Trip Time (RTT) que calcula o atraso em enlaces bidirecionais (ADYA et al., 2004). Para medir o valor de RTT, um pacote de prova contendo um indicador de tempo (ou *timestamp*) é enviado periodicamente para cada nó vizinho. Ao receber o pacote de prova o nó responde com outro pacote de prova, permitindo que o nó de envio calcule a RTT. A métrica de RTT computada para

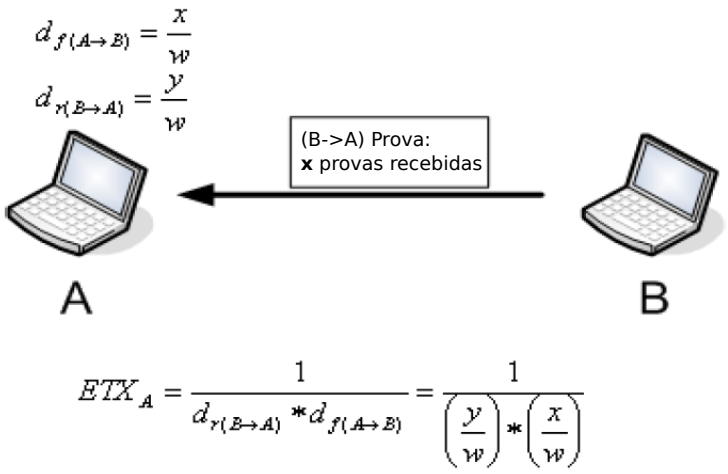

Figura 2.6: Estimativa da métrica ETX do nó A
(PARISSIDIS; KARALIOPOULOS, 2009)

a rota é simplesmente a soma dos RTTs dos enlaces que a compõem. A métrica RTT depende da carga da tráfego da rede, visto que ela acumula os tempos de enfileiramento, contenção do canal, bem como os atrasos com retransmissões em nível MAC.

2.3.4.4 Métricas baseadas na Mobilidade

As métricas baseadas em provas ativas também são adequadas para cenários de redes estáticas. Porém, em cenários móveis, são ineficientes, visto que, enquanto os nós se movem, os enlaces podem se romper ou podem se reconstruir, alterando a todo instante o conjunto de rotas ótimas.

Draves et al. (2004) afirmam que, como as métricas baseadas em provas ativas precisam de algum tempo para atualizar suas estimativas de qualidade dos enlaces, pode haver grande degradação do desempenho da rede quando as rotas se modificam durante a transferência dos dados. Afirmam também que, com métricas baseadas em minimização dos saltos, os novos enlaces devem ser usados assim que estejam disponíveis para evitar perda de pacotes.

As métricas baseadas na mobilidade visam escolher rotas que tenham o maior tempo de vida para minimizar o *overhead* relacionados às mudanças de rotas e os respectivos impactos na vazão da rede. Geralmente essas métricas observam as taxas de variação na intensidade do sinal para inferir sobre a estabilidade dos enlaces e das rotas. O grau de associação entre caminhos, proposto pelo Associativity-Based Routing (ABR) em (TOH, 1997b), e a métrica de afinidade, definida em (PAUL, 1999) e utilizada pelo protocolo Route-Lifetime Assessment Based Routing (RABR) em (AGARWAL;

AHUJA, 2000), são exemplos de métricas dessa categoria. Algumas técnicas para o cálculo da associação e estabilidade dos enlaces são discutidas a seguir.

Os nós móveis transmitem regularmente *beacons* de camada de enlace e computam o nível de associação, chamado de *ticks* de associação, com base no número de *beacons* recebidos dos demais nós. Os *ticks* de associação revelam a estabilidade atual dos enlaces. Valores de *ticks* de associação abaixo de um limiar A_{thr} implicam que os nós possuem alta mobilidade e, consequentemente, produzem enlaces pouco duradouros.

O grau de estabilidade de associação A^R_{avg} da rota R é estimado em função dos *ticks* de associação de todos os enlaces da rota como mostrado na Equação 2.4, onde l é uma função lógica, que avalia se o grau de associação do enlace é maior que o limiar, e n é o número de enlaces da rota R. No roteamento baseado na associação, as rota escolhida é a de maior grau de associação dos enlaces. Caso duas rotas possuam o mesmo grau de associação, a de menor número de saltos é escolhida.

$$A^R_{avg} = \frac{1}{n} \sum_{l \in R} l_{A_1 \geq Athr} \tag{2.4}$$

Propõem outra métrica de roteamento que define a probabilidade da disponibilidade dos enlaces sujeitos a falhas em decorrência da mobilidade. Para isso é tomado como base um modelo de movimentação aleatória. Cada nó é caracterizado por valores que descrevem uma distribuição estatística da média e da variância da velocidade em um intervalo de tempo e em um raio de comunicação. Eles derivam uma função sofisticada para estimar a probabilidade da disponibilidade de um enlace.

2.4 PROTOCOLO DYNAMIC SOURCE ROUTING

O Dynamic Source Routing (DSR) é um protocolo de roteamento simples e eficiente, desenvolvido especialmente para redes *ad hoc* uniformes com suporte a múltiplos saltos. Sua especificação original bem como as suas extensões podem ser encontradas na RFC 4728 e no artigo original de seus criadores em (JOHNSON et al., 2001b). Quando em execução, a rede se torna auto-organizável e autoconfigurável, dispensando infraestruturas pré-definidas e entidades de administração central. A comunicação em múltiplos saltos é possível por meio da cooperação entre os nós no encaminhamento de pacotes aos locais mais distantes.

Dentre os objetivos do seu projeto, procurou-se reduzir ao máximo a sobrecarga de pacotes de controle e proporcionar uma rápida resposta às mudanças na rede por meio de uma abordagem reativa (JOHNSON et al., 2001a). Mesmo que exista bastante mobilidade e que os nós entrem e saiam da rede a todo momento, seu algoritmo busca e atualiza automaticamente as informações de roteamento quando solicitado.

O aspecto mais interessante do seu funcionamento é que os pacotes de dados carregam consigo o caminho completo de roteamento da origem até o destino em uma estrutura de dados denominada de *Source Route*. Consequentemente, os nós intermediários não participam nas decisões de roteamento, mas apenas encaminham para o próximo nó conforme foi designado. Essa abordagem é vantajosa para redes *ad hoc*, visto que é inerentemente livre de *loops* e dispensa processamento extra nos demais nós.

Apesar do protocolo DSR ser pouco robusto em termos de desempenho e escalabilidade em comparação com outros protocolos reativos e proativos como pode ser verificado em (THAKARE, 2010) e (AHUJA et al., 2000), despertou o interesse desta pesquisa devido a alguns benefícios para a implementação em ambientes de computação ubíqua. O primeiro benefício é a capacidade de guardar no *cache* de rotas mais de uma alternativa de rota para um mesmo destino (JOHNSON et al., 2001b). Isso ocorre para otimizar o mecanismo de manutenção de rotas.

Sempre que um nó escuta mensagens em *broadcast* de RREP com um novo caminho para um determinado destino, guarda no seu *cache*, possibilitando o uso futuro ao enfrentar uma quebra de enlace. Outro benefício é a boa operação em ambientes dinâmicos com alto grau de mobilidade e baixa cooperação, que é uma característica dos ambientes de computação ubíqua.

2.4.1 Premissas do Protocolo Dynamic Source Routing

O projeto do DSR estabelece algumas premissas, conforme mostrado em (JOHNSON et al., 2001a):

- Todos os nós estão sempre dispostos a cooperar no encaminhamento de pacotes e respeitam plenamente as regras estabelecidas pelo protocolo;
- Opera em uma rede *ad hoc* pequena com diâmetro entre 1 e 10 saltos. O termo *diâmetro* se refere ao número de saltos exigido para se atingir o limite extremo da rede (nó mais afastado);

- Os pacotes podem ser perdidos ou corrompidos durante a transmissão, devendo o destino descartá-los se forem detectados erros;

- Os nós da rede podem se mover livremente, sem qualquer aviso, em velocidades variáveis ou contínuas. As velocidades devem ser moderadas para não comprometer a capacidade de transmissão e a latência suportadas pelo *hardware* de rede;

- Os nós são capazes de habilitar o modo promíscuo da interface de rede, possibilitando que o *hardware* envie todos os pacotes ao *driver* de rede, sem ocorrer filtragem com base nos endereços da camada de enlace (MAC);

- A comunicação sem fio entre os nós pode ser bidirecional ou unidirecional. Porém algumas otimizações só são suportadas em enlaces bidirecionais, como, por exemplo, a reversão de rotas. Os protocolos MAC mais comuns, como o MACA [11], MACAW [12] e o 802.11, operam com enlaces bidirecionais;

- Cada nó deve escolher um único endereço IP que será o seu identificador na rede *ad hoc*. Mesmo que existam várias interfaces de rede instaladas, uma deve ser escolhida para interação com os demais nós.

2.4.2 Mecanismos de Descoberta e Manutenção de Rotas

O protocolo DSR é formado por dois mecanismos que trabalham em conjunto para permitir a conectividade em uma rede *ad hoc* (JOHNSON et al., 2001a):

- **Descoberta de Rotas:** ocorre quando um nó A deseja enviar pacotes para um nó E, porém A desconhece uma rota para chegar em E. Para isso é iniciado o processo de descoberta a fim de obter uma rota para o nó E.

- **Manutenção de Rotas:** possibilita que o nó A seja avisado sobre a ocorrência de quebra de enlaces usando determinado rota com o destino E. Nesse caso, o nó A poderá usar outra rota que conheça ou poderá disparar novamente o mecanismo de Descoberta de Rotas.

[11] MACA (Multiple Access with Collision Detection) é um protocolo de acesso ao meio usado em redes sem fio que evita colisões causadas pelos problemas de terminal exposto e terminal escondido (KARN, 1990). Simplificando, o nó emite um aviso antes de enviar os quadros fazendo com que os demais tenham que aguardar.

[12] MACAW (Multiple Access with Collision Avoidance for Wireless) é um protocolo de acesso ao meio concebido para as redes *ad hoc* (BHARGHAVAN; DEMERS, 1994). Seu mecanismo resolve o problema de terminal escondido, sendo incorporado em outros protocolos, como o IEEE 802.11.

Tanto o mecanismo de descoberta de rotas quanto o de manutenção de rotas somente são invocados quando existe a comunicação entre A e E, reforçando a abordagem reativa do protocolo. Logo o DSR não requer qualquer tipo de pacote de controle periódico, tais como para avisos de rotas, estado de enlaces, detecção de nós vizinhos, controle de topologia, etc.

Quando a rede *ad hoc* é relativamente estática - a mobilidade existe, mas mantém um padrão que preserve a topologia -, o número de pacotes de controle tende a zerar. Em ambientes mais dinâmicos, por sua vez, o protocolo escalará em relação a sobrecarga de mensagens de controle, visto que estas serão geradas exclusivamente para resolver as rotas em uso.

Um efeito importante do mecanismo de descoberta de rotas, devido ao uso de mensagens em *broadcast*, é que permite que os nós vizinhos aprendam novas rotas para os destinos, poupando mensagens futuras de descoberta de rotas. Essa capacidade de armazenar novas rotas no *cache* de rotas para um mesmo destino foi um aspecto importante desta pesquisa, pois possibilitou a investigação de novas métricas para escolha de rotas, visto que a implementação padrão do protocolo DSR prioriza o uso de rotas com menor número de saltos.

Os algoritmos de descoberta e manutenção de rotas foram desenhados para suportar tanto enlaces unidirecionais, quanto bidirecionais, permitindo a presença de rotas assimétricas entre um par de nós. Isso é possível graças a possibilidade de um nó detectar que um caminho reverso foi interrompido e iniciar um novo processo de descoberta de rotas ou, ainda, usar um rota alternativa presente em seu *cache* de rotas.

2.4.2.1 Funcionamento do Mecanismo de Descoberta de Rotas

Quando um nó A envia um novo pacote ao nó de destino E, ele adiciona no cabeçalho do pacote uma *Source Route* que carrega a sequência de saltos que o pacote deve seguir para alcançar E. Geralmente um *Source Route* é obtido a partir de uma consulta no *cache* de rotas, que é uma estrutura de dados especial de armazenamento das rotas aprendidas, mas, caso não encontre uma rota para o destino, ele inicia o processo de descoberta de rotas para encontrar uma nova rota para E.

A Figura 2.7 ilustra o processo de descoberta de rotas, no qual o nó A tenta se comunicar com o nó E. Inicialmente, o nó A envia uma mensagem *Route Request* (RREQ) em *broadcast*. Todos os nós que estão ao alcance de A receberão a mensagem. As

mensagens RREQ carregam o endereço da origem e do destino, um identificador único (ID), que foi determinado pelo nó A, e um registro que armazenará a lista de endereços dos nós intermediários que repassarão a mensagem RREQ.

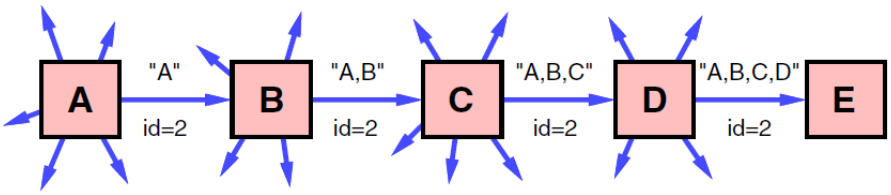

Figura 2.7: Exemplo de descoberta de rotas
(JOHNSON et al., 2001b)

Quando um nó qualquer recebe a mensagem RREQ, ele primeiramente verifica se é o destino E da descoberta de rotas. Em caso positivo, ele envia uma mensagem *Route Reply* (RREP) contendo uma cópia do registro que armazena os endereços por onde a mensagem passou. Quando o nó A recebe o RREP, ele armazena o registro como uma *Source Route* no seu *cache* de rotas para ser usado na transmissão de pacotes de dados ao destino E. Caso o nó intermediário constate que já recebeu a RREQ, comparando o *ID*, ou que foi inserido no registro de endereços, ele descartará o pacote. Caso contrário, o nó intermediário adiciona seu endereço no registro e repassa a RREQ, preservando o ID, por meio de *broadcast* local.

Para enviar a RREP, o nó E, mostrado na Figura 2.7, consulta primeiramente o seu *cache* de Rotas em busca de uma rota para o A e, se encontrar, a utiliza no envio do pacote. Se não possuir uma rota, ele pode disparar o processo de descoberta de rotas, mas a fim de evitar recursividade infinita, deve transportar a mensagem RREP anexada ao pacote de RREQ para A. Por fim, se forem usados enlaces bidirecionais e protocolos MAC com esse suporte, como no caso do IEEE 802.11, o nó E poderia simplesmente inverter a rota recebida no RREQ.

O nó de origem pode receber vários pacotes de RREP para um mesmo destino, sendo necessário escolher uma das rotas a partir de algum critério como o número de saltos, atraso, etc. Tipicamente é escolhida a rota com menor número de saltos para determinado destino. Todos os nós que participam no encaminhamento dos pacotes ou que apenas escutam os demais nós também podem armazenar as informações de rotas em seu *cache* de rotas, visando um uso futuro.

Para a manutenção da rota, caso os nós intermediários descubram que uma rota foi quebrada, enviam uma mensagem *Route Error* (RERR) usando o caminho inverso até

aquele ponto. Os nós que recebem a RERR retiram aquele caminho do *cache* de rotas e o nó de origem deverá enviar uma nova RREQ para descobrir uma rota alternativa.

Antes de iniciar a descoberta de rotas, o nó de origem armazena os pacotes de dados, que estão aguardando por uma *Source Route*, em um *buffer* interno chamado de *Send Buffer*. No *Send Buffer* os pacotes de dados recebem uma marcação de tempo para possibilitar o descarte após certo período, evitando estouro da área. É possível ocorrer novos processos de descoberta de rotas para o mesmo destino enquanto os pacotes aguardam no *Send Buffer*, mas o número é limitado, pois o nó de destino pode estar em uma região particionada da rede e não existir uma sequência de saltos que o alcance. A ocorrência de partições na rede dependerá da densidade da rede e dos padrões de movimentação dos nós.

2.4.2.2 Funcionamento da Manutenção de Rotas

Quando um nó envia algum pacote usando determinado *Source Route*, ele tem a responsabilidade de verificar se foi recebido pelo próximo salto. Caso não possa confirmar a recepção, poderá fazer retransmissões (em um número limitado) até que seja recebido. Na Figura 2.8, por exemplo, o nó A deve verificar se o pacote foi recebido por B, já o nó B deve verificar a recepção do nó C e assim por diante.

Essa verificação não exige mensagens de controle adicionais por parte do DSR, pois pode ser utilizado, por exemplo, as mensagens de *acknowledgement* em nível MAC existentes no IEEE 802.11 (MAN; SOCIETY, 2007). Outra alternativa é usar a técnica de *acknowledgement* passivo, proposta em (JUBIN; TORNOW, 1987), por meio do qual o nó B poderia confirmar a recepção do pacote em C averiguando que o pacote foi retransmitido para D. Caso nenhuma confirmação seja possível, o DSR ainda prevê uma mensagem de verificação (*Acknowledgement Request*) que pode ser solicitada pelo nó transmissor.

Caso o número máximo de retransmissões de pacotes de dados seja atingido sem que seja recebida qualquer confirmação, o nó deverá retornar uma mensagem RERR ao nó de origem com a identificação do enlace que foi rompido. Por exemplo, na Figura 2.8, se C não conseguir transmitir o pacote para D, então C deve retornar uma mensagem RREP para o nó de origem A, informando que o enlace entre C e D foi rompido.

Em seguida, o nó A remove a rota quebrada do seu *cache* de rotas e poderá escolher outra rota proveniente de descobertas de rota anteriores ou da observação das men-

sagens de roteamento em *broadcast*. Se não possuir uma rota, deve iniciar um novo processo de descoberta de rotas em busca de nova rota.

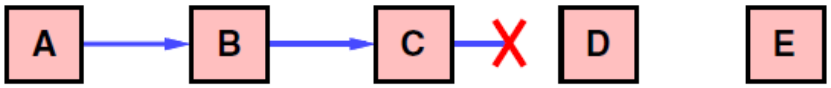

Figura 2.8: Exemplo de manutenção de rotas
(JOHNSON et al., 2001b)

2.4.3 Recursos Adicionais do Dynamic Source Routing

Após o projeto original do DSR, seus criadores e outros pesquisadores propuseram alguns recursos adicionais que, mais tarde, foram incorporados à especificação da RFC 4728 (JOHNSON et al., 2001a). O objetivo desses novos recursos foi melhorar o desempenho dos mecanismos de descoberta e manutenção de rotas nos aspectos de redução da sobrecarga de mensagens, aumento da largura de banda disponível para os pacotes de dados e redução do tempo de convergência de controle topológico. As subseções que seguem resumem o funcionamento desses recursos.

2.4.3.1 *Caching* das Informações de Roteamento

Para aprimorar o processo de descoberta de rotas, os nós são capazes de armazenar novas rotas no *cache* de rotas a partir da observação dos pacotes que são difundidos pela rede. As informações de roteamento podem ser obtidas em pacotes de dados, pois carregam uma *Source Route* válida em mensagens RREQ e RREP.

Este recurso fica limitado quando existem apenas enlaces unidirecionais, conforme relatado em (JOHNSON et al., 2001b), visto que só haverá *caching* de rotas no sentido do fluxo dos pacotes de dados e mensagens RREQ e RREP. A Figura 2.9 exemplifica uma situação em que ocorrem tais limitações. O nó A usa uma determinada *Source Route* para se comunicar com E. Como o nó C repassa os pacotes entre A e E, ele pode aprender sobre os caminhos no sentido direto do roteamento, porém não pode deduzir nada sobre o retorno, visto que o enlace é unidirecional.

Caso similar ocorre quando o nó C escuta o repasse de pacotes entre o nó X e o nó Y em um enlace unidirecional contendo uma *Source Route* diferente. Todavia, se os enlaces fossem bidirecionais, o nó C já poderia assimilar as rotas entre o nó E e o nó A e, do mesmo modo, entre o nó Z e o nó V.

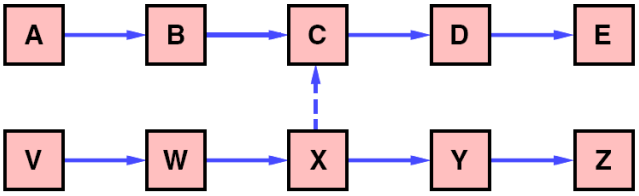

Figura 2.9: Limitação do *caching* de informações de roteamento em enlaces unidirecionais. (JOHNSON et al., 2001b)

2.4.3.2 Envio de Mensagens *Route Reply* por Nós Intermediários

Quando os nós intermediários recebem uma mensagem RREQ, devem verificar se existem rotas para o alvo pretendido em seu *cache* de Rotas, antes de reencaminhar a solicitação. Caso localizem uma rota para o alvo, devem concatenar o caminho acumulado na RREQ com o trecho da rota presente em seu *cache* de rotas e providenciar a RREP para a origem.

No processo de concatenação de caminhos, a duplicação de nós deve ser evitada. A Figura 2.10 apresenta uma situação em que o nó C conhece a rota entre o nó A e o nó F e entre o nó F e o nó E. Dessa forma pode aprender o caminho entre o nó A e o nó E, porém deve remover o nó F do caminho ao criar a RREP. Um primeiro benefício dessa restrição é o aumento da probabilidade da rota resultante ser válida, pois é possível que o nó F receba uma RERR.

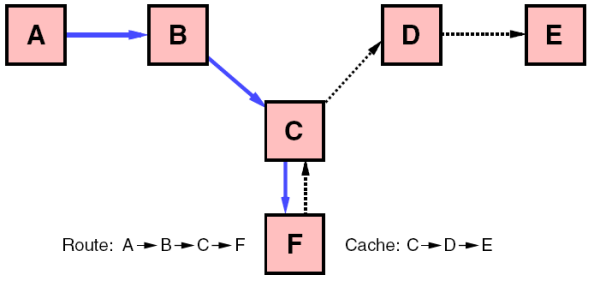

Figura 2.10: Possível duplicação de nós em caminhos concatenados no processo de criação RREP.

2.4.3.3 Prevenção contra Tempestades de *Route Reply*

Devido à capacidade dos nós intermediários responderem mensagens RREQ, é possível que ocorra uma "tempestade"de mensagens RREP. Caso determinado nó emita uma RREQ em *broadcast* e seus nós vizinhos possuam rotas para o destino em seus *caches* de rotas, cada nó tentará retornar uma RREP, provocando uma redução da largura de banda disponível e aumentando a probabilidade de colisões naquela região da rede.

A Figura 2.11 mostra uma situação em que os nós B, C, D, E e F recebem uma RREQ para o nó de destino G. Pode-se observar que cada nó possui uma alternativa de rota em seu *cache* de rotas de tamanhos variados (C-B-G, B-G, G, etc).

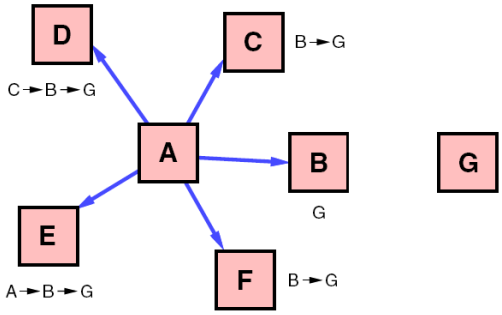

Figura 2.11: Exemplo de tempestade de mensagens *Route Reply*.
(JOHNSON et al., 2001b)

Este problema é resolvido com a escolha de tempos de atraso aleatórios antes do envio das mensagens RREP. Como os nós colocam suas interfaces em modo promíscuo, podem escutar o meio e verificar se outro nó já encaminhou uma RREP para o destino, livrando a rede mensagens desnecessárias.

A Equação 2.5 apresenta o cálculo do tempo de atraso aleatório d, onde h é o número de saltos a ser retornado na RREP, r é um valor aleatório entre 0 e 1 e H é uma constante calculada com base no atraso máximo de propagação do enlace multiplicada pelo número de saltos da RREP.

$$d = H \times (h - 1 + r) \tag{2.5}$$

Caso um nó receba mensagens RREP com um valor h menor que o seu, pode concluir que uma rota melhor foi enviada ao nó de origem. Neste caso, encerra o processo de espera e não envia a RREP. Além disso, a constante H otimiza o recurso de prevenção, pois aumenta a chance das melhores mensagens RREP serem enviadas antecipadamente.

2.4.3.4 Limitador do Número de Saltos em Mensagens *Route Request*

Cada RREQ possui um limitador do número de saltos que é utilizado para restringir o número de nós intermediários que poderão repassar a mensagem. A medida que a RREQ é repassada, o valor deste limitador é decrementado até que se atinja o valor

zero - momento em que a mensagem é descartada pelos nós intermediários. Uma das aplicações deste recurso é a possibilidade de consultar rotas apenas nos nós vizinhos.

Por exemplo, ao se limitar o número de saltos em 1, o nó de origem pode verificar apenas em sua vizinhança se existe uma rota para determinado destino (os nós vizinhos decrementarão o limite para 0). Esse recurso faz com que o *cache* de rotas dos nós vizinhos seja na verdade uma extensão do *cache* de rotas do nó de origem. Caso não seja recebida uma RREP, o nó de origem poderá em seguida dispensar o recurso de limitação de saltos e difundir a mensagem RREQ por toda a rede.

Outra aplicação da limitação de saltos, é a possibilidade de se fazer uma consulta incremental por rotas chamada de busca *expanding ring* em (JOHNSON et al., 2001a). Por exemplo, um nó poderia enviar uma RREQ inicial não propagável, e caso nenhuma RREP seja recebida, o nó poderia criar uma nova RREQ com o limite de saltos igual a 1. Progressivamente será possível explorar outras vizinhanças em busca de rotas para o alvo até que se propague por toda a rede. Contudo esse mecanismo de busca incremental aumenta a latência do mecanismo descoberta de rotas.

2.4.3.5 Resgate de Pacotes em Rotas Quebradas

Após enviar uma mensagem RERR, o nó que verificou o enlace quebrado pode tentar resgatar o pacote de dados antes de descartá-lo. Para isso, o nó pesquisa no seu próprio *cache* de rotas a fim de obter uma *Source Route* para o destino pretendido. Se for encontrada uma rota, o nó cancela a mensagem RERR e substitui a rota original pela rota do seu *cache* de rotas. Em seguida encaminha o pacote para o próximo nó indicado na *Source Route*.

Quando um pacote é resgatado dessa forma, é colocada uma marca especial para prevenir que um único pacote seja recuperado várias vezes por diferentes nós. Um mecanismo alternativo para recuperação de pacotes seria substituir apenas o sufixo inutilizado da rota original com a nova rota encontrada no *cache* de rotas, ou seja, uma rota obtida a partir da concatenação do prefixo da rota original com o sufixo da rota presente no *cache* de rotas. Esse mecanismo previne que o pacote seja retransmitido mais uma vez pelos nós intermediários. Ao concatenar as rotas devem ser tratadas as duplicações de nós para prevenir a ocorrência de *loops*.

2.4.3.6 Redução Automática de Rotas

As rotas podem ser reduzidas automaticamente caso os saltos intermediários se tornem desnecessários. Como os nós escutam o meio, devido à atuação em modo promíscuo, podem examinar as partes inutilizadas das rotas. Por exemplo na Figura 2.12, o nó C percebe que pode reduzir a rota (A-B-C-D) para apenas (A-C-D), pois consegue escutar a transmissão do pacote de A para B. Nesse caso o nó C envia uma mensagem RREP gratuita para o nó de origem A, informando a opção de rota mais curta (A-C-D) que será usada no envio dos próximos pacotes.

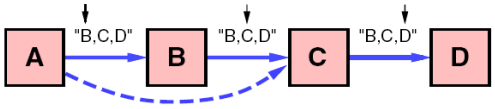

Figura 2.12: Exemplo de redução automática de rota.
(JOHNSON et al., 2001b)

2.4.3.7 Aumento da Propagação de Mensagens *Route Error*

Quando o nó de origem recebe uma mensagem RERR deverá propagar essa informação aos vizinhos, embutindo a mensagem RERR na nova mensagem RREQ. Dessa forma, os demais nós removerão a rota inválida dos seus *caches* de rotas e deixarão de enviar mensagens RREP com essa informação de rota.

Por exemplo no cenário da Figura 2.8, o nó A aprende que o enlace entre C e D foi quebrado a partir da mensagem RERR recebida de C. Em seguida o nó A remove essa rota do seu *cache* de rotas e inicia um novo processo de descoberta de rotas, caso não conheça uma rota alternativa, e divulga a RERR na própria mensagem RREQ. Outra forma de propagar mensagens RERR seria enviando a mensagem de erro pela mesma rota que resultou no enlace quebrado a fim de possibilitar que o nó que enviou a mensagem RREP seja atualizado. Dessa forma não enviará mais aquela opção de rota problemática.

2.4.3.8 Armazenamento de Informações Negativas

O registro de falhas nos enlaces pode ser uma informação de grande valia para aprimorar o protocolo DSR em algumas situações relatadas em (MALTZ et al., 1999). Ao receberem mensagens RERR, os nós podem preservar as informações negativas sobre os enlaces em uma estrutura de dados a parte, ao invés de apenas remover a rota do *cache* de rotas. Assim em futuras descobertas de rotas, os enlaces problemáticos podem ser

evitados. Outras informações negativas relevantes são as ocorrências de oscilações nos sinais dos enlaces por conta do alcance de rádio entre os nós.

Todavia é necessário atribuir um tempo de expiração das informações negativas para permitir que enlaces remediados sejam novamente incluídos no *cache* de rotas dos nós. Por exemplo na Figura 2.8, se o nó A preservar a informação sobre o enlace rompido entre C e D, poderia filtrar apenas mensagens *RREP* que não apresentem o enlace C-D durante um tempo determinado. Os criadores do DSR afirmam que uma lacuna na implementação desse recurso é o cálculo de um período de expiração adequado para os diversos cenários de rede (MALTZ et al., 1999).

2.4.3.9 Extensão de Estado de Fluxo

O DSR permite uma extensão chamada de Estado de Fluxo que reduz a sobrecarga dos pacotes, mas ainda mantém o funcionamento básico do protocolo. O objetivo do Estado de Fluxo é livrar os pacotes do transporte de uma *Source Route* em toda comunicação, substituindo a rota por apenas um identificador do fluxo (ID do fluxo). Um fluxo seria uma combinação do endereço da origem, endereço do destino e do ID do fluxo.

O nó de origem é o responsável pela definição do ID do fluxo e do tempo de expiração do fluxo que será usado pelos demais nós. O mapeamento do ID do fluxo para uma *Source Route* será feito localmente por uma tabela de fluxos em cada nó intermediário assim que o primeiro pacote com a rota é transmitido. Após os estabelecimento de um fluxo, os pacotes poderão apenas informar o ID do fluxo no cabeçalho do DSR até que se atinja o período de expiração.

2.4.4 Formato dos Cabeçalhos do Dynamic Source Routing

O protocolo DSR faz uso de um cabeçalho especial chamado de *DSR Options* que armazena as informações de controle e pode ser incluído em qualquer pacote IP. O cabeçalho *DSR Options* possui uma parte estática com 4 octetos, mostrada na Figura 2.13, e pode ser seguido por outros cabeçalhos *DSR Options*.

O *DSR Options* deve ser inserido logo após o cabeçalho IP, antes do inicio do cabeçalho do protocolo de transporte. Para o *DSR Options* ser considerado na pilha TCP/IP, deve ser informado o número 48 no campo protocolo do cabeçalho IP conforme a regulamentação da Internet Assigned Numbers Authority (IANA, 2012). As subseções

que seguem exploram os detalhes das estruturas de dados das mensagens usadas pelo protocolo DSR.

2.4.4.1 Estrutura do Cabeçalho *DSR Options*

O cabeçalho *DSR Options* possui uma parte fixa com 4 octetos que é carregada em todas mensagens do DSR. O formato da parte fixa é apresentado na Figura 2.13. O primeiro octeto se refere ao campo `Próximo Cabeçalho` que tem a função de identificar o tipo de cabeçalho que virá logo em seguida do cabeçalho *DSR Options*. O nono bit é um *flag* que deve ter o valor 0 nos cabeçalhos do tipo *DSR Options* e o valor 1 nos cabeçalhos do tipo *DSR Flow State*). Os próximos 7 bits são reservados com valor 0 e ignorados pelo protocolo. O campo `Tamanho` de 16 bits deve informar o tamanho do cabeçalho *DSR Options* desconsiderando a parte fixa.

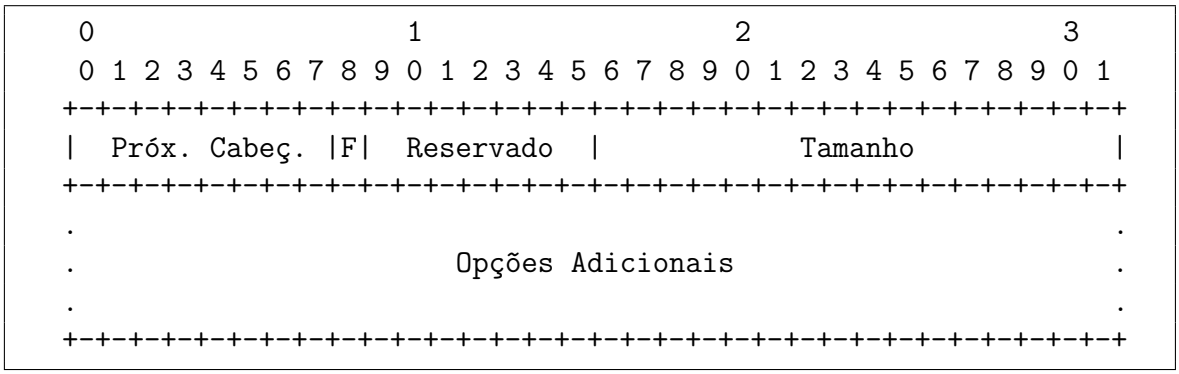

Figura 2.13: Cabeçalho DSR Options

Após a parte fixa do *DSR Options* segue o campo `Opções Adicionais` que possui tamanho variável e especificado pelo campo `Tamanho`. As opções adicionais são sequências de outros tipos de *DSR Options*, tais como: *Route Request Option*, *Route Reply Option*, *Route Error Option*, *Acknowledgement Request Option*, *Acknowledgement Option* e *Source Route Option*. As subseções seguintes explicarão os formatos destes tipos de *DSR Options*.

2.4.4.2 Estrutura do Cabeçalho *Route Request Option*

Uma mensagem RREQ requer um cabeçalho do tipo *Route Request Option* conforme apresentado na Figura 2.14. O campo `Identificação` armazena o ID único que evita a ocorrência de *loops* e retransmissões. O campo `Tipo` é responsável pela identificação do tipo de *DSR Options* e está presente nas demais opções. O campo `Tamanho` especifica o tamanho total do cabeçalho, excluindo-se o campo `Tipo` e `Tamanho`. O campo

`Endereço do Destino` armazena o endereço do nó de destino e os campos `Endereço[n]` armazenam os endereços dos nós que encaminharam o pacote de forma cumulativa.

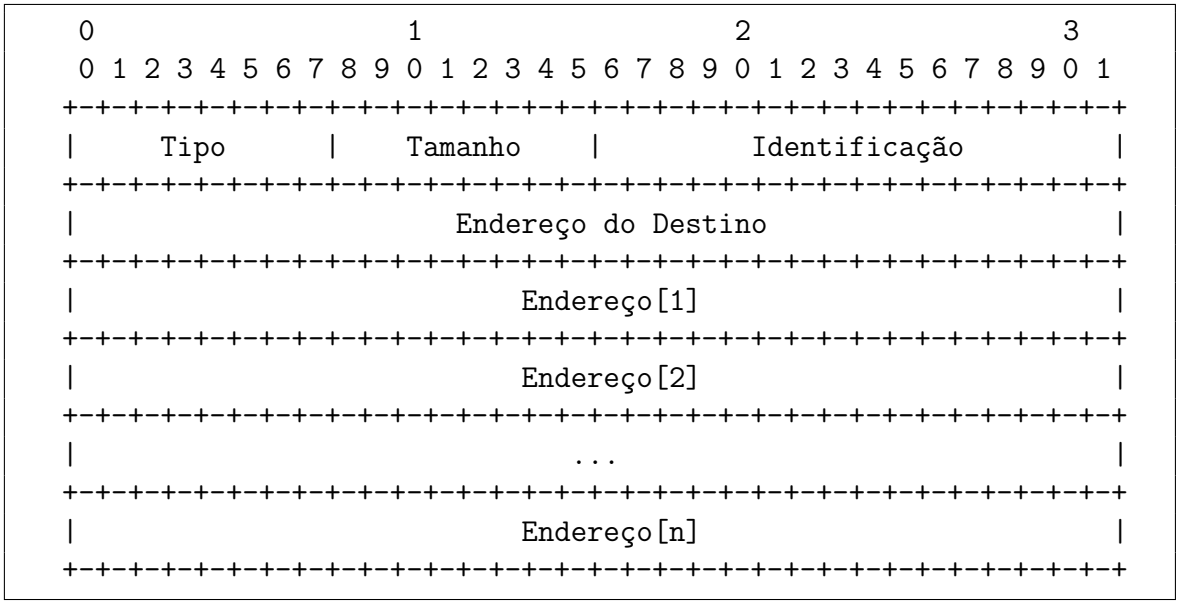

Figura 2.14: Estrutura do cabeçalho *Route Request Option*

Para que as mensagens do DSR trafeguem pela rede é necessário especificar os campos adequados do cabeçalho IP. O campo `Endereço de Origem` deve receber o endereço IP do nó de origem, o campo `Endereço de Destino` deve ser ajustado para o endereço de *broadcast* (255.255.255.255) e o campo `TTL` pode ser variado entre os valores 1 e 255 para implementar, por exemplo, as buscas incrementais de rotas.

2.4.4.3 Estrutura do Cabeçalho *Route Reply Option*

As mensagens RREP também exigem um tipo especial de cabeçalho - o *Route Reply Option* - mostrado na Figura 2.15.

Os campos `Tipo`, `Tamanho` e `Reservado` são especificados de forma similar aos demais cabeçalhos do *DSR Options*. O campo `L` é um *flag* que serve para informar que o último salto foi realizado fora da rede do DSR, permitindo que os nós evitem o uso de rotas com essa característica no futuro. A sequência de campos `Endereço[n]` carrega o *Source Route* que será sugerido ao nó de origem.

Em relação ao cabeçalho IP, o campo `Endereço de Origem` deve ser o endereço IP do nó que gerou a mensagem RREP. O campo `Endereço de destino` deve conter o endereço IP do nó de origem obtido da mensagem RREQ.

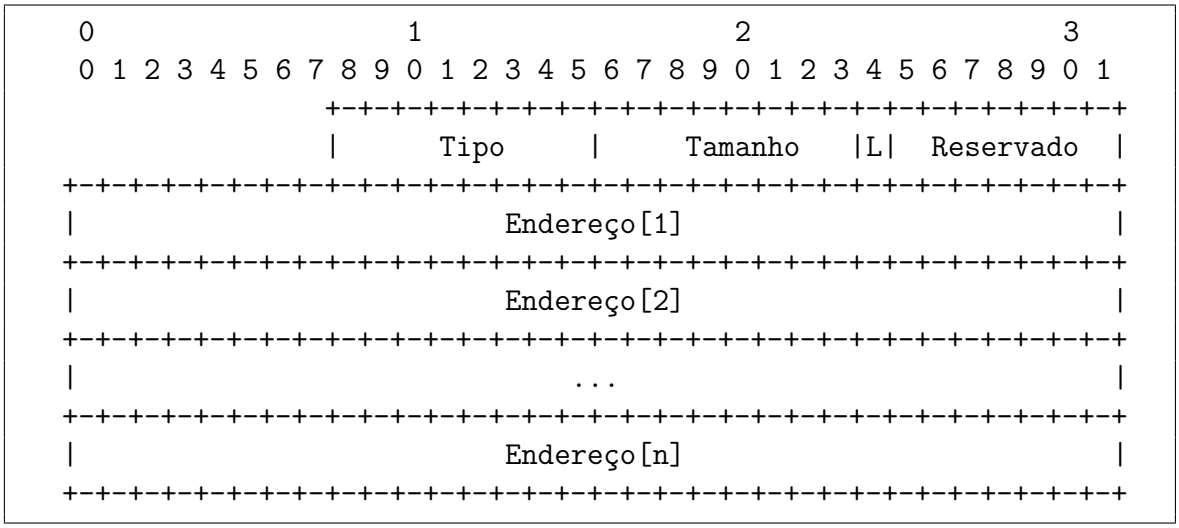

Figura 2.15: Estrutura do cabeçalho *Route Reply Option*

2.4.4.4 Estrutura do Cabeçalho *Source Route Option*

Os pacotes de dados carregam a *Source Route* contendo o caminho completo da origem até o destino no cabeçalho *Source Route Option* mostrado na Figura 2.16.

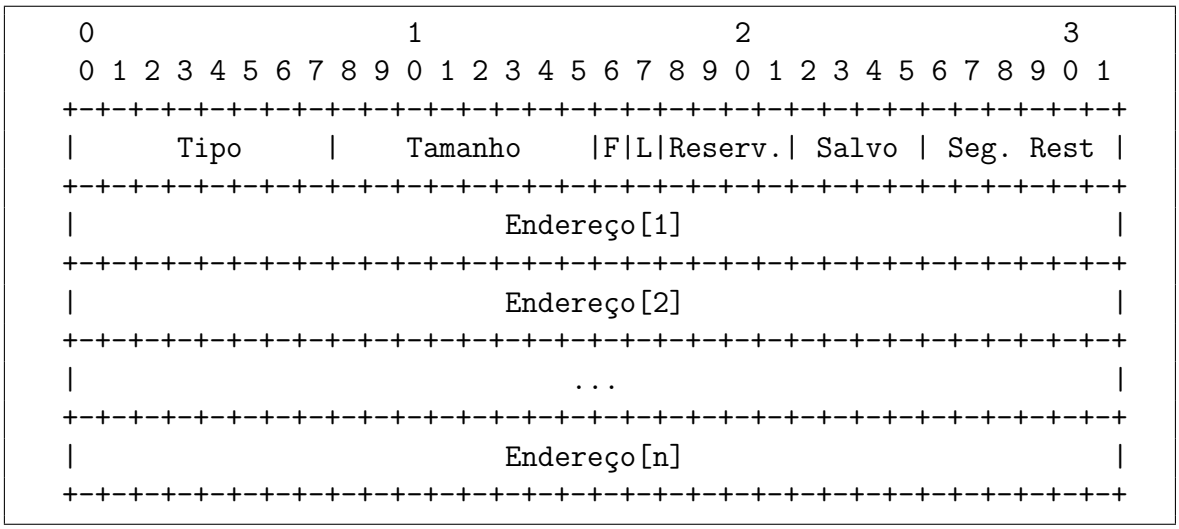

Figura 2.16: Estrutura do cabeçalho *Source Route Option*

Os campos `Tipo`, `Tamanho` e `Reservado` são similares aos demais cabeçalhos. Os *flags* `F` e `L` servem para informar o primeiro e o último salto ocorrido fora das fronteiras da rede DSR, respectivamente. O campo `Salvo` é um contador que armazena a quantidade de vezes que o pacote foi resgatado de um enlace quebrado. O campo `Segmentos Restantes` informa o número de trechos da rota que o pacote ainda deve percorrer para que alcance o seu destino.

2.4.4.5 Estrutura do Cabeçalho *Route Error Option*

Ao detectar uma quebra de enlace o nó gera uma mensagem RERR que possui o cabeçalho *Route Error Option*, conforme a Figura 2.17.

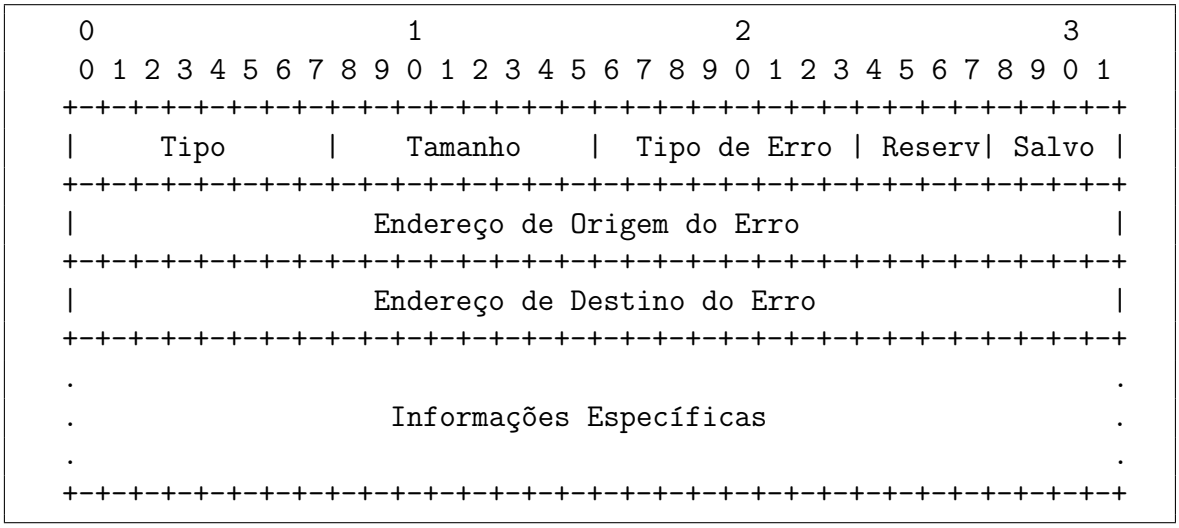

Figura 2.17: Estrutura do cabeçalho *Route Error Option*

Os campos `Tipo`, `Tamanho` e `Reservado` são similares aos demais cabeçalhos. O campo `Tipo de Erro` especifica um dos seguintes valores de erro: 1 - nó não alcançável; 2 - estado de fluxo não suportado; e 3 - opção não suportada. O campo `Salvo` tem a função de informar quantas vezes o pacote foi resgatado após quebras de enlaces. O campo `Endereço de Origem do Erro` indica o endereço do nó que detectou a falha no enlace e o campo `Endereço de Destino do Erro` indica o endereço do nó que deverá receber a mensagem RERR (nó que sugeriu a rota com erro).

No caso de erro do tipo 1 (nó não alcançável), o endereço do nó que deixou de receber o pacote deve ser colocado no campo `Informações Específicas`. Com isso, os nós que receberem a mensagem RERR podem usar o endereço do nó não alcançado e a informação do campo `Endereço de Origem do Erro` para determinar o enlace quebrado e removê-lo das entradas do seus *caches* de rotas.

2.4.4.6 Estrutura do Cabeçalho *Acknowledgement Option* e *Acknowledgement Request Option*

O protocolo DSR conta com mensagens de *acknowledgement* via *software* para o caso em que não seja possível usar as técnicas de *acknowledgement* passivo ou *acknowledgement* de camada MAC. As mensagens de *acknowledgement* requerem os cabeçalhos

do tipo *Acknowledgement Request Option* e *Acknowledgement Request* mostrados nas Figuras 2.18 e 2.19, respectivamente.

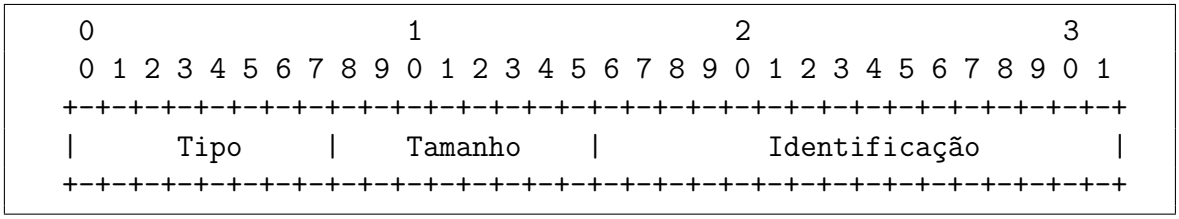

Figura 2.18: Estrutura do cabeçalho *Acknowledgement Request Option*

Os campos `Tipo` e `Tamanho` são similares aos demais tipos de cabeçalhos. O campo `Identificação` armazena um número aleatório determinado pelo nó que deseja verificar o enlace. O campo `Endereço de Origem do ACK` indica o endereço do nó que responde ao pedido de *acknowledgement* e o campo `Endereço de Destino do ACK` indica o endereço do nó que fez a solicitação.

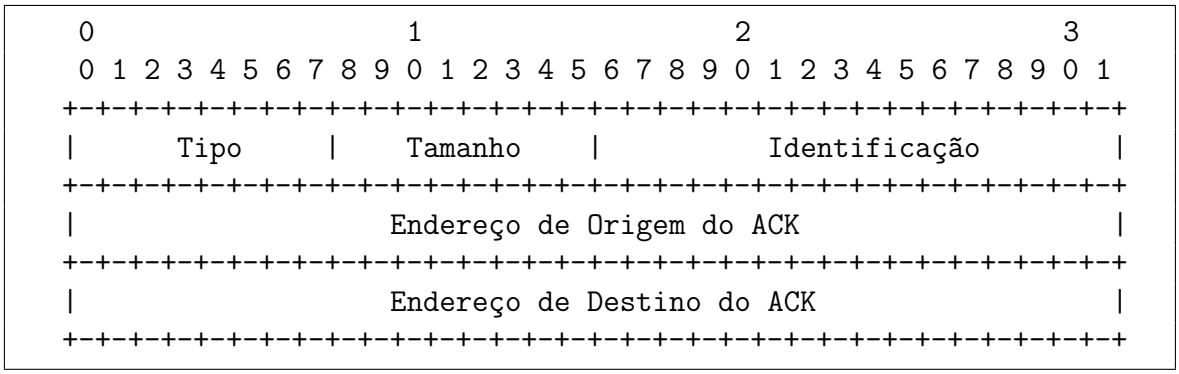

Figura 2.19: Estrutra do cabeçalho *Acknowledgement Option*

2.5 SEGURANÇA E CONFIANÇA EM REDES AD HOC

Apesar da confiança ser um conceito bem conhecido e bastante empregado no nosso cotidiano, é um termo de difícil definição. Gambetta (2000) provê uma definição que é bastante usada por pesquisadores de diversas áreas de conhecimento:

> *A confiança (ou, simetricamente, a desconfiança) é um determinado nível de probabilidade subjetiva com que um agente avalia que outro agente ou grupo de agentes irá realizar determinada ação, antes que ele possa monitorá-la e no contexto em que isso afete sua própria ação.*

Mui & Mohtashemi (2001) abordam outro aspecto da confiança relevante para a pesquisa ao dizerem que confiança é "uma expectativa subjetiva que um agente tem sobre o comportamento futuro de outro agente com base no histórico de seus encontros". Nesse, o ponto importante é o elemento de predição associado aos eventos passados.

Outro conceito bastante relacionado à confiança é a reputação. Sabater & Sierra (2002) definem reputação como uma opinião de alguém sobre algo e pode apresentar três tipologias: reputação individual, reputação social e reputação ontológica. Reputação individual refere-se a como as impressões de um individuo são julgadas por outros. Reputação social refere-se a impressões sobre indivíduos baseada na reputação do grupo social a que pertence. Já a reputação ontológica depende do contexto em que se encontra a reputação, onde a reputação de um individuo pode ter diferentes classificações em comunidades distintas.

Gambetta (1990) expõe que a confiança é baseada na reputação e que deve ser desenvolvida com base no histórico de comportamento através do tempo. Albuquerque (2008) afirma também que a reputação pode ser definida em cenários onde não existem informações suficientes para realizar a inferência sobre a confiança de um membro, e, para descobrir se a entidade é confiável, outros membros da rede podem ser questionados.

A reputação tem sido objeto de estudo de diversas outras áreas de pesquisa, tais como em Ciência da Computação e em Ciências Humanas. Apesar de a reputação ser um artefato antigo, é um conceito que vem sendo bastante utilizado, principalmente em comunidades eletrônicas, mais precisamente em redes sociais, visto que estas comunidades precisam das noções de confiança e credibilidade a fim de fortalecer o comprometimento e a responsabilidade entre os parceiros (GUERRA, 2012).

Outro conceito relacionado à reputação e à confiança é a reciprocidade. A reciprocidade consiste na troca mútua de realizações entre duas entidades. A reciprocidade pode ser representada como um valor associado ao relacionamento entre duas entidades da rede, quanto mais alto este valor, maior a probabilidade de novas interações entre estes duas entidades.

Mui et al. (2002) ilustram, de acordo com a Figura 2.20, o relacionamento entre confiança, reputação e reciprocidade, onde a direção das setas indica o sentido da influência entre os conceitos. De acordo com essa análise, o incremento da reputação de uma enti-

dade pode aumentar a confiança que as demais entidades depositam nesta entidade. O incremento da confiança em uma entidade também pode aumentar a probabilidade de esta agir em reciprocidade. Por fim, o incremento do número de realizações recíprocas pode aumentar a reputação de uma entidade.

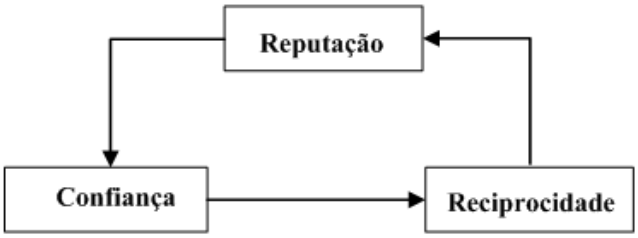

Figura 2.20: Relacionamento entre confiança, reputação e reciprocidade.
(MUI et al., 2002)

2.5.1 Confiança em Ambientes de Computação Ubíqua

A computação ubíqua oferece uma série de desafios em relação à segurança, em especial aos requisitos de privacidade e confiança, por ser alicerçada nas redes *ad hoc*. As suas características (ubiquidade, escalabilidade, mobilidade, autonomia, conectividade global, informações incompletas, etc.) afetam a forma como a segurança deve ser tratada. Por exemplo, mesmo que uma entidade móvel se encontre em um ambiente hostil e desconectada de sua infraestrutura de segurança, deve ser capaz de atribuir privilégios a outras entidades. Porém esses privilégios terão que se basear em informações incompletas que a entidade possui das demais.

As redes *ad hoc* ainda adicionam novos problemas de segurança e confiança. Primeiro, porque a vitalidade da rede depende da cooperação dos nós nas tarefas de organização da rede e encaminhamento de pacotes, mas essa cooperação não pode ser garantida, visto que os nós seguem suas próprias intenções e motivações. Caso os nós não se voluntariem, haverá perda de pacotes e comprometimento do desempenho da rede. Adnane et al. (2013) afirmam que os mecanismos de descoberta e manutenção de rotas intoduzem problemas de segurança específicos dos protocolos de roteamento e as soluções para assegurar tais protocolos é o uso de algumas unidades de controle centralizado ou terceiros confiáveis que acabam limitando a característica da auto-organização das redes *ad hoc*.

Nesse sentido, algumas propostas de sistemas de reputação, incentivos e punições visam livrar a rede do mal comportamento como, por exemplo, em (KHOSRAVI; TYSON,

2006). Outra iniciativa que tem despertado bastante interesse frente as desafios de segurança é a criação de infraestruturas de segurança baseadas em redes de confiança similares as que usamos em nosso cotidiano ao se relacionar com instituições, empresas e outras pessoas, mesmo que desconhecidas (CORDASCO; WETZEL, 2008). Krukow et al. (2008) coloca que dessa forma a confiança se baseará em abstrações computacionais inspiradas no conceito humano de confiança, visando apoiar a tomada de decisões por agentes computacionais na presença de entidades desconhecidas, incontroláveis ou potencialmente perigosas e em contextos sem informações confiáveis, o que torna inútil a aplicação das técnicas clássicas de segurança (BLAZE et al., 1996).

Como o esperado de qualquer conceito relacionado às emoções e experiências humanas, a confiança surge em diferente formas, tais como linguagens de descrição de *middlewares*, redes sociais e gerenciamento de identidades para a interação homem-máquina (KRUKOW et al., 2008). Essas dependem de uma variedade de teorias matemáticas, incluindo, por exemplo, lógica, teoria de jogos, semântica, algorítmica, probabilidade e estatística.

2.5.2 Ataques em Redes *Ad Hoc*

A ausência de infraestrutura das redes *ad hoc* móveis oferecem facilidades aos atacantes. Primeiramente, a natureza de *broadcast* das redes *ad hoc* permite que qualquer dispositivo dotado de interface sem fio possa escutar o meio e capturar pacotes. Adicionalmente, a ausência de entidade de controle central permite que os nós sigam suas próprias motivações nas tarefas da rede, podendo, por exemplo, burlar as regras dos protocolos de roteamento. O papel de roteamento atribuído aos nós também possibilita a interceptação e modificação no repasse dos pacotes.

Foram encontrados em (WANG et al., 2010), (LUNDBERG, 2000), (MARIAS et al., 2006) e (STAJANO; ANDERSON, 2000) os principais ataques contra os protocolos de roteamento de redes *ad hoc*:

- **Ataques Passivos:** os atacantes capturam pacotes da rede, porém não os modificam nem interferem no funcionamento da rede, tornando difícil sua detecção. O tipo mais comum é o *eavesdropping* ou espionagem, no qual o atacante visa o roubo de informações sigilosas;
- **Ataques Ativos:** diferentemente dos ataques passivos, esse visam interferir no funcionamento da rede a partir de modificações nos pacotes ou na injeção de

pacotes maliciosos. As principais técnicas foram reunidas na Tabela 2.6.

Tabela 2.6: Principais técnicas de ataque em redes *ad hoc*

Classe	**Técnica**	**Descrição**
Passivo	*Eavesdropping*	O atacante intercepta os pacotes de dados e extrai informações sigilosas.
	Egoísmo	Para poupar recursos, os nós deixam de fazer o encaminhamento de pacotes.
Ativo	*Flooding*	O atacante injeta mensagens com rotas falsas e pacotes inválidos, causando indisponibilidade e sobrecarga na rede.
	Blackhole	Os atacantes prometem rotas mais curtas ao destino, atraindo o tráfego da rede, mas, quando recebem os pacotes, fazem o descarte ou encaminham para nós maliciosos.
	Rushing	Os atacantes encaminham rapidamente mensagens RREQ, sem as modificações necessárias, fazendo que os demais nós dispensem o fluxo legítimo da mensagem.
	Wormhole	os atacantes desviam as mensagens RREQ por um túnel formado por outros atacantes, interrompendo a comunicação.
	Route Table Overflow	O atacante injeta anúncios de rotas para nós inexistentes até que as tabelas de rotas estourem, impedindo a inclusão de rotas válidas.
	Sleep Deprivation Torture	o atacante visa consumir a bateria do nó atacado, requisitando rotas ou encaminhando pacotes desnecessários.
	Sinkhole	O atacante atrai o tráfego para um determinado nó da rede, que está sob seu controle, fazendo com que a estação base seja incapaz de obter dados da rede.
	Sybil	O atacante cria múltiplas identidades por meio do roubo de identidades de nós legítimos.
	Spoofing	O atacante altera sua identidade para ser visto na rede como outro nó privilegiado.
	Fabricação	O atacante gera mensagens de roteamento e pacotes de dados falsos a fim de enganar a rede e o esquema de roteamento.
	Location disclosure	O atacante adquire dados de localização dos nós ou da topologia de toda rede, permitindo descobrir nós adjacentes e os caminhos utilizados.
	Byzantine attack	Nós maliciosos agem em conjunto, interferindo no roteamento, escolhendo os piores caminhos, criando *loops* e descartando pacotes seletivamente.
	Replay	O atacante repete o envio dos pacotes dados a fim de sobrecarregar a rede e confundir o protocolo de roteamento.

Garantir a segurança em sistemas distribuídos, como as redes *ad hoc*, é uma tarefa difícil, ainda mais em redes com grande número de nós móveis. Os quesitos de auto-organização e auto-configuração da rede limitam a ação de entidades de controle central, como os sistemas de *firewall* e de detecção de intrusões. Além disso, nas redes *ad hoc*, as tarefas da rede são confiadas aos nós que, em príncipio, seguem suas próprias motivações.

Brandao & Fraga (2004) afirmam que a primeira idéia para resolver problemas de segurança é a implantação de uma arquitetura de controle centralizado, na quais todos os eventos são submetidos a um sistema de gerenciamento central que seja capaz de correlacionar e analisar todos os eventos, porém essa arquitetura só funciona bem em redes de pequeno porte, devido ao grande volume de dados que trafegam pela rede e da necessidade de alto poder de processamento para analisar todos os dados. Além disso, a centralização implica em ponto único de falhas, que poderia ser facilmente explorado por atacantes.

As desvantagens dos sistemas de segurança centralizados poderiam ser minimizadas e a eficiência ampliada com a adoção de sistemas que façam a coleta, a correlação e a análise de eventos provenientes de diferentes origens distribuídas pela rede. Ainda assim haveriam dificuldades quanto à correlação temporal dos eventos a ao volume de informações a serem analisadas (BRANDAO; FRAGA, 2004).

Uma solução para a contenção de ataques, é o emprego de sistemas de segurança distribuídos e móveis. Puttini (2004), por exemplo, apresenta um modelo para segurança de redes móveis *ad hoc* que faz uma combinação de serviços de segurança preventiva e corretiva por meio de três serviços de segurança que interagem entre si: um serviço de certificação digital que permite impor uma política de segurança específica para discriminar nós confiáveis e não confiáveis na rede, um serviço de autenticação que possibilita assegurar as mensagens advindas de nós não confiáveis de acordo com a política de segurançae um serviço de detecção de intrusão permite identificar e eliminar nodos comprometidos na rede. A proposta é imediatamente aplicada na segurança dos protocolos de roteamento e autoconfiguração.

2.5.3 Mecanismos de Confiança para Redes *Ad Hoc*

As subseções seguintes fazem uma visão geral dos principais mecanismos de confiança projetados paras as redes *ad hoc*: Watchdog, Pathrater, CONFIDANT, CORE, SORI e OCEAN. Algumas idéias desses esquemas foram incluídas na proposta do METRUbi.

2.5.3.1 *Watchdog* e *Pathrater*

Marti et al. (2000) abordam duas técnicas que em conjunto podem detectar nós mal comportados e excluí-los das rotas, aumentando o desempenho da rede. A primeira técnica é o *Watchdog* que faz com que os nós monitorem uns aos outros no encaminhamento de pacotes por meio de interfaces de rede em modo promíscuo, permitindo fazer o *acknowledgment* passivo (JUBIN; TORNOW, 1987). A segunda é o *Pathrater* que elenca as rotas mais confiáveis com base nas informações coletadas dos nós.

Quando um nó na rede é recém descoberto pelo protocolo de roteamento, o *Pathrater* atribui uma pontuação neutra de 0,5. Se todos os nós forem neutros, o algoritmo seleciona a rota de menor distância. A adaptação do *Pathrater* é garantida por incrementos periódicos de 0,01 a cada 200 ms na pontuação de todos nós que participam ativamente de uma rota. O valor máximo de pontuação do nó é de 0,8. Quando ocorre quebra de enlace, a pontuação do nó é decrementada em 0,05 até o valor mínimo de 0,0. Os nós inativos não são pontuados pelo esquema. Com isso o protocolo de roteamento tem condições de escolher rotas com maior pontuação.

2.5.3.2 *CONFIDANT: Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks*

Buchegger & Le Boudec (2002) propuseram um mecanismo de confiança, chamado CONFIDANT, projetado para ser uma extensão do protocolo DSR. O objetivo do esquema é evitar a participação dos mal comportados no roteamento. A premissa básica é que pacotes de nós mal comportados não são encaminhados pelos nós bem comportados e vice-versa. O mecanismo conta ainda com a possibilidade de redenção e reintegração de nós mal comportados na rede, caso eles deixem de agir maliciosamente ou sejam vítimas de falsas acusações.

O CONFIDANT implementa quatro componentes funcionais em cada nó da rede: (a) Monitor que é o responsável pelo vigilância dos nós vizinhos; (b) Gerenciador de Confiança que gerencia o recebimento e envio de mensagens de alarmes relacionadas aos nós maliciosos; (c) Sistema de Reputação que faz a manutenção de listas negras e das tabelas de pontuação; e (d) Gerenciador de Rotas que seleciona as rotas com melhores métricas, evitando os nós maliciosos. Os criadores usaram o termo reputação para se referir à avaliação do comportamento dos nós no encaminhamento de pacotes, e usaram o termo confiança para a participação no protocolo de roteamento.

Os nós monitoram os seus vizinhos e atribuem valores de reputação de acordo com o

seu comportamento. Um nó pode detectar o comportamento egoísta do próximo nó da rota, presente no *Source Route* do DSR, tanto diretamente, com monitoramento passivo por meio do modo promíscuo, quanto indiretamente, por meio da observação de anomalias no protocolo de roteamento. O componente *Monitor* é que registra esses desvios de padrão.

Após a ocorrência de mal comportamentos, o componente Sistema de Reputação é invocado, e mensagens de alarme são enviadas para o componente Gerenciador de Confiança. As mensagens de alarme são geradas pelos próprios nós após observarem ou receberem um aviso de comportamento malicioso de outro nó. Os receptores das mensagens de alarme, chamados de amigos, são mantidos em uma lista de amigos. Alarmes recebidos de nós desconhecidos são checadas em termos de confiabilidade antes de disparar qualquer reação. A desvantagem desse mecanismo é que requer relações de confiança pré-configuradas.

Se existirem evidências suficientes de que um nó citado nos alarmes é malicioso, essa informação é enviada para o componente Sistema de Reputação. Este componente gerencia uma tabela que consiste de entradas relacionadas aos nós e suas pontuações. A pontuação é alterada se duas condições forem verdadeiras: (i) existe evidência suficiente de comportamento malicioso, e , (ii) o mal comportamento ocorreu várias vezes, excedendo um limite especificado. O cálculo da pontuação é alterado em função da natureza da percepção do mal comportamento. Pesos maiores são atribuídos por experiências próprias, pesos menores pelas observações capturadas na vizinhança, e pesos ainda menores para experiências capturadas de terceiros.

Se a pontuação de reputação de um dado nó atingir um limiar mínimo no componente Sistema de Reputação, o componente Gerenciador de Rotas é ativado. Este componente exclui rotas que contenham nós mal comportados e os isola, prioriza as rotas do *cache* com menor número de nós mal comportados, e encaminha alarmes sobre os nós.

Segundo Buchegger & Le Boudec (2003), a primeira versão do CONFIDANT era sujeita ao fenômeno do espalhamento de boatos [13]. Na versão atual esse problema foi resolvido com um modelo Bayesiano que classifica e exclui os nós mentirosos.

[13] O fenômeno conhecido na literatura como *Rumor Spreading Phenomena* consiste na deterioração dos sistemas de reputação frente à ação de nós que emitem avisos mentirosos na rede.

Tanto as reputações negativas quanto positivas são calculadas por meio de um "fator de cooperação". Este fator consiste na frequência de mal comportamentos em relação à atividade acumulada do nó. Cada nó i mantém um fator de cooperação de cada nó j, chamado de R_{ij}, que é expressado em função de α e β, que são, respectivamente, o número de mal comportamentos e bons comportamentos. Esses números são atualizados com base na experiências mais recentes. Uma recomendação será aceita se for compatível, ou seja, se os valores de reputação (negativa e positiva) não forem completamente distintos.

2.5.3.3 *CORE: A Collaborative Reputation Mechanism for node cooperation in Ad hoc Networks*

Esse esquema foi proposto por Molva & Michiardi (2001) e também usou o protocolo DSR como plataforma de desenvolvimento. Ele estimula a colaboração dos nós por meio de monitoramento do cooperativismo dos nós e de um mecanismo de reputação, usando experiências pessoais e de terceiros de forma combinada em uma função especializada. A função utiliza um mecanismo de *Watchdog* para avaliar o comportamento dos demais nós. Caso o comportamento observado seja diferente do que foi calculado pela função, a pontuação do nós é alterada.

Cada nó da rede monitora o comportamento dos nós vizinhos, com relação à função, e coleta observações sobre a execução da função. Essas observações são registradas em uma tabela de reputação, mantida por cada nó. Cada registro da tabela corresponde a um nó vizinho e consiste em quatro entradas: um identificador único, uma coleção de observações pessoais recentes, uma lista de valores de reputação providos por terceiros e o valor de reputação calculada pela função de monitoramento. Com isso, cada nó mantém uma tabela de reputação para cada função de monitoramento. Finalmente, uma tabela de reputação global é usada para combinar os diferentes valores de reputação resultantes das diferentes funções.

O CORE diferencia os valores de reputação entre reputação subjetiva, que assume o intervalo [-1, 1], reputação indireta vinda de terceiros, e reputação funcional, que são ponderadas para prover um valor combinado de reputação. A fórmula usada para avaliar o valor de reputação evita os falsos positivos usando um fator de temporalidade que dá maior relevância às observações passadas. A fórmula usada para avaliar os valores de reputação evita falsas detecções por meio de um fator de temporalidade que dá maior relevância às observações mais recentes. Contudo, esse esquema é vulnerável

aos ataques nos quais os nós constroem uma boa reputação para posteriormente se comportarem mal.

Os valores de reputação armazenados na tabela de reputação são decrementados com o passar do tempo. Por exemplo, caso um nó entre em modo sleep, sua reputação será até um valor nulo que corresponde a um comportamento neutro. Além disso, as informações de reputação podem ser obtidas de nós de outras vizinhanças, caso a função monitorada conte com mensagens de *reply*. Neste caso, apenas as pontuações positivas são atribuídas aos nós que participaram da função.

O esquema do CORE é imune aos ataques de espalhamento de boatos, pois nenhuma pontuação negativa é disseminada, sendo impossível que um nó malicioso decremente a reputação de outros nós. Porém, dois ou mais nós podem se coligar para o incremento mútuo de suas reputações por meio de mensagens recíprocas de pontuação positiva. Para prevenir esse fenômeno, o CORE fornece implicitamente alguma proteção visto que atribui pesos maiores para as reputações subjetivas do que para as indiretas.

O CORE permite que as redes *ad hoc* isolem gradualmente os nós mal comportados. Quando a reputação de um nó fica abaixo de um limiar predefinido, os serviços deixam de ser fornecidos a este nó. Os nós mal comportados podem ser reintegrados à rede, caso voltem a cooperar com as operações da rede. Vale ressaltar que o CORE não consegue distinguir os nós mal comportados de nós com mal funcionamento.

Por fim, o mecanismo do CORE assume que todos os nós usam cálculos idênticos dos valores de reputação, atribuindo os pesos idênticos para as mesmas funções. Porém nem sempre isso pode ser assumido nas redes *ad hoc*, nas quais os nós podem ser equipados com diferentes recursos e serviços, requerendo níveis diferenciados de importância paras funções.

2.5.3.4 *SORI: A Secure and Objective Reputation-based. Incentive Scheme for Ad-hoc Networks*

O foco do SORI, proposto por Khosla (2004), é na função de encaminhamento de pacotes dos protocolos de redes *ad hoc*. Ele possui três componentes básicos: o Monitoramento de Vizinhos, a Propagação da Reputação e a Punição. O modo promíscuo é assumido a fim de possibilitar a escuta dos pacotes de sua vizinhança e manutenção de uma lista de nós vizinhos.

Cada função de encaminhamento dos nós vizinhos é associada a dois parâmetros: o $RF_N(X)$ (*request-for-forwarding)* e o $HF_N(X)$ (*has-forwarded*). O indicador $RF_N(X)$ é usado para indicar o número total de pacotes que o nó N transmitiu ao nó X para que fosse encaminhado. Já o $HF_N(X)$ corresponde ao número total de pacotes que foi encaminhado pelo nó X por meio do monitoramento do nó N. Por meio do $RF_N(X)$ e do $HF_N(X)$, o nó N cria um registro, chamado de registro de validação local. Esse registro é denotado por $LER_N(X)$ para o vizinho X e contém uma métrica de confiança que é usada para determinar o quanto o nó N confia no nó X a partir do julgamento de sua reputação.

O SORI combina características de observação pessoal com o a observação das informações de reputação difundidas na rede. Nesse esquema, os nós trocam informações de reputação apenas com seus vizinhos. Dessa forma, um nó não cooperativo será punido por todos os vizinhos, e não apenas pelo nó que foi afetado pelo mal comportamento. Cada nó atualiza periodicamente seu $LER_N(X)$ para cada nó vizinho X com base nos valores correntes de $RF_N(X)$ e $HF_N(X)$.

O registro atualizado é divulgado na vizinhança, caso a relação $\frac{RF_N(X)}{HF_N(X)}$ tenha uma mudança significativa. O nó N usa seu próprio $LER_N(X)$ e os valores recebidos da vizinhança para calcular seu registro geral de avaliação do nó X, denotado por $OER_N(X)$. Para fazer isso, ele leva em conta a credibilidade dos nós que contribuem para o calculo da reputação. Isso torna dificulta ataques de múltiplas identidades, isto é, tentativas de se passar por outro nó com melhor reputação.

Se o $OER_N(X)$ está abaixo de um limite pré-definido, o nó N toma uma ação punitiva por meio do descarte dos pacotes originados por X de forma probabilística. Esse mecanismo foi projetado para tratar mais generosamente os nós que não descartam pacotes intencionalmente.

2.5.3.5 *OCEAN: Observation-based Cooperation Enforcement in Ad hoc Networks*

Bansal & Baker (2003) propuseram o esquema que introduz uma camada intermediária entre a camada de rede e a camada MAC. Essa camada auxilia os nós para fazer decisões mais inteligentes sobre o roteamento e o repasse de pacotes. Foi projetado inicialmente para o protocolo DSR, porém pode ser aplicado a outros protocolos de roteamento. O OCEAN adota apenas as observações pessoais dos nós.

Cada nó mantém pontuações para cada nó vizinho e monitora seus comportamento

por meio do monitoramento em modo promíscuo. Eventos positivos ou negativos são registrados com base nas reações de um vizinho ao qual se espera o encaminhamento de pacotes. As pontuações são inicializadas com um valor neutro. Baseado em estudos empíricos, os valor absoluto de um decremento é escolhido para ser maior do o valor de um incremento. Quando a pontuação de um nó fica abaixo de um limiar, chamado de limiar de penalidade, o nó é adicionado a uma lista negra. Essa lista é construída usando as experiências pessoais do nó e é anexada à mensagem RREQ do DSR a fim de ser divulgada na rede.

Uma rota é classificada como boa ou ruim, dependendo se o próximo salta pertence à lista negra. Os receptores das mensagens RREQ decidem descartá-la ou dar continuidade ao fluxo. Dessa forma, cada nó ao longo da rota faz sua própria decisão sobre a confiança dos demais nós e possui apenas controle nas rotas em que pertençam. Os nós devem rejeitar os pacotes provenientes de nós que estejam nas listas negras, logo, os nós mal comportados são eventualmente isolados. Porém o mecanismo prevê uma forma de redenção, permitindo que os nós que outrora foram mal comportados se tornem operacionais novamente. Após certo período de tempo, os nós são excluídos das listas negras e recebem uma pontuação neutra.

O OCEAN usa uma política diferenciada para lidar com os nós que não participam no processo de descoberta de rotas. Essa política, baseada em sistema de créditos, requer um hardware a prova de fraudes ou um servidor central. Cada nó mede o comportamento dos seus vizinhos por meio de interações diretas. Os nós registram a taxa de encaminhamento dos seus vizinhos mantendo um contador para cada nó. O contador aumenta quando a solicitação de encaminhamento do pacote é enviada ao nó e diminui com uma requisição daquele nó. Suponha que um nó B não participa do estabelecimento da rota com o nó de origem A. Se o nó B demanda um encaminhamento para o nó A, então, o A irá punir o B e rejeitar suas solicitações enquanto o contador de B for baixo.

2.6 APRENDIZAGEM INDUTIVA E ÁRVORES DE DECISÃO

A ideia central da aprendizagem é possibilitar que os agentes aumentem sua capacidade de agir no futuro (RUSSELL; NORVIG, 1995). O processo de aprendizagem ocorre a medida que o agente observa suas interações com o mundo e seu processo interno de tomada de decisões. Indução, em oposição a dedução, é o processo de se obter uma hipótese a partir de exemplos e fatos já existentes. A aprendizagem indutiva seria,

então, um método de aprendizagem por meio de exemplos.

Por exemplo, para o conjunto de pares $(x, f(x))$ mostrados na Figura 2.21, deseja-se encontrar uma função $h(x)$, chamada de hipótese, que se aproxime de f. Podem existir diferentes hipóteses para um mesmo conjunto de exemplos. No exemplo da Figura 2.21, pode-se aproximar h a uma reta ou a uma função senoidal. O grau de qualidade das hipóteses depende do tamanho do espaço de amostras utilizado para aproximar f.

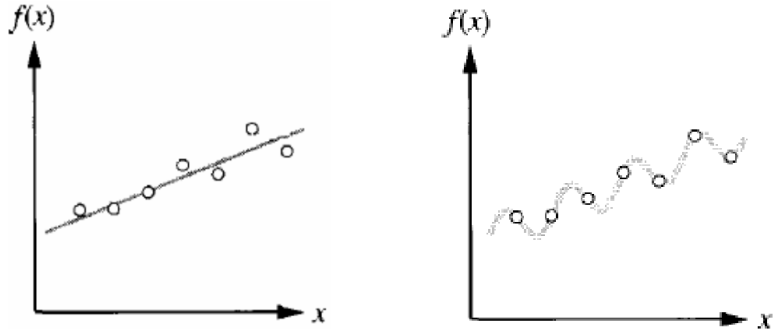

Figura 2.21: Aproximações de $f(x)$ a partir de pares $(x, f(x))$

Mitchell (1997) coloca que a aprendizagem, por meio da qual o sistema deduz o conhecimento pela observação do seu ambiente, possui duas abordagens: a aprendizagem supervisionada e a aprendizagem não supervisionada. Na aprendizagem supervisionada, o sistema modela uma função (ou classificador) com base em dados de treinamento compostos por um valor de entrada (tipicamente um vetor de valores) e um valor de saída (ou classe). A função deve inferir o valor de saída para qualquer entrada válida. Mesmo que os valores de entrada sejam incompletos ou desconhecidos, o sistema deve estimar razoavelmente os valores de saída por meio de generalizações e abstrações. Já na aprendizagem não supervisionada, o sistema é provido com dados de treinamento compostos apenas por valores de entradas, ou seja, nenhum valor de saída é definido. O sistema deve observar os exemplos e reconhecer padrões autonomamente, resultando em um conjunto de regras para as classes.

2.6.1 Árvores de Decisão

Árvores de decisão são métodos de aprendizado simbólico bastante utilizados para a inferência indutiva. Podem ser usadas para dar ao agente a capacidade de aprender, bem como para tomar decisões. A aprendizagem é indutiva, pois cria uma hipótese baseada em instâncias particulares que gera conclusões gerais (RUSSELL; NORVIG, 1995). Os algoritmos de árvores de decisão basicamente dividem um problema complexo em partes menores e mais simples e, após isso, refazem o mesmo processo com os subproblemas encontrados e assim sucessivamente até chegar na solução.

A Figura 2.22 (a) mostra a representação gráfica de uma árvore de decisão. No topo, temos o nó raiz. Cada nó de decisão contem um teste de um atributo. Os ramos descendentes correspondem aos valores possíveis do respectivo atributo. As folhas estão associadas às classes. Cada percurso da árvore - do nó raiz à folha - corresponde a uma regra de classificação.

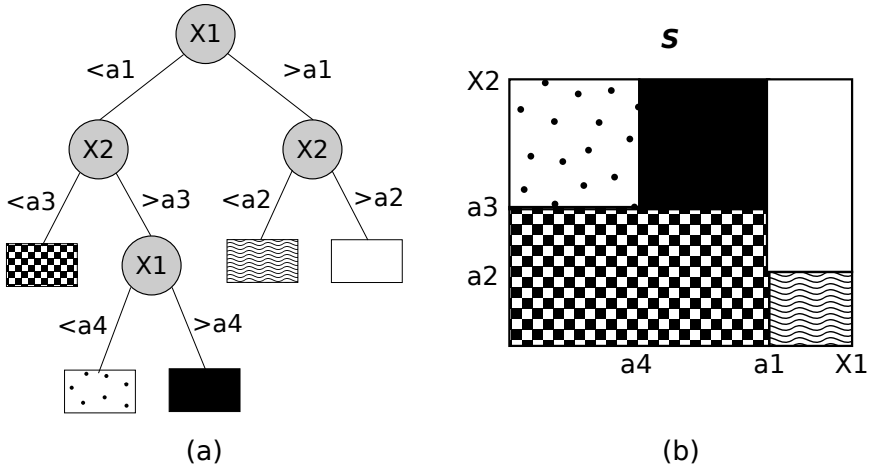

Figura 2.22: Exemplo de árvore de decisão

A capacidade de classificação das árvores de decisão vem da divisão do espaço definido pelos atributos em subespaços conforme mostrado na Figura 2.22 (b). Cada subespaço corresponde a uma classe. Note que a classe referente à região em xadrez pode ser definida pela regra: $X1 < a1 \wedge X2 < a3$. O espaço de todos valores dos atributos S pode ser representado pela Equação 2.6. Pode-se notar que uma árvore de decisão representa uma disjunção de conjunções, na qual cada ramo é uma conjunção.

$$\begin{aligned} S = (\ X1 < a1\ \wedge\ X2 < a3\ )\ \vee \\ (\ X1 < a1\ \wedge\ X1 > a3\ \wedge\ X1 < a4\ )\ \vee \\ (\ X1 < a1\ \wedge\ X2 > a3\ \wedge\ X1 > a4\ )\ \vee \\ (\ X1 > a1\ \wedge\ X2 < a2\ )\ \vee \\ (\ X1 > a1\ \wedge\ X2 > a2\ ) \end{aligned} \tag{2.6}$$

A implementação de árvores de decisão é similar a de regras *if-then*, sendo estruturas muito usadas em sistemas especialistas e em problemas de classificação. As árvores de decisão tomam como entrada uma situação descrita por um conjunto de atributos e retornam uma decisão após o processamento das regras. Os atributos de entrada podem

ser discretos ou contínuos. O aprendizado de valores discretos é chamado classificação (RUSSELL; NORVIG, 1995).

Mitchell (1997) afirma que o aprendizado utilizando árvore de decisão é indicado para problemas com as seguintes características:

- **Instâncias são representadas através de pares atributo-valor:** Instâncias são descritas por um conjunto fixo de atributos (por exemplo: Temperatura) e seus respectivos valores (por exemplo: Quente). A situação mais fácil de aprendizado utilizando árvore de decisão é quando cada atributo assume um número pequeno de possíveis valores disjuntos (por exemplo, Quente, Moderado, Frio). Porém, extensões para o algoritmo básico permitem atribuir valores reais, por exemplo, representando Temperatura numericamente.
- **A função tem valores discretos:** A árvore de decisão classifica com valores lógicos (Verdadeiro: sim ou Falso: não) para cada exemplo. Métodos de árvore de decisão podem ser facilmente estendidos para funções com mais de dois valores possíveis. Uma extensão mais significativa permite utilizar funções reais, entretanto a aplicação de árvores de decisão neste tipo de caso é menos comum.
- **Permitem descrições disjuntas** Como notado acima, árvores decisão naturalmente representam expressões disjuntas. Os dados de treinamento podem conter erros. As árvores de decisão são robustas a erros, tanto erros nas classificações dos exemplos de treinamento, quanto erros nos valores dos atributos que descrevem estes exemplos.
- **Os dados de treinamento podem conter valores de atributo indefinidos:** Podem ser usados métodos de árvore de decisão até mesmo quando alguns exemplos de treinamento têm valores desconhecidos (por exemplo, se a Umidade do dia é conhecida somente em alguns dos exemplos de treinamento).

Muitos problemas práticos possuem estas características. Aprendizado utilizando árvore de decisão foi aplicado então a problemas como classificar os pacientes médicos pela doença, causa de maus funcionamentos de equipamentos, e a probabilidade de candidatos a empréstimo ficarem inadimplentes. Tais problemas, nos quais a tarefa é classificar exemplos em possíveis categorias discretas, são frequentemente chamados de problemas de classificação.

Para a construção de árvores de decisão, deve-se selecionar o atributo que é mais eficiente para dividir os exemplos em subconjuntos de exemplos. Esse processo se dá de forma descendente - começando pelo nó raiz (i.e., o primeiro nó de decisão), passando pelos ramos (i.e., os nós de decisão subordinados) até chegar às folhas (i.e., as classes). A parada ocorre quando a árvore classifica todos os exemplos de treinamento ou até que todos os atributos sejam usados (MITCHELL, 1997). Duas medidas estatísticas são empregadas para auxiliar na tarefa de descoberta dos atributos mais eficientes - a entropia e o ganho de informação - discutidas nas próximas subseções.

2.6.2 Entropia da Informação

A entropia é uma propriedade estatística que mede a homogeneidade, isto é, a (im)pureza de uma coleção arbitrária de exemplos (MITCHELL, 1997). Essa propriedade pode ser usada para determinar qual atributo de uma coleção separa melhor os exemplos em subconjuntos mais homogêneos. A entropia é utilizada para a otimizar as árvores de decisão, escolhendo os nós de decisão (ou atributos) que mais são úteis.

Para ilustrar, considere uma coleção S que contém exemplos positivos e negativos sobre algum conceito objetivo, a entropia de S pode ser calculada conforme a Equação 2.7, na qual $p_{\oplus}$ é a quantidade de exemplos positivos no conjunto de exemplos S e p_{minus} é a quantidade de exemplos negativos em S. Considera-se para os cálculos que a expressão $0 \log_2 0 = 0$.

$$Info(S) = -p_{\oplus} \log_2 p_{\oplus} + -p_{\ominus} \log_2 p_{\ominus} \tag{2.7}$$

Por exemplo, suponha que S seja uma coleção de 14 exemplos de algum conceito que inclui 9 exemplos positivos e 5 exemplos negativos denotados por $[9+, 5-]$. Então, a entropia de S pode ser calculada conforme a Equação 2.8.

$$Info([9+, 5-]) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940 \tag{2.8}$$

Vale observar que a entropia é zero se todos os membros de S pertencem à mesma classe . Por exemplo, se todos os membros são positivos $p_{\oplus} = 1$), então p é zero e $Info(S) = -1 \log_2 1 - 0 \log_2 0 = -1 \times 0 - 0 \times \log_2 0 = 0$. A Entropia é igual a 1 quando a coleção contém o mesmo número de exemplos positivos e negativos. Se a coleção

contém números desiguais de exemplos positivos e negativos, a entropia gira em torno de 0 e 1.

De forma geral, se uma classe pode assumir c valores diferentes, então a entropia de S é definida mediante a Equação 2.9, na qual p_i é a quantidade de exemplos de S que pertence à classe i. Note que o logaritmo está na base 2, para que a entropia seja tratada como uma medida do tamanho da codificação em *bits*. Ainda é possível observar que, se uma classe pode assumir c possíveis valores, a entropia pode ser tão grande quanto $\log_2 c$.

$$Info(S) = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{2.9}$$

2.6.3 Ganho de Informação

Visto que a entropia determina a impureza do conjunto de exemplos, pode ser calculada a medida de eficiência de classificação de um dado atributo, chamada de ganho de informação, a partir da avaliação da redução da entropia com o particionamento dos exemplos de acordo com este atributo (MITCHELL, 1997).

O ganho de informação, $Gain(S, A)$, do atributo A em relação ao conjunto de exemplos S é definido conforme a Equação 2.10, na qual $Values(A)$ representa a gama de valores possíveis do atributo A e S_v é um subconjunto de S em que o atributo A tem valor igual a v (i.e., $S_v = \{s \in S | A(s) = v\}$).

$$Gain(S, A) = Info(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot info(S_v) \tag{2.10}$$

Em outras palavras, o ganho de informação é basicamente a diferença entre a entropia do conjunto original de exemplos e a soma das entropias de cada subconjunto ponderadas pela sua proporção de exemplos no conjunto original. Quanto maior a medida de ganho de informação de um atributo, mais eficiente será sua classificação, visto que gerará subconjuntos de exemplos mais homogêneos.

Em árvores de decisão, a priorização de atributos com maior ganho de informação nos pontos de maior hierarquia da árvore garante estruturas mais simples e um número

reduzido de regras de decisão, com maior poder de generalização.

2.6.4 Algoritmo ID3

O ID3 é um algoritmo desenvolvido por Quinlan (QUINLAN, 1993) para criação de árvores de decisão. Ele usa o conceito de ganho de informação para selecionar os atributos que serão utilizados a cada passo, enquanto constrói a árvore. A Figura 2.23 apresenta o pseudocódigo do algoritmo ID3.

```
ID3(Exemplos, Classes, Atributos):
Crie o Nó_Raiz da árvore;
Se todos os exemplos forem da Classe_X:
  Retorne apenas o Nó_Raiz com o nome Classe_X;
Se não existirem atributos:
  Retorne apenas o Nó_Raiz com o nome da classe mais frequente;
Caso contrário:
 A = atributo que melhor classifica Exemplos;
 Nó_Raiz = A;
 Para cada valor V_i de A:
   Adicione novo ramo relativo ao teste A = V_i;
   Exemplos_{V_i} = subconjunto de Exemplos onde A = V_i;
   Se Exemplos_{V_i} for vazio:
     Crie folha com nome da classe mais frequente de Exemplos_{V_i};
   Caso contrário:
     Crie ramo com ID3(Exemplos_{V_i}, Classes, Atributos - A);
Retorne Nó_Raiz;
```

Figura 2.23: Pseudocódigo do algoritmo ID3

O melhor atributo é selecionado e usado como o teste no nó raiz da árvore. Um descendente do nó raiz é criado para cada possível valor deste atributo, e os exemplos de treinamento são particionados e associados a cada nó descendente para selecionar o melhor atributo para ser testado naquele ponto na árvore.

2.6.5 Algoritmo C4.5

O C4.5 é uma evolução do algoritmo básico do ID3 desenvolvido por Quinlan (1993). As principais melhorias implementadas pelo algoritmo foram: redução na taxa de erros de "poda", proteção contra super-treinamento, suporte a atributos de valores contínuos, suporte a dados de treinamento com valores de atributos ausentes e algoritmo com maior eficiência computacional.

O algoritmo constrói árvores de decisão a partir de um conjunto de exemplos de treinamento, também usando as propriedades de entropia e ganho de informação. A Figura

2.24 ilustra um conjunto de exemplos de treinamento sobre a decisão "jogar ou não jogar tênis ?". Os exemplos já devem ser previamente classificados, pois se trata de um método de treinamento supervisionado. Cada exemplo é composto por um vetor que apresenta os seus atributos e a respectiva classe. No caso ilustrado, a definição do vetor de atributos seria `(tempo, temperatura, umidade, vento)` e as classes seriam `Sim` ou `Não`.

```
ensolarado, 85, 85, false, Não
ensolarado, 80, 90, true, Não
nublado, 83, 78, false, Sim
chuvoso, 70, 96, false, Sim
chuvoso, 68, 80, false, Sim
chuvoso, 65, 70, true, Não
nublado, 64, 65, true, Sim
ensolarado, 72, 95, false, Não
ensolarado, 69, 70, false, Sim
chuvoso, 75, 80, false, Sim
ensolarado, 75, 70, true, Sim
nublado, 72, 90, true, Sim
nublado, 81, 75, false, Sim
chuvoso, 71, 80, true, Não
```

Figura 2.24: Conjunto de exemplos de treinamento para árvores de decisão

Para cada nó da árvore, o C 4.5 escolhe um atributo que melhor divide o conjunto de exemplos em subconjuntos com menor entropia. Esse critério é definido com base no ganho de informação de cada atributo, que deve ser maximizado. Após isso, o C4.5 faz a mesma operação recursivamente em cada subconjunto (escolhe novamente um atributo que melhor divide o subconjunto), obtendo novos ramos. A Figura 2.25 mostra a árvore de decisão resultante do treinamento. O número à direita de cada classe denota a quantidade de exemplos pertencentes ao ramo.

```
tempo = nublado: Sim (4.0)
tempo = ensolarado:
|   umidade <= 75 : Sim (2.0)
|   umidade > 75 : Não (3.0)
tempo = chuvoso:
|   vento = true: Não (2.0)
|   vento = false: Sim (3.0)
```

Figura 2.25: Árvore de decisão gerada com o C4.5

A estrutura de uma árvore de decisão pode ser facilmente convertida em regras do tipo

if-then que poderão ser incorporadas nas aplicações. Em relação ao exemplo da decisão sobre jogar tênis, as regras de decisão são mostradas na Figura 2.26.

```
if (tempo == nublado){
   jogarTenis=Nao;
}
if (tempo == ensolarado){
   if (umidade <= 75) {
      jogarTenis=Sim
   } else {
      jogarTenis=Nao
   }
}
if (tempo == chuvoso){
   if (vento) {
      jogarTenis=Nao
   } else {
      jogarTenis=Sim
   }
}
```

Figura 2.26: Regras *if-then* para uma árvore de decisão

3 PROPOSTA DO MECANISMO DE AVALIAÇÃO DE CONFIANÇA DE ROTAS

Este capítulo se inicia com a apresentação das definições e terminologias adotadas a fim de uma melhor compreensão da proposta. Em seguida são apresentadas as premissas para a plena operação do METRUbi, visando orientar futuros experimentos. A arquitetura geral e seus componentes são especificados, incluindo os seus relacionamentos e fluxos. Como foco central é apresentado o algoritmo do componente Indutor-DT do METRUbi e explicado seu funcionamento.

3.1 DEFINIÇÕES E TERMINOLOGIAS ADOTADAS

A especificação do METRUbi, incluindo sua arquitetura, seus componentes, suas métricas e seus esquemas, fez uso de definições e terminologias que frequentemente são controversas ou omissas na literatura da área. Esta seção tem o objetivo de fazer um alinhamento dos termos adotados a fim de melhorar a compreensão do texto.

Nós egoístas e nós maliciosos. A primeira discussão gira em torno da tipologia de nós no contexto de sistemas distribuídos e autônomos. Nesse âmbito, foi adotada a classificação de Pirzada et al. (2006) que faz distinção entre nós maliciosos e nós egoístas. Nós maliciosos são aqueles que executam ataques passivos de *eavesdropping* [14] ou ataques ativos [15] de fabricação, modificação e interrupção. Os nós egoístas são aqueles que apenas deixam de cooperar no encaminhamento de pacotes visando a economia de energia.

Nós bem comportados e nós mal comportados. De forma geral, a literatura engloba os nós egoístas, os nós maliciosos e os nós em modo *sleep* no grupo de nós mal comportados, pois estes se desviam do papel esperado na rede rede. Os nós bem comportados são aqueles que correspondem corretamente aos seus papéis na rede.

Redes *ad hoc*. O termo "rede *ad hoc*"é comumente usado para fazer menção às redes

[14] *Eavesdropping* é um ataque passivo em que o atacante intercepta os pacotes de dados e extrai informações sigilosas usando, geralmente, uma ferramenta de *sniffing*.

[15] Stallings (2007) define as classes de ataques como interceptação, fabricação, modificação e interrupção.

ad hoc móveis ou MANETs. Porém, no contexto dos ambientes de computação ubíqua, poderia se estender para um sentido mais amplo, envolvendo um *mix* de arquiteturas de redes improvisadas ou não por interfaces sem fio e pontos de acesso fixos de forma que suportem a conectividade de toda gama de dispositivos. Esse é o sentido adotado para redes *ad hoc*.

Confiança de rotas. Com base nos conceitos sobre confiança discutidos na Seção 2.5, as expressões "confiança de rotas" ou "confiança no roteamento" adotadas se referem à expectativa subjetiva de que um nó tem sobre um grupo de nós intermediários, associados por uma rota, na ação de encaminhamento dos seus pacotes até o seu destino, da qual ele depende e pode ser afetado. No mecanismo, esta expectativa é estimada por meio da observação do comportamento passado do grupo de nós da rota.

Métricas de confiança de rotas. Uma métrica de confiança, como relatada em (CHO et al., 2011) e (THEODORAKOPOULOS; BARAS, 2004), é a medida probabilística do grau de confiança de uma rota. Esse grau pode ser calculado com base nas confianças dos enlaces e dos nós. Nesta proposta a métrica de confiança mede a chance de um pacote chegar com sucesso ao seu destino quando submetido ao protocolo de roteamento.

Distância da rota. O termo distância da rota é a medida da quantidade de saltos entre os nós. Com isso, a partir da distância é possível deduzir o número de enlaces da rota.

Nós, ramos e folhas. Estas convenções fazem parte do estudo de aprendizagem de máquina, particularmente da representação de árvores de decisão. Nós representam desvios (i.e., ramos) condicionados pelos valores de atributos. As folhas são as classes esperadas de um conjunto de exemplos.

Pacotes de dados e mensagens (ou pacotes) de roteamento. Os pacotes de dados transportam os dados da aplicação do usuário. Já as mensagens de roteamento são pacotes que transportam essencialmente as informações de controle de roteamento e topológico.

3.2 PREMISSAS DO MECANISMO

O METRUbi incentiva o aproveitamento das estruturas de dados e mensagens já existentes nos protocolos, procurando acrescentar apenas uma maior compreensão das informações da rede, por meio da captura de todos pacotes propagados, filtragens e aprendizagem de máquina e adicionando mais raciocínio interno nos nós quanto às decisões de roteamento. Seguem as principais condições para que o mecanismo opere satisfatoriamente:

- Os nós devem ser capazes de determinar suas velocidades por meio de um sistema de coordenadas relativas, como proposto em (CAPKUN et al., 2002);
- Os nós se movem em velocidades moderadas, excluindo, então, dispositivos veiculares em alta velocidade. Procurou-se criar cenários próximos aos de ambientes urbanos, aonde usuários portam dispositivos como celulares, tablets e etc. Conforme o estudo apresentado em (KNOBLAUCH et al., 1996), as velocidades típicas de pedestres giram em torno de 0 a 3 m/s. Foram contemplados cenários com velocidades de até 5 m/s para considerar as transições de ambientes veiculares, como por exemplo, um usuário chegando ao local em uma bicicleta ou ônibus;
- O protocolo de roteamento deve incluir, em campos apropriados da camada IP, o caminho completo da rota e a velocidade de cada nó intermediário. No caso do protocolo DSR [16], o caminho a ser percorrido pelos pacotes já é conhecido desde o nó de origem por meio do *Source Route*;
- Os nós egoístas deixam de cooperar apenas no repasse de pacotes de dados. Já os pacotes de roteamento (Route Request, Route Reply, Hello, etc) são encaminhados normalmente, visto que demandam menos recursos dos nós;
- Os nós em modo *sleep* desativam completamente suas interfaces em intervalos cíclicos, deixando de repassar pacotes de roteamento e de dados;
- Não existem nós maliciosos na rede. Assim podemos garantir a integridade das informações de roteamento, bem como os atributos verificados nas rotas;
- Os nós não alteram suas identidades, tendo um endereço único e imutável durante toda existência na rede para evitar inconsistências nas tabelas de roteamento;

[16] Os experimentos foram realizados com o protocolo DSR que já possui boa parte das premissas expostas.

- Todos os pacotes são transmitidos em *broadcast*, podendo ser recebido por todos os nós da vizinhança. Essa é uma propriedade inerente das comunicações em meios sem fio;
- Os nós operam em modo promíscuo recebendo todos os pacotes transmitidos por outros nós a seu alcance e podem determinar o endereço de quem originou cada pacote;
- Os enlaces são bidirecionais, permitindo que ao mesmo tempo que o nó A transmita algo para o nó B, o nó B também possa transmitir para o nó A. Essa propriedade é sempre válida quando se usa o protocolo de camada MAC IEEE 802.11 (MAN; SOCIETY, 2007);
- Os nós contam com um sistema de vigilância dos nós vizinhos. Assim podem saber se houve ou não descarte de pacotes. Uma sugestão prática e leve é o uso do esquema *Watchdog* proposto em (MARTI et al., 2000). Para divulgar o sucesso nos repasses, o Watchdog faz uso de *acknowledgements* de MAC ou de *acknowlegements* passivos que requerem os enlaces bidirecionais (JUBIN; TORNOW, 1987) .

3.3 MÉTRICAS DE CONFIANÇA DA ROTA

Esta seção apresentará as métricas de confiança sugeridas pelo mecanismo. As descrições e formas de cálculo são relatadas nas subseções que seguem. A rotina de cálculo das métricas em linguagem Parsec pode ser encontrada no Apêndice B.

3.3.1 Atividade da Rota - Act_{path}

A ideia da métrica de atividade é dar uma noção do quanto o nó está presente na rede. Esse parâmetro foi pensado para contornar problemas com nós em modo *sleep* ou que apresentem defeitos ou que simplesmente se desliguem da rede. A atividade da rota é calculada em relação aos dados de atividade dos demais nós presentes no *cache* de rotas. Pode ser entendida como a quantidade de interações de um nó com as tarefas da rede. Com isso, se a rede estiver ociosa, não haveria penalização de rotas com nós inativos.

Para calcular a atividade da rota, primeiramente deve ser calculada a atividade de cada nó intermediário. A Equação 3.1 mostra como fazer o cálculo da atividade do nó. $Frwd_{data}$ é a quantidade de pacotes de dados encaminhados pelo nó i e $Frwd_{routing}$ é

a quantidade de pacotes de roteamento encaminhados pelo nó i. Essa diferenciação se vale da possibilidade de um nó encaminhar seletivamente apenas um dos tipos.

$$Act_i = Frwd_i = Frwd_{data} + Frwd_{routing} \tag{3.1}$$

Adicionalmente, deve ser calculada a atividade média da rede conforme mostrado na Equação 3.2, aonde N é a quantidade de nós armazenados no Registrador de Confiança e Act_i é a atividade do nó intermediário i.

$$Act_{avg} = \frac{\sum_{i=1}^{N} Act_i}{N} \tag{3.2}$$

Foram definidos dois limiares para determinar a atividade da rota. O limiar mínimo de atividade Act_{low} e o limiar máximo de atividade Act_{high} são calculados conforme as Equações 3.3 e 3.4, respectivamente. Sendo que Act_{min} é a atividade do nó menos ativo do Registrador de Confiança, já Act_{max} é a atividade do nó mais ativo.

$$Act_{low} = \frac{Act_{avg} + Act_{min}}{2} \tag{3.3}$$

$$Act_{high} = \frac{Act_{max} - Act_{avg}}{2} \tag{3.4}$$

Por fim, a atividade da rota Act_{path} é calculada pela média de atividade dos nós intermediários que participam da rota, conforme mostrado na Equação 3.5. Act_i é a atividade do nó i e R é quantidade de nós intermediários presentes na rota.

$$Act_{path} = \frac{\sum_{i=1}^{N} Act_i}{R} \tag{3.5}$$

Como a métrica Act_{path} será tomada como um atributo das árvores de decisão do METRUbi, seus valores devem ser discretizados, conforme o esquema da Equação 3.6. Quinlan (1993) afirma que atributos com valores contínuos podem gerar um número excessivo de regras e reduzir a capacidade de generalização das árvores de decisão.

Dessa forma foi adotado que uma rota poderá ter atividade igual a 1 (baixa atividade), 2 (média atividade) ou 3 (alta atividade). Por exemplo, uma rota com $Act_{path} = 1$ será considerada de baixa atividade, pois se aproxima da atividade do nó menos ativo encontrado na rede.

$$Act_{path-d} = \begin{cases} 1 & \text{se } Act_{path} \leq Act_{low}, \\ 2 & \text{se } Act_{low} < Act_{path} \leq Act_{high}, \\ 3 & \text{se } Act_{path} > Act_{high}. \end{cases} \tag{3.6}$$

3.3.2 Cooperação da Rota - Cop_{path}

A cooperação da rota, Cop_{path}, está associada à chance de sucesso no encaminhamento de pacotes para quem a usa. É um processo puramente estocástico que se baseia na observação do histórico de interações entre os nós. A cooperação da rota é determinada a partir da cooperação dos nós intermediários. A Equação 3.7 calcula a cooperação Cop_i do nó i, sendo que $Frwd_i$ é a quantidade de pacotes da dados encaminhados pelo nó i e $Drop_i$ é a quantidade de pacotes de dados descartados pelo nó i.

$$Cop_i = \frac{Frwd_i}{Frwd_i + Drop_i} \tag{3.7}$$

A cooperação da rota Cop_{path} consiste no produto das probabilidades de encaminhamento de cada nó, conforme mostrado na Equação 3.8, onde C é a quantidade de nós intermediários.

$$Cop_{path} = \prod_{i=1}^{C} Cop_i \tag{3.8}$$

Os resultados de Cop_{path} também foram discretizados, podendo assumir o valor 1 (baixa confiança), 2 (média confiança) e 3 (alta confiança), conforme a Equação 3.9.

$$Cop_{path-d} = \begin{cases} 1 & \text{se } Cop_{path} \leq 0,4, \\ 2 & \text{se } 0,4 < Cop_{path} \leq 0,8, \\ 3 & \text{se } Cop_{path} > 0,8. \end{cases} \tag{3.9}$$

3.3.3 Mobilidade da Rota - Mob_{path}

A mobilidade da rota, Mob_{path}, pode ser encarada como um indicador da estabilidade. À medida que os nós se movimentam, diversos efeitos indesejados ocorrem, tais como: quebra de enlaces, ruídos e atenuações nos sinais, problemas de terminal exposto e de terminal escondido (JAYASURIYA et al., 2004) e outros. Uma rota mais estática oferece melhores condições para o tráfego de pacotes além de maior previsibilidade.

Para calcular a mobilidade da rota, primeiramente deve ser obtida a velocidade dos nós intermediários Spd_i mostrado na Equação 3.10, aonde Δd_i é a variação da distância do nó i e Δt é o tempo gasto no trajeto.

$$Spd_i = \frac{\Delta d_i}{\Delta t} \tag{3.10}$$

A mobilidade da rota é a média das velocidades dos nós intermediários que participam da rota. A Equação 3.11 denota o calculo da mobilidade da rota Mob_{path}, na qual C é a quantidade de nós intermediários.

$$Mob_{path} = \frac{\sum_{i=1}^{C} Spd_i}{N} \tag{3.11}$$

Como nas métricas anteriores, os valores devem ser discretizados conforme a Equação 3.12 para o cálculo de Mob_{path-d}. Os limiares de 1 e 2 m/s foram pensados para um ambiente urbano conforme as sugestões obtidas em (GERLA et al., 2005) e (KNOBLAUCH et al., 1996).

$$Mob_{path-d} = \begin{cases} 1 & \text{se } Mob_{path} \leq 1\ m/s, \\ 2 & \text{se } 1\ m/s < Mob_{path} \leq 2\ m/s, \\ 3 & \text{se } Mob_{path} > 2\ m/s. \end{cases} \tag{3.12}$$

3.3.4 Distância da Rota - Dst_{path}

Por último, a distância da rota, Dst_{path}, do nó origem até o nó destino. Essa medida para influenciar na confiança da rota, já que quanto mais saltos, maior a probabilidade de conter nós mal comportados, bem como, de ocorrer falhas. A Equação 3.13 mostra o cálculo do número de saltos Dst_{path} a partir do número M de nós envolvidos na comunicação, incluindo a origem e o destino.

$$Dst_{path} = M - 1 \tag{3.13}$$

3.4 ARQUITETURA DO METRUBI

A arquitetura do METRUbi foi pensada para proporcionar ao protocolo de comunicação um mecanismo de avaliação de rotas com as seguintes características:

- **Baixo *overhead*:** o mecanismo não requer mensagens adicionais de controle. A criação das regras de decisão é feita com a observação dos nós vizinhos a partir de *acknownledge* passivo e das mensagens nativas do protocolo de roteamento. Apesar do mecanismo não prever mensagens para divulgação das informações dos nós por toda rede, na prática isso não seria requerido, visto que cada nó poderia fazer a avaliação usando apenas o que conhece de seu "pequeno mundo";
- **Aprendizagem em tempo de execução:** é utilizada a aprendizagem de máquinas por meio do treinamento de árvores de decisão, que é feito em tempo de execução, considerando as informações coletadas até o momento. A margem de erro do mecanismo de avaliação será reduzida com o passar do tempo. Para evitar o *overtraining* e exclusão perpétua dos nós mal comportados, as regras são remontadas a cada ciclo de amostragem;
- **Adaptativo:** as regras de decisão sobre as rotas se modificam com novas observações. Isso faz com que, em ambientes mais hostis, exista maior critério na seleção das rotas e vice-versa. Além disso, como as métricas adotadas são relativas a média da rede, pode ocorrer a reintegração de nós egoístas em futuras rotas, caso voltem a ter um bom comportamento. Mesmo que algumas informações dos nós sejam desconhecidas, é possível avaliar as rotas, pois as árvores de decisão conseguem inferir nessas situações;

- **Portável:** o mecanismo pode ser instalado em outros protocolos de roteamento, pois não está associado às rotinas nativas do protocolo de roteamento. Para isso, bastaria adequar as mensagens de roteamento, ativar o modo promíscuo e implementar a chamada da função de avaliação no algoritmo de seleção de rotas;
- **Extensível:** novas métricas de confiança podem ser adicionadas ao mecanismo, permitindo outros estudos e aplicações.

Conforme mostrado na Figura 3.1, o mecanismo proposto possui cinco componentes funcionais: Monitor, Registrador de Confiança, Indutor DT, *Cache* de Rotas e Avaliador de Rotas. Nas subseções seguintes serão descritos os papéis de cada componente.

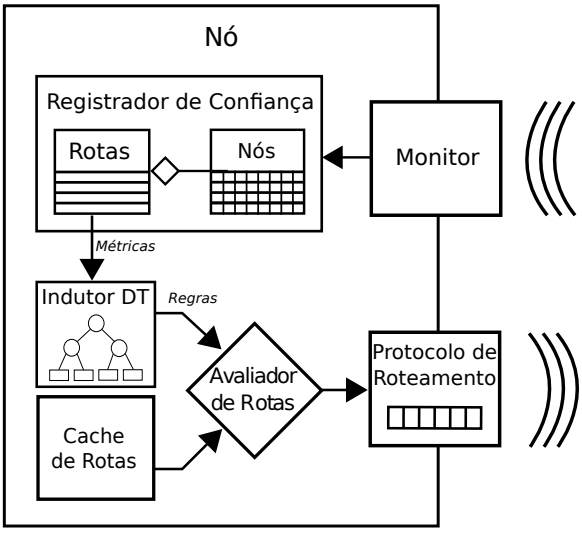

Figura 3.1: Arquitetura do METRUbi

O METRUbi foi implementado na linguagem de programação Parsec (UCLA, 1999) e validado no ambiente de simulação GlomoSim (GLOMOSIM, 1999). Parsec é uma linguagem de simulação baseada em C, desenvolvida pelo Laboratório de Computação Paralela da Universidade da Califórnia (LA), para execução paralela e sequencial de modelos de simulação de eventos discretos.

3.4.1 Registrador de Confiança

O Registrador de Confiança é responsável por cálculo e armazenamento das métricas de confiança dos nós e das rotas. É composto por duas estruturas de dados específicas - Tabela de Métricas de Nós e Tabela de Métricas de Rotas. As informações dos Registrador de Confiança de um dado nó não são divulgadas diretamente aos demais

nós, configurando-se apenas como o ponto de vista particular. A Tabela 3.1 ilustra um conjunto de amostras da Tabela de Métricas de Nós que contém as observações particulares sobre a atividade, cooperação e velocidade dos nós.

Tabela 3.1: Tabela de Métricas de Nós

i	Act_i	Cop_i	Spd_i
A	10	0,6	9,7
B	5	0,9	3,1
C	2	0,5	1,8
..	..	..	..

A partir da Tabela de Métricas de Nós é realizado o cálculo da métricas das rotas usando as Equações de Act_{path} como mostrado na Tabela 3.2.

Tabela 3.2: Tabela de Métricas de Rotas

$Rota$	Dst_{path}	Act_{path}	Cop_{path}	Mob_{path}	$Resultado$
A-D-E	2	1	3	3	bad
A-B-F-E	3	2	3	1	good
A-B-F-E	3	1	1	2	good
..	..	..	..	..	..

3.4.2 Monitor

O componente Monitor é responsável pela captura de pacotes da rede, filtragem das informações relevantes, cálculo das métricas e atualização do Registrador de Confiança. A captura é possível devido à natureza de *broadcast* das redes sem fio e da ativação do modo promíscuo. Também pode descobrir nós mal comportados de outras vizinhanças usando técnicas como o *Watchdog* (MARTI et al., 2000).

Os critérios da filtragem do Monitor dependem do protocolo de roteamento utilizado. No caso do protocolo DSR, usado nos experimentos de validação, as filtragens e atualizações foram feitas conforme mostrado nos esquemas da Figura 3.2. As primeiras linhas das tabelas denotam as inferências por meio do *Source Route* recebido em cada nó, e as segundas linhas, as inferências por monitoramento passivo. As notações $act(i)$ e $cop(i)$ se referem aos valores das métricas Act_i e Cop_i do nó i, respectivamente. Vale ressaltar que as métricas de um nó i podem ter valores distintos nos diferentes nós, pois as inferências dependem do posicionamento na rede.

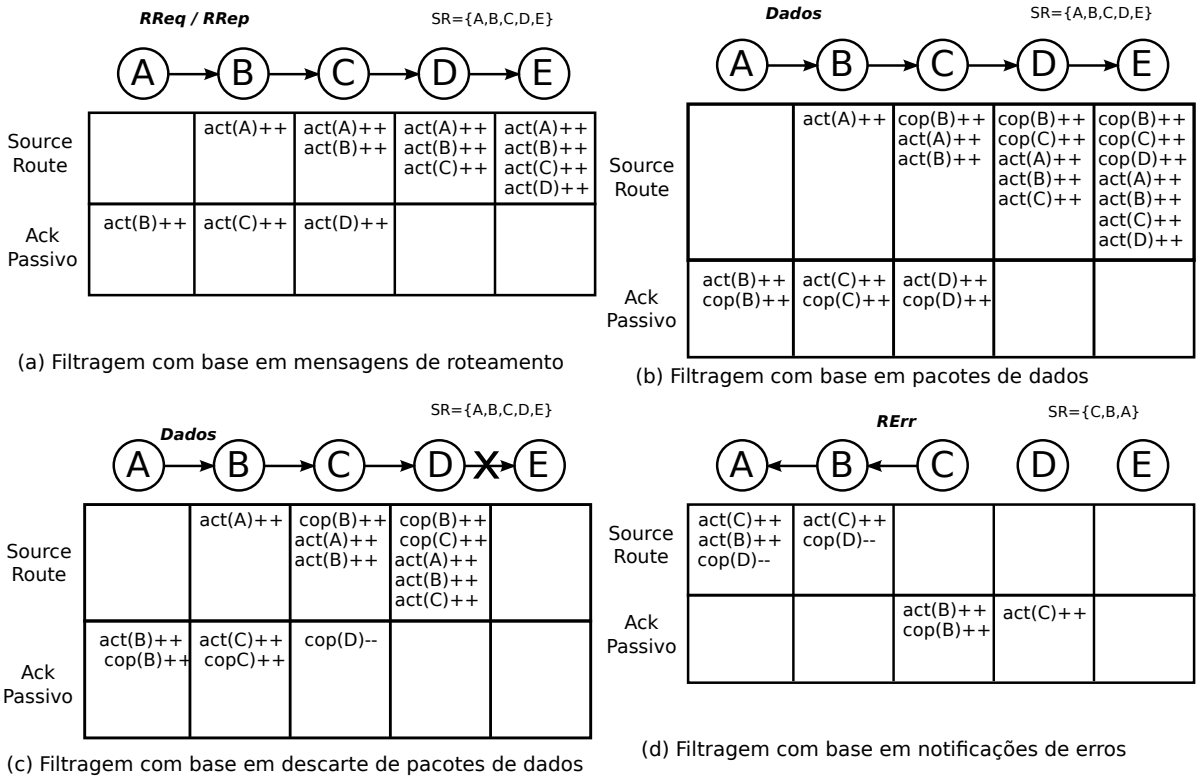

Figura 3.2: Esquema de filtragem de informações do DSR pelo componente Monitor

No esquema (a) da Figura 3.2, mensagens RREQ ou RREP são encaminhadas pelos nós. Como o *Source Route* acumula os IDs dos nós por onde passou, o próximo nó pode incrementar os valores de atividade Act_i de todos os nós relacionados na rota, pois pode inferir que tais nós encaminharam as mensagens corretamente. Paralelamente, por meio de *ack* passivo, o nó anterior pode vigiar o seguinte e, em caso positivo, incrementar o seu valor de atividade. No esquema (b) da Figura 3.2, os nós encaminham pacotes de dados. Com a mesma análise anterior, os nós podem inferir sobre a cooperação Cop_i e sobre a atividade Act_i dos demais nós do caminho. No esquema (c), o nó C pode decrementar os valores de cooperação do nó D por vigilância. Após isso, no esquema (d), o nó C enviará uma mensagem de *Route Error* ao nó de origem A. Com isso os demais nós poderão decrementar os valores de cooperação do nó D. Vale ressaltar que os nós vizinhos que não participam da comunicação poderão fazer as mesmas atualizações, visto que as mensagens trafegam em *broadcast*

As mensagens de roteamento e os pacotes de dados devem carregar também as informações de velocidade dos nós que as encaminham. Dessa forma, os nós presentes no *Source Route* e os seus vizinhos poderão manter as mobilidades Mob_i dos demais nós. As distâncias das rotas também podem ser capturadas em todos os pacotes, visto que

sempre acumulam os respectivos saltos.

O campo *Classe* da Tabela de Métricas de Rotas é ajustado para *true* sempre que mensagens *Route Request*, *Route Reply* e pacotes de dados contendo um *Source Route* são capturados na rede, pois pode-se inferir que outrora esses caminhos foram bem-sucedidos. Já as mensagens de *Route Error* ajustam a entrada para $false$.

Em suma, os pacotes de dados e mensagens de roteamento do protocolo DSR são os únicos insumos usados pelo METRUbi para determinar os valores das métricas dos nós e das rotas, segundo as associações que seguem:

- **Mensagens de *Route Request* e *Route Reply*:** indicam quais nós estão ativos na rede, ou seja, que não estão em modo *sleep*, e quais rotas foram bem-sucedidas;
- **Mensagens de *Route Error*:** revelam os nós egoístas, inoperantes e inalcançáveis na rede, bem como as rotas malsucedidas;
- **Pacotes de dados com *Source Route*:** indicam quais nós são cooperativos, isto é, que não são egoístas, e quais rotas foram bem-sucedidas;
- **Todas mensagens e pacotes:** indicam as velocidades dos nós que participaram no encaminhamento e o número de saltos das rotas.

3.4.3 Indutor DT

No componente Indutor DT (*Decision Tree*) opera o algoritmo de aprendizagem de máquina por meio do treinamento de árvores de decisão. Tem o papel de realizar o treinamento e exportar as regras de decisão para o Avaliador de Rotas. Para o treinamento é utilizado o conjunto de amostras armazenadas na Tabela de Métricas de Rotas. O critério de atualização das regras se dá pela quantidade de informações coletadas.

O algoritmo de treinamento de árvores de decisão utiliza uma adaptação do C 4.5. O pseudocódigo apresentado na Figura 3.3 descreve seu funcionamento, considerando que: T é o conjunto de métricas, isto é, $T = \{Act_{path}, Cop_{path}, Mob_{path}, Dst_{path}\}$; C é o conjunto de classes de rotas, isto é, $C = \{true, false\}$; R é o conjunto de amostras da

Se $|R|$ mod $R_{max} \equiv 0$:

$GeraRegras(T, C, R)$:

Se R só possui amostras de uma mesma classe, retorne um único nó de decisão com o valor da classe

Se $T = \{\}$, retorne um único nó de decisão com valor da classe mais frequente

Faça M = métrica com maior ganho de informação $Gain(R, M)$

Sejam V_1, V_2,..,V_n os valores da métrica M

Sejam R_1, R_2,..,R_n os subconjuntos de R particionados pelos valores V_i

Crie um nó de decisão M

Para $i = 1$ até n:

Adicione um ramo ao nó M que corresponda ao teste $M = V_i$

Crie novas regras $GeraRegras(T - \{M\}, S, R_i)$ no ramo

Exporte as regras para o Avaliador de Rotas

Remova todas amostras da Tabela de Métricas de Rotas

Reinicie o processo

Figura 3.3: Pseudocódigo do componente Indutor DT

Tabela de Métricas de Rotas; $|R|$ representa a quantidade de amostras em R; e R_{max} é a quantidade máxima de amostras permitida em R.

Para calcular o ganho de informação de uma certa métrica M, primeiramente deve ser calculada a entropia total do conjunto de amostras R, conforme a Equação 3.14, na qual c é o número de valores possíveis da classe C, $freq(C_j, R)$ é uma função que contabiliza o número de ocorrências do valor C_j da classe no conjunto de amostras R e $|R|$ é a quantidade de amostras de R. O ganho de informação da métrica M em relação ao conjunto de amostras R é calculado segundo a Equação 3.15.

$$Info(R) = \sum_{j=1}^{c} -\frac{freq(C_j, R)}{|R|} \cdot log_2 \frac{freq(C_j, R)}{|R|} \tag{3.14}$$

$$Gain(R, M) = Info(R) - \sum_{m \in V_1..V_n} \frac{|R_m|}{|R|} \cdot info(R_m) \tag{3.15}$$

3.4.4 *Cache* de Rotas

Estrutura de dados que armazena as rotas de acordo com cada protocolo de roteamento. Protocolos como o DSR, DSDV e AODV já a implementam nativamente. A Figura 3.4 exibe o código dessas estruturas.

```
1  typedef struct RCE
2  {
3      NODE_ADDR destAddr;
4      int hopCount;
5      NODE_ADDR path[DSR_MAX_SR_LEN];
6      struct RCE *prev;
7      struct RCE *next;
8
9  } DSR_RouteCacheEntry;
10
11 typedef struct
12 {
13     DSR_RouteCacheEntry *head;
14     int count;
15
16 } DSR_RouteCache;
17
```

Figura 3.4: Estruturas de dados do *cache* de rotas do protocolo DSR

O protocolo DSR mantém as entradas `DSR_RouteCacheEntry` em uma lista ordenada, primariamente, pelo endereço do nó de destino `destAddr` (linha 3) e, sem seguida, pelo número de saltos `hopCount` (linha 4). Sendo assim, a primeira entrada encontrada para um determinado endereço de destino, será, necessariamente, a mais curta. Caso existam duas entradas com o mesmo número de saltos para um destino, o protocolo prioriza a mais recente. A lista duplamente encadeada, por sua vez, é mantida na estrutura `DSR_RouteCache` (linha 11) que representa o *Cache* de rotas.

3.4.5 Avaliador de Rotas

Faz a avaliação da confiança das rotas com base nas regras de decisão. Sua saída é binária, podendo ser 0 para uma rota avaliada como ruim e 1 para uma rota avaliada como boa. Pode ser integrado aos protocolos de roteamento para que façam melhores escolhas das rotas. Caso haja mais de uma opção boa, pode ser avaliado um segundo critério, como por exemplo, o menor número de saltos.

Para fins de otimização, o Avaliador de Rotas foi implementado no protocolo DSR apenas reordenando as entradas do cache com um novo critério de confiança. A variável

`trust` (linha 5) foi adicionada conforme a Figura 3.5. O novo código de ordenação pode ser encontrado no Apêndice C.

```
1  typedef struct TRCE
2  {
3      NODE_ADDR destAddr;
4      int hopCount;
5      int trust;
6      NODE_ADDR path[TDSR_MAX_SR_LEN];
7      struct TRCE *prev;
8      struct TRCE *next;
9
10 } TDSR_RouteCacheEntry;
```

Figura 3.5: Entrada de *cache* modificada para o Avaliador de Rotas

3.5 FUNCIONAMENTO DO METRUBI

Como foi explicado na Seção 3.2, todo pacote deve carregar as informações por onde passou e a velocidade dos nós que já fizeram o encaminhamento. Isso foi feito com a adição de novos campos de dados na camada IP. No caso do DSR, já existe o campo *Source Route* que pôde ser utilizado.

O Monitor tem o papel de capturar as amostras que servirão de base para o algoritmo de aprendizagem de máquina. Quando um pacote chega ao nó de destino, o Monitor do nó de destino e de todos os nós vizinhos armazena as informações da rota no Registrador de Confiança. O Registrador de Confiança tomará duas ações: i) Modificará a Tabela de Métricas de Nós, incrementando o Act_i e o Cop_i e atualizando o Spd_i de todos nós intermediários que constam na rota, pois encaminharam com sucesso o pacote; e ii) Adicionará uma nova entrada na Tabela de Métricas de Rotas, fazendo os cálculos das métricas (Act_{path}, Cop_{path}, Mob_{path} e Dst_{path}) com base no que consta na Tabela de Métricas de Nós. Nesse caso, a classe da rota será $sucesso = true$.

Quando um pacote é descartado por um nó intermediário, o Monitor dos nós vizinhos armazena a rota no Registrador de Confiança. O Registrador de Confiança tomará três ações: i) Incrementará o Act_i e o Cop_i e atualizará o Spd_i dos nós intermediários com exceção do último nó que foi o descartador; ii) Decrementará o Cop_i do último nó que consta na rota (nó descartador); e iii) Adicionará uma nova entrada na Tabela de Métrica de Rotas, fazendo os cálculos das respectivas métricas (Act_{path}, Cop_{path}, Mob_{path} e Dst_{path}). Nesse caso, a classe da rota será $sucesso = false$.

Após o preenchimento da Tabela de Métricas de Rotas, as amostras coletadas são lidas pelo Indutor DT. O Indutor primeiramente faz a discretização dos valores conforme foi explicado na Seção 3.3. Em seguida, faz o treinamento da árvore de decisão usando o algoritmo C 4.5. As regras obtidas são então exportadas para o Avaliador de Rotas.

Deve ser definido um critério de expiração das regras do Avaliador. O critério de tempo poderia ser usado, porém acreditamos que, em redes mais ociosas, haveria processamento extra sem mudança significativa nas regras, e, em redes mais ativas, poderia haver consumo excessivo de memória, o que não é interessante para os dispositivos móveis. A Tabela de Métricas de Nós tem o tamanho estimado em $N \times 24$ *bytes*, aonde N é o número de nós da rede. Já a Tabela de Métricas de Rotas tem o tamanho crescente e cada entrada consome cerca de 29 bytes. Sugerimos o critério de parada com base no tamanho da Tabela de Métricas de Rotas conforme o esquema mostrado na Equação 3.16, aonde R é a quantidade corrente de entradas da Tabela de Rotas e R_{max} é o número máximo de entradas até a expiração das regras do Avaliador de Rotas.

$$R \bmod R_{max} \equiv 0 \tag{3.16}$$

Quando se atinge o critério de parada, o Indutor DT faz a leitura das rotas armazenadas e executa o treinamento da árvore a fim de gerar novas regras para o Avaliador de Rotas. Após isso, a Tabela de Métricas de Rotas é apagada e se inicia um novo ciclo de amostragem. Isso torna o algoritmo do METRUbi mais adaptado as novas circunstâncias, permitindo, por exemplo, o reingresso de nós que já foram mal comportados nas rotas.

3.6 COMPARAÇÃO COM OUTROS ESQUEMAS DE CONFIANÇA EM REDES *AD HOC*

Embora o METRUbi use um algoritmo inovador para orientar as decisões de roteamento em ambientes de computação ubíqua, pode ser feita uma análise comparativa de seus componentes funcionais, indicando as possíveis vantagens frente aos esquemas *Pathrater*, CONFIDANT, CORE, SORI e OCEAN. Não foi possível fazer uma comparação direta dos resultados de desempenho da rede, pois as os cenários experimentados foram distintos. Por exemplo, os esquemas não trataram as redes com nós em modo *sleep*, problemas de mobilidade e níveis acentuados de egoísmo.

O esquema *Pathrater* tem similaridades com o METRUbi em relação ao tratamento de nós egoístas com base nas informações da métrica de cooperação. Primeiro porque implementa o protocolo *Watchdog* que faz o monitoramento passivo dos nós vizinhos a fim de detectar nós descartadores de pacotes. Segundo porque o *Pathrater* também atribui pontuações aos nós da rede e partir disso calcula a pontuação da rota total.

A arquitetura do CONFIDANT foi uma referência para a concepção da arquitetura do METRUbi. O componente Monitor, por exemplo, desempenha o mesmo papel, porém, no METRUbi, a função de monitoramento dos nós vizinhos foi estendida para analisar os nós que pertencem ao caminho da rota e estão fora do alcance de rádio, como explicado na Figura 3.2. Já o componente Gerenciador de Rotas foi substituído pelo Avaliador de Rotas que, ao invés de interferir no algoritmo de seleção de rotas nativo, apenas reordena o *cache* de rotas.

O METRUbi dispensa a emissão de mensagens de alarme na rede, como faz o CONFIDANT e o SORI, pois aproveita as mensagens nativas dos protocolos de roteamento (e.g., RREQ e RREP) para envio e recepção de suas informações. O mecanismo não exige a convergência das informações da rede, logo cada nó poderá ter um ponto de vista diferente sobre os comportamentos dos demais nós. É provável que essa estratégia reduza o *overhead* em relação aos demais esquemas, mas são necessários estudos comparativos nesse sentido.

A proposta do METRUbi oferece algumas vantagens em relação aos esquemas abordados. O principal benefício é a capacidade de tratamento de problemas de diferentes naturezas (e.g., mobilidade, nível de atividade e cooperação) por meio do emprego de métricas multidimensionais. Ao se analisar várias métricas simultaneamente, pode-se verificar qual contribui mais para o sucesso do roteamento e atribuir pesos apropriados para cada cenário.

O METRUbi determina a melhor métrica por meio da observação dos eventos passados e submissão dessas informações a um sistema de aprendizagem de máquina gerando regras completamente dinâmicas, conforme mostrado no Experimento C (Seção 4.4). Em contraste, os outros esquemas usam regras estáticas e não abordam múltiplos problemas. Como os padrões de comportamento que influenciam o desempenho do roteamento se alteram a todo instante, regras estáticas podem limitar a atuação dos esquemas. Por exemplo, a mobilidade, que, em princípio, reduz o desempenho da rede, pode ser algo benéfico para redes saturadas de nós egoístas, visto que a variação da

topologia pode oportunizar novas rotas, conforme mostrado no Experimento A (Seção 4.2). Esse tipo de percepção não é possível nos outros esquemas.

Outro aspecto comum nos esquemas de confiança é que fazem uso de limiares fixos para determinar se os nós são mal comportados e se as rotas são de baixa qualidade. Porém o METRUbi atualiza seus limiares a cada sessão de treinamento, visando identificar os valores otimizados para as decisões de roteamento. Por exemplo, em redes altamente estáveis e cooperativas, os valores limites crescem para filtrar as rotas que estão acima da média em qualidade. Já, em ambiente instáveis e pouco cooperativos, os limiares se reduzem para garantir a sobrevivência da rede.

Por fim, o METRUbi não implementa recursos de incentivos e penalizações. Assim como o *Pathrater*, apenas usa as informações da rede para pontuar as rotas (de acordo com sua medida de mobilidade, cooperação e atividade), o que aumenta a probabilidade de escolha de caminhos confiáveis. As rotas de baixa qualidade e os nós mal comportados continuam ativos na rede, não sendo rejeitados caso sejam as únicas opções.

4 SIMULAÇÕES E RESULTADOS EXPERIMENTAIS

A metodologia da pesquisa foi o estudo comparativo por meio de simulações computacionais. Os cenários foram modelados para se aproximarem dos ambientes urbanos de computação ubíqua, que possuem velocidades moderadas (entre 0 e 5 m/s), intervalos de pausa e falta de cooperação na rede. Com isso, espera-se comprovar os benefícios da inclusão do METRUbi nos protocolos de roteamento que operem em redes com tais características, bem como verificar suas limitações.

Foram realizados quatro experimentos com coleta de dados e análise dos resultados, usando o simulador de redes Glomosim. Seguem os objetivos e as descrições de cada experimento:

- **Experimento A - Estudo dos efeitos da densidade, do dinamismo, do egoísmo e do modo *sleep* no desempenho da rede.** O objetivo dessa etapa foi a constatação desses efeitos em ambientes de computação ubíqua e a verificação da aplicabilidade das métricas de confiança sugeridas. Foram caracterizados diversos cenários contendo mobilidade, nós egoístas e nós em modo *sleep* em diferentes proporções. Após isso, foram realizadas simulações usando o Glomosim a fim de comparar os resultados. Foi utilizado o *script* de automação das simulações (Apêndice A) para extrair dados de cerca de 2 mil cenários diferentes;
- **Experimento B - Validação das métricas de confiança.** Esse experimento teve a finalidade de verificar os valores das métricas de confiança (atividade, cooperação, mobilidade e distância) e sua relação com os resultados de roteamento. A partir dos resultados, também foi possível inferir sobre os intervalos que deveriam pertencer às classes de valores discretos, conforme proposto nas Equações 3.6, 3.9 e 3.12 do Capítulo 3. Foram coletadas informações de roteamento de 60 cenários com diferentes proporções de nós egoístas e nós em modo *sleep*;
- **Experimento C - Comparação do desempenho com o mecanismo de avaliação da confiança ativado em um ambiente não cooperativo com mobilidade aleatória**: Foi realizada a comparação dos resultados das simulações usando o protocolo DSR padrão e as rotinas do METRUbi com o objetivo de

comprovar seus benefícios e constatar suas limitações em redes *ad hoc* mais espontâneas e sujeitas a particionamentos. Foram estudados 60 cenários de curta duração contendo diferentes proporções de nós egoístas e nós em modo *sleep*. Os padrões de mobilidade dos nós foram completamente aleatórios, incluindo as velocidades e posicionamentos iniciais. Os resultados foram baseados nas médias dos valores coletados nas simulações;

- **Experimento D - Comparação do desempenho com o mecanismo de avaliação da confiança ativado em um ambiente não cooperativo com mobilidade controlada**: A última etapa consistiu na comparação dos resultados das simulações usando o protocolo DSR padrão e as rotinas do METRUbi com o objetivo de comprovar seus benefícios e constatar suas limitações em redes *ad hoc* mais controladas. Foi estudado um cenário de longa duração contendo uma única proporção de nós egoístas e nós em modo *sleep*. O padrão de mobilidade foi orientado por um arquivo de rastreamento (*trace-file*) a fim de evitar particionamentos na rede.

4.1 FERRAMENTAS E RECURSOS UTILIZADOS

Devido ao grande número de simulações, foi essencial a criação de uma ferramenta de automação. Nesse sentido, foram elaborados *scripts* em Bash e AWK que alteram os parâmetros e reiniciam a simulação de forma automatizada, tabulando e armazenando cada resultado coletado. Essa ferramenta é apresentada no Apêndice A, podendo ser aproveitada em outras pesquisas. O Gnuplot foi usado nas plotagens dos resultados e criação dos elementos gráficos.

As subseções que seguem discorrem sobre a adoção do protocolo DSR e do Glomosim como ferramentas de simulação.

4.1.1 Protocolo Dynamic Source Routing

O protocolo DSR despertou o interesse desta pesquisa devido aos seguintes fatores:

- É amplamente documentado e padronizado pela RFC 4728 (JOHNSON et al., 2001a);
- Possui implementações para as principais plataformas de simulação e sistemas operacionais;

- É um protocolo adotado como *baseline* em diversas pesquisas, visto que possui poucos recursos de otimização e se aproxima mais de uma rede *ad hoc* pura, pois não delega papéis diferenciados aos nós da rede;
- Já opera com um *cache* de rotas e seu algoritmo acumula qualquer rota observada pelos nós, conferindo o suporte a múltiplas rotas para um mesmo destino.

As rotinas do METRUbi foram inseridas no protocolo DSR a fim de estudar seus benefícios. Para isso, algumas funções do protocolo DSR foram modificadas. Segue a relação das principais alterações:

- Inclusão das rotinas do Registrador de Confiança em pontos estratégicos do protocolo. As funções de tratamento de mensagens de roteamento e pacotes de dados foram modificadas a fim de popular as informações dos nós e rotas;
- Inclusão das rotinas do Avaliador de Rotas mostradas nos Apêndices B e C;
- Extensão das estruturas de dados das mensagens de roteamento *Route Request*, *Route Reply* e *Route Error* para carregarem as informações de velocidade dos nós.

4.1.2 Simulador de Redes Glomosim

O Glomosim (Global Mobile Information Simulator) é um simulador de ambientes de redes comunicação sem fio e cabeadas (NUEVO, 2004). Ele usa a capacidade de simulação de eventos paralelos discretos fornecida pelo Parsec - uma linguagem de simulação baseada em C. O Glomosim simula redes com até milhares de nós conectados com diferentes recursos que incluem *multicast*, comunicações assimétricas usando difusões via satélite, comunicações em múltiplos saltos sem fio usando redes ad hoc e os protocolos tradicionais da Internet.

Nuevo (2004) expõe que uma técnica de agregação de nós foi introduzida no Glomosim para dar benefícios significantes ao desempenho da simulação. Inicializando cada nó como uma entidade separada inerentemente limita a escalabilidade, porque a demanda por memória aumenta dramaticamente para um modelo com um número grande de nós. Com a agregação de nós, uma simples entidade pode simular muitos nós de rede no sistema. A técnica de agregação de nós implica em aumento do número de nós no

sistema mantendo o mesmo número de entidades na simulação. Assim os nós de rede que uma determinada entidade representa são determinados pela posição física dos nós.

O Glomosim foi adotado nos experimentos por ser uma ferramenta superior para simulações de redes *ad hoc* em termos de produtividade (i.e., simulações mais rápidas) e oferecer um robusto sistema de análise de parâmetros, permitindo detalhar cada evento por camada do TCP/IP. Além disso é amplamente documentado e modularizado, possibilitando uma fácil extensão de suas funcionalidades nativas por meio de *plugins* em linguagem Parsec. Vale ressaltar que o projeto, ainda mantido como software livre, possui uma versão comercial, conhecida como OPNET®, incubada recentemente pela empresa Riverbed®.

Por padrão o Glomosim não tem suporte a nós egoístas e nós em modo *sleep*, o que seria vital para simular os ambientes computação ubíqua. Esse suporte foi implementado, podendo ser aproveitado em outras pesquisas. Além disso, para obter os dados necessários para a plotagem dos gráficos e para treinamento das árvores de decisão do C 4.5, foram criadas novas estatísticas no arquivo de saída *glomo.stat*.

As seguintes modificações na implementação do simulador foram realizadas para atender a metodologia da pesquisa:

- Inclusão do suporte ao comportamento egoísta de forma a descartar pacotes de dados e encaminhar apenas as mensagens de roteamento (*Route Request*, *Route Reply* e *Route Error*);
- Geração aleatória de nós egoístas e nós em modo *sleep* a partir dos parâmetros `NUMBER-OF-SELFISH-NODES`, `NUMBER-OF-SLEEPY-NODES` e `SLEEP_TURN_TIME` adicionado ao arquivo de configuração `config.in`;
- Criação do componente Monitor para que colete e filtre as informações da rede;
- Criação do Registrador de Confiança na camada de rede do nó. Nesse local são calculadas as métricas da rota (atividade, cooperação, mobilidade e distância);
- Inclusão de rotina no carregador (driver) do simulador para que armazene em arquivo texto os dados contidos na Tabela de Métricas de Rotas do Registrador de Confiança e as regras criadas no Indutor DT. Com isso obtivemos uma base de dados para analisar a relevância das métricas nos resultados de roteamento.

4.2 EXPERIMENTO A: ESTUDO DOS EFEITOS DA DENSIDADE, DO DINAMISMO, DO EGOÍSMO E DO MODO *SLEEP* NO DESEMPENHO DA REDE

A Tabela 4.1 reúne os parâmetros usados na simulação do Experimento A. Foram criados 216 cenários e para cada um foram realizadas 10 simulações com diferentes topologias iniciais, resultando em 2.160 combinações. Os resultados obtidos foram calculados com base nas médias dos resultados obtidos.

Tabela 4.1: Parâmetros da simulação da Experimento A

Parâmetro	**Valor**
Tempo de simulação	600 s
Dimensões do terreno	1000 m x 1000 m
Valores de *seed*	3, 9, 17, 22, 31, 44, 58, 65, 77 e 88
Número de nós	50, 100, 150, 200, 250 e 300
Número de nós egoístas	0, 10, 20, 30, 40 e 50
Número de nós em modo *sleep*	0, 10, 20, 30, 40 e 50
Posicionamento dos nós	Aleatório
Modelo de mobilidade	*Random-Waypoint*
Velocidade do nó	0 a 10 m/s
Pausa entre percursos	20 s
Potência do rádio	5.0 dBm
Ganho da antena	0 dB
Protocolo de MAC	802.11
Protocolo de roteamento	DSR (Dynamic Source Routing)
Aplicação	CBR (Constant Bit Rate)

As Figuras 4.1, 4.2, 4.3 e 4.4 mostram os principais resultados obtidos desse estudo.

Pode-se observar na Figura 4.1 que o desempenho é bastante degradado com a presença de nós egoístas. Contudo, em redes de maior densidade, esse efeito é mitigado devido ao maior número de rotas alternativas. Igualmente, a Figura 4.2 revela que o dano causado pelos nós em modo *sleep* pode ser compensado com o aumento da densidade da rede. Esses resultados foram obtidos por meio da fixação da velocidade média dos nós em 1 m/s.

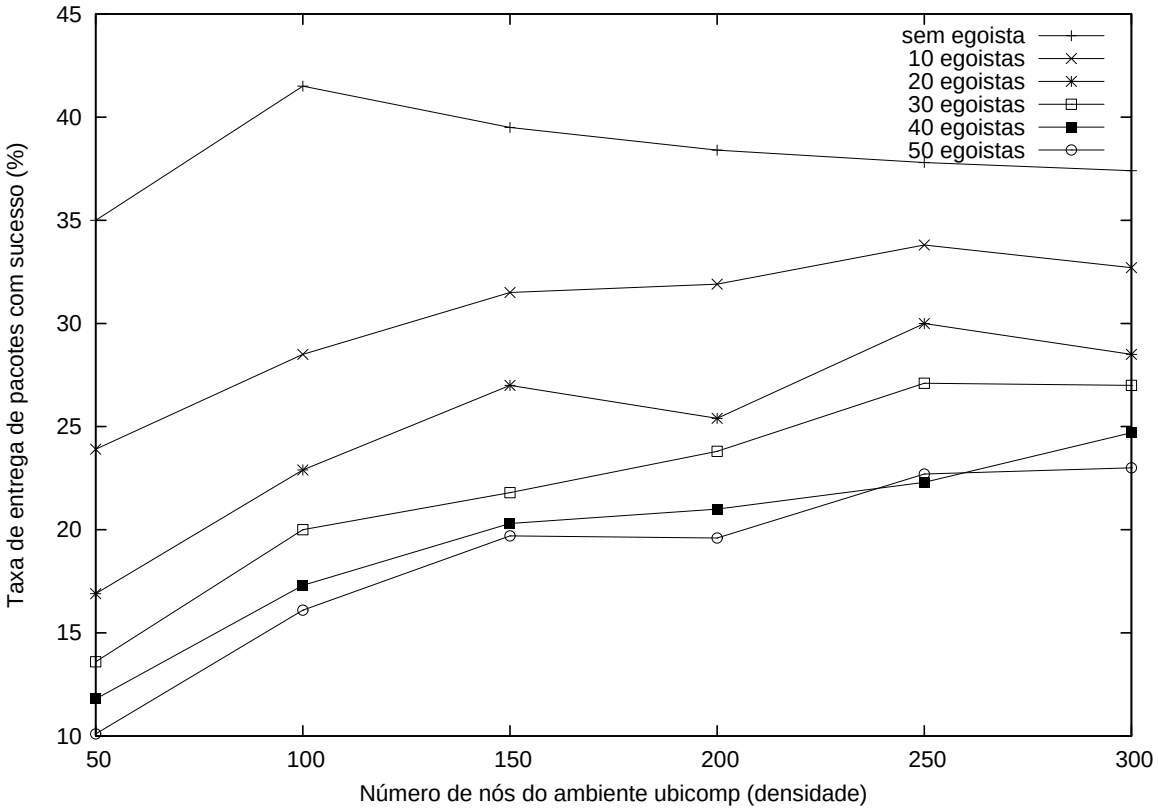

Figura 4.1: Efeito da densidade na taxa de entrega de pacotes com diferentes proporções de nós egoístas

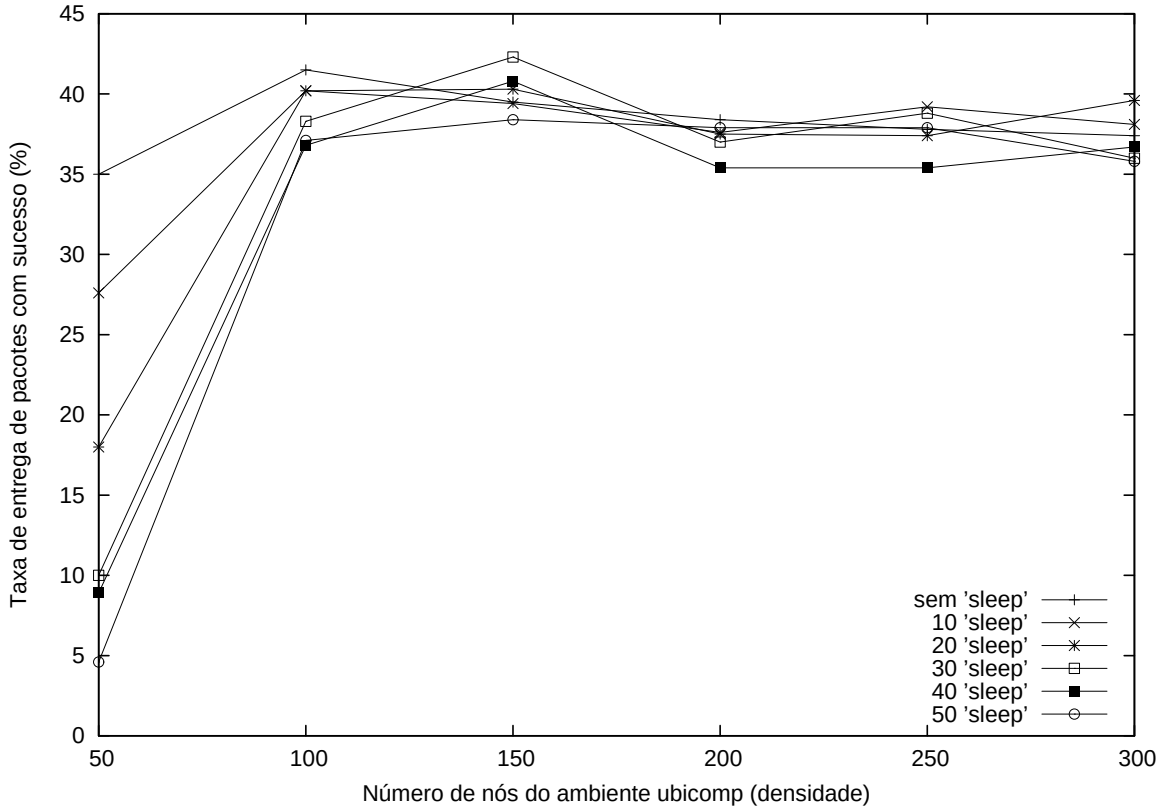

Figura 4.2: Efeito da densidade na taxa de entrega de pacotes com diferentes proporções de nós em modo *sleep*

A Figura 4.3 mostra que o desempenho decai mais rapidamente com a presença de nós egoístas comparado com os nós em modo *sleep*. Como os nós egoístas ainda participam das tarefas de repasse de mensagens de roteamento, a rede acredita que também são voluntários para o encaminhamento de pacotes dados, o que na verdade não se concretiza, ocasionando grande perda de pacotes. Já os nós em modo *sleep* degradam o desempenho por reduzirem o número de nós integrados à rede, reduzindo a vazão sua vazão global. Esses resultados foram obtidos para uma rede com 300 nós com velocidade média de 1 m/s.

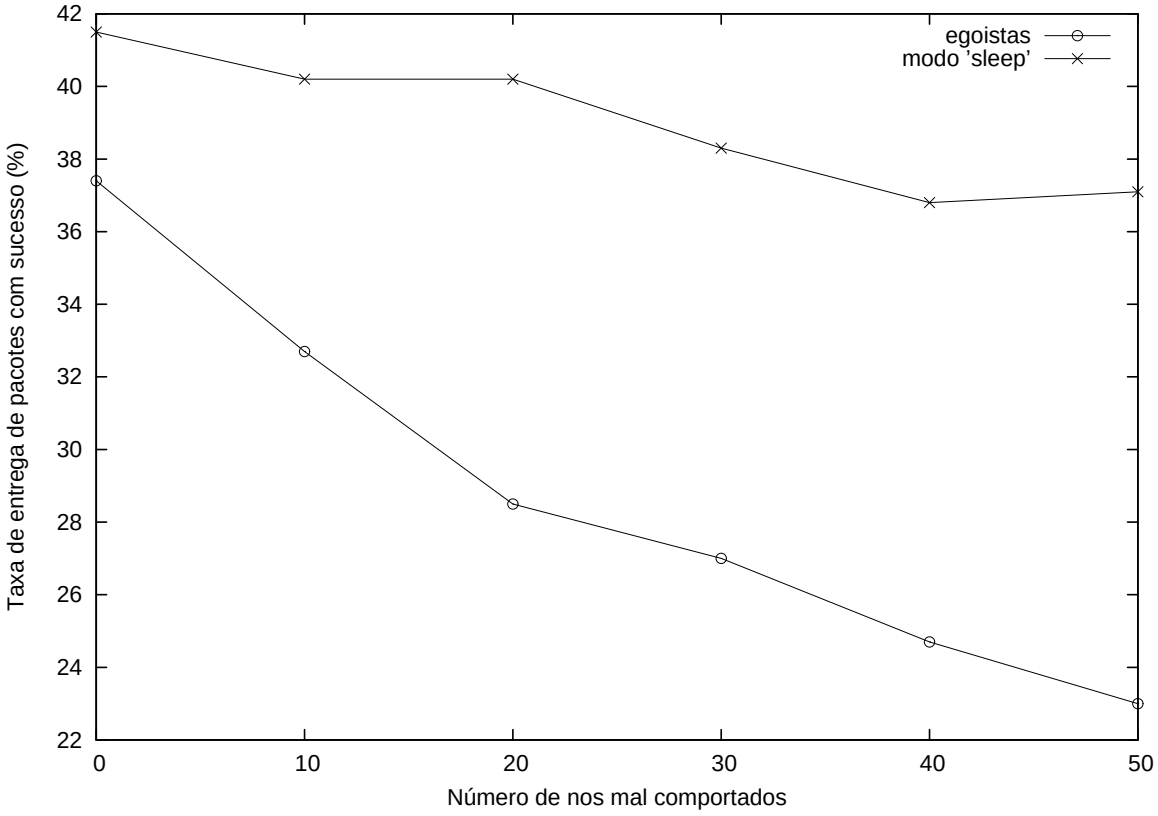

Figura 4.3: Comparação dos efeitos de nós egoístas e nós em modo *sleep*

Por fim, a Figura 4.4 demonstra que o dinamismo é benéfico em redes saturadas por nós mal comportados. Em redes não cooperativas, a mobilidade pode ser uma aliada, visto que a chance de encontrar um nó vizinho cooperativo é ampliada com a movimentação dos nós. Em uma rede completamente estática, os nós imersos em regiões não cooperativas, ficaram particionados da rede. Esse resultado foi obtido para uma rede com 100 nós livre de nós em modo *sleep*.

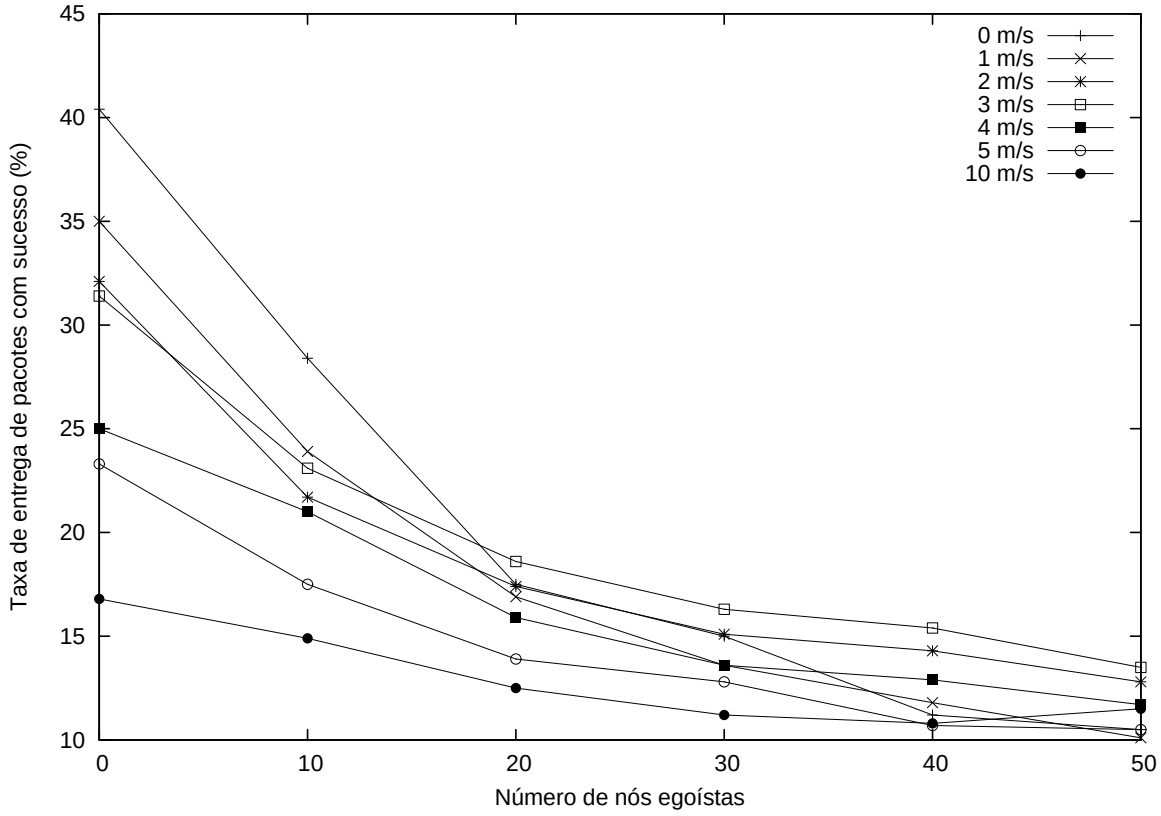

Figura 4.4: Efeito da mobilidade na taxa de entrega de pacotes com diferentes proporções de nós egoístas

4.3 EXPERIMENTO B: VALIDAÇÃO DAS MÉTRICAS DE CONFIANÇA

A Tabela 4.2 reúne os parâmetros usados na simulação. Foram elaborados 6 cenários e para cada um foram realizadas 10 simulações com diferentes topologias iniciais. Os resultados obtidos foram calculados com base nas médias obtidas. O critério de parada foi o tempo com o objetivo de capturar os dados do Registrador de Confiança de todos os nós para a análise das métricas.

Tabela 4.2: Parâmetros da simulação da Experimento B

Parâmetro	**Valor**
Tempo de simulação	300 s
Dimensões do terreno	1000 m x 1000 m
Valores de *seed*	3, 9, 17, 22, 31, 44, 58, 65, 77 e 88
Número de nós	50
Número de nós egoístas	0, 10, 20, 40 e 50
Número de nós em modo *sleep*	0, 10, 20, 40 e 50
Posicionamento dos nós	Aleatório
Modelo de mobilidade	*Random-Waypoint*
Velocidade do nó	0 a 5 m/s
Pausa entre percursos	20 s
Potência do rádio	8.0 dBm
Ganho da antena	0 dB
Protocolo de MAC	802.11
Protocolo de roteamento	DSR (Dynamic Source Routing)
Aplicação	CBR (Constant Bit Rate)

Analisando a dispersão das tuplas de valores presentes na Tabela de Métricas de Rotas para o cenário com 10 nós egoístas, foi observada a relevância de cada métrica no resultado do roteamento. A Figura 4.5 mostra a análise da dispersão da métrica de atividade Act_{path} e cooperação Cop_{path} das rotas. A métrica de cooperação divide bem as amostras a partir do valor 0.8 e a métrica de atividade a partir do valor 50.

A Figura 4.6 faz a mesma análise para a métrica de mobilidade. A mobilidade tem pouca influência nos resultados para o cenário com 10 nós egoístas. Já a Figura 4.7 mostra que o aumento do número de saltos (distância) reduz as chances de sucesso no

roteamento, principalmente para rotas pouco confiáveis. Foram constatadas algumas rotas mal sucedidas a um salto devido às falhas e às movimentações dos nós egoístas e em modo *sleep*.

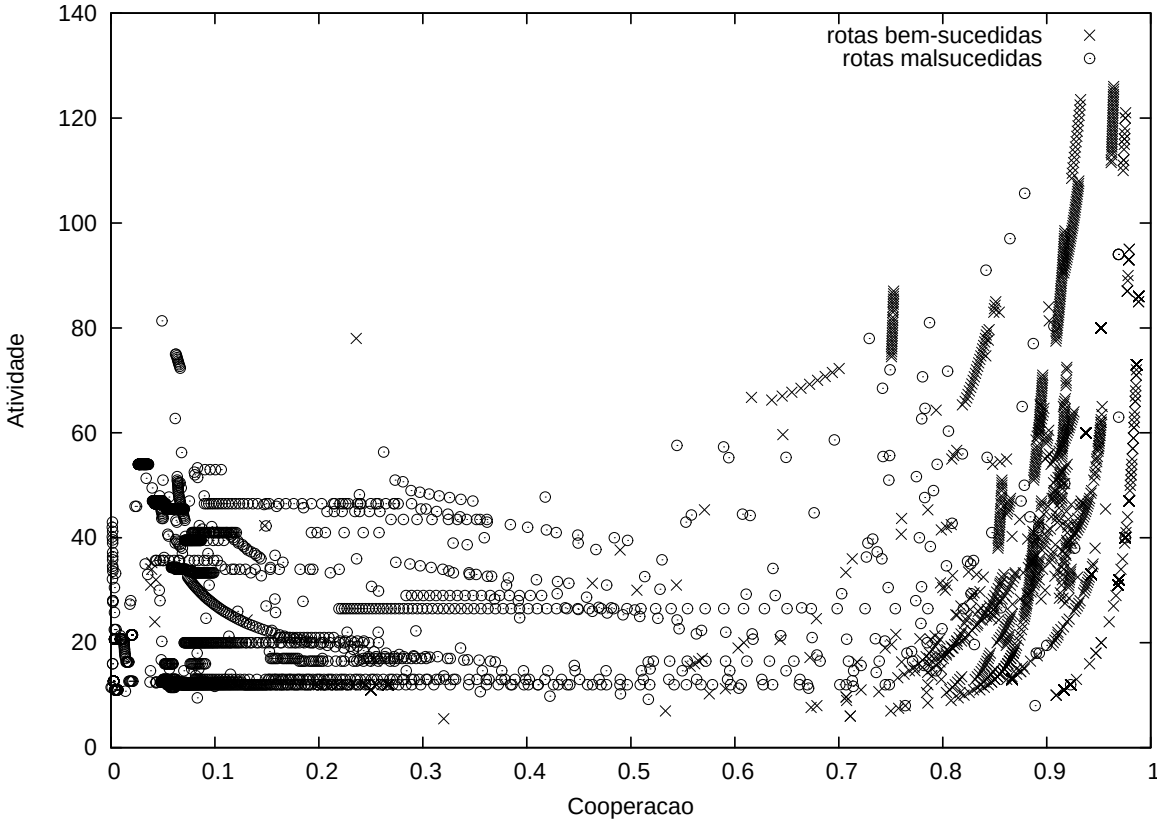

Figura 4.5: Análise da dispersão da atividade e cooperação das rotas

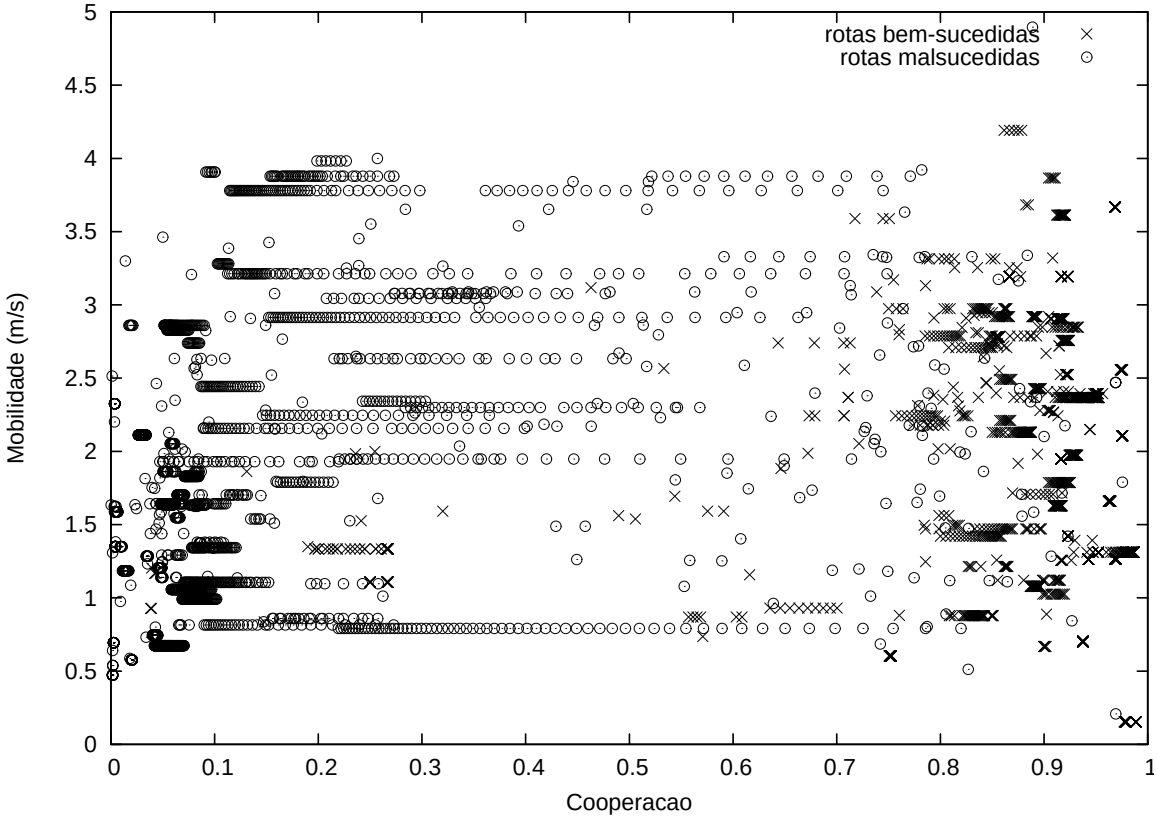

Figura 4.6: Análise da dispersão da mobilidade e cooperação das rotas

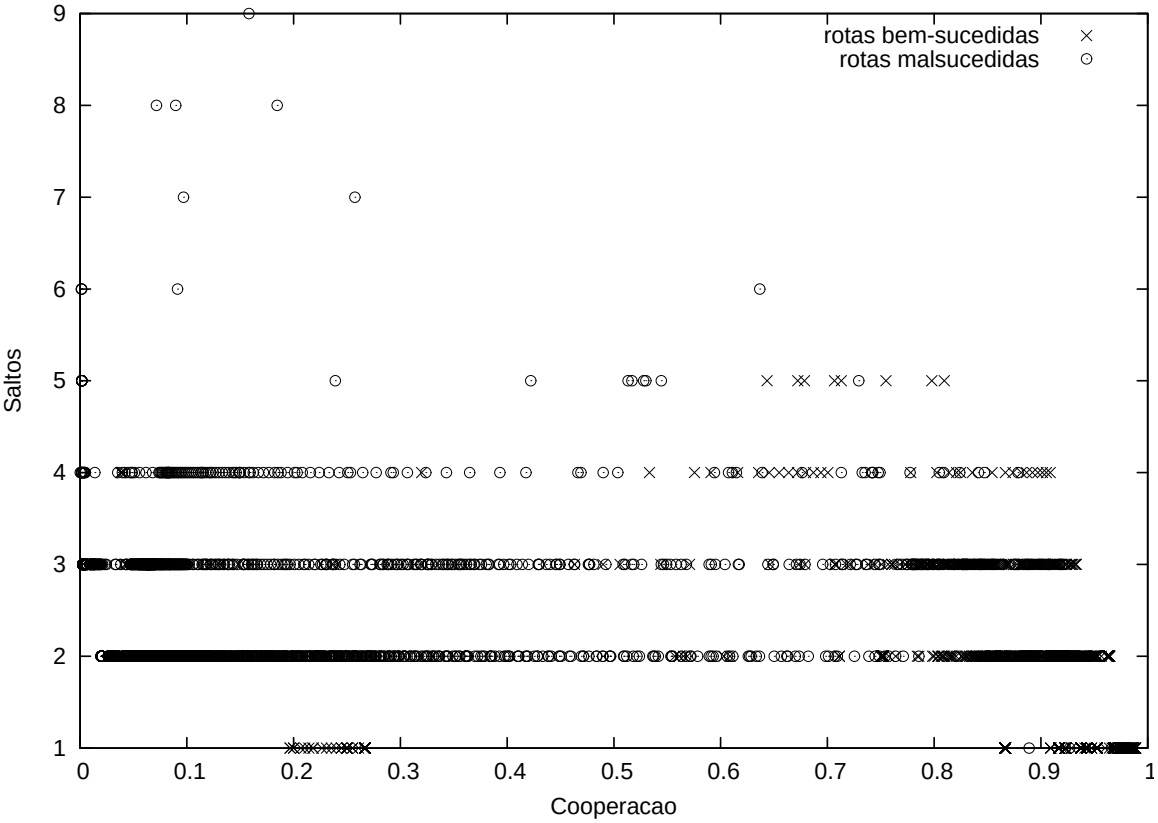

Figura 4.7: Análise da dispersão do número de saltos e cooperação das rotas

4.4 EXPERIMENTO C: COMPARAÇÃO DO DESEMPENHO COM O MECANISMO DE AVALIAÇÃO DA CONFIANÇA ATIVADO EM AMBIENTE NÃO COOPERATIVO COM MOBILIDADE ALEATÓRIA

A Tabela 4.3 reúne os parâmetros usados na simulação. Foram criados 6 cenários e foram realizadas 10 simulações para cada um com diferentes topologias iniciais. Os resultados obtidos foram calculados com base nas médias dos valores.

Tabela 4.3: Parâmetros da simulação da Experimento C

Parâmetro	**Valor**
Tempo de simulação	1200 s
Dimensões do terreno	1000 m x 1000 m
Valores de *seed*	3, 9, 17, 22, 31, 44, 58, 65, 77 e 88
Número de nós	100
Número de nós egoístas	0, 10, 20, 30, 40 e 50
Número de nós em modo *sleep*	10
Permanência em *sleep*	60 s
Posicionamento dos nós	Aleatório
Modelo de mobilidade	*Random-Waypoint*
Velocidade do nó	0 a 5 m/s
Pausa entre percursos	20 s
Potência do rádio	8.0 dBm
Ganho da antena	0 dB
Protocolo de MAC	802.11
Protocolo de roteamento	DSR [+ METRUbi]
Aplicação	CBR (Constant Bit Rate)

A Figura 4.8 apresenta o resultado para cenários com mobilidade entre 0 m/s e 5 m/s,

10% de nós em modo *sleep* e uma gradação de 0% a 50% de nós egoístas. O METRUbi beneficiou o protocolo DSR com até 30% de ganho na taxa de entrega de pacotes conforme mostrado na 4.9.

Vale observar que o mecanismo apresentou momentos de piora em relação ao DSR em cenários com grande número de nós mal comportados. Como foi utilizado um padrão de mobilidade completamente aleatório, foi inevitável a ocorrência de particionamentos na rede, isto é, um grupo de nós se tornam inalcançáveis. Nessa situação o DSR padrão reage mais rapidamente, visto que, ao receber mensagens de quebras de enlaces, dispara um novo processo de descoberta de rotas que inerentemente evita saltos a regiões particionadas. Já com o METRUbi, outra rota do *cache* será utilizada, gerando maior probabilidade de integrar um salto a um nó que já esteja na região particionada.

Para contornar esse ponto fraco do mecanismo, poderia ser incluído um recurso que desabilita o METRUbi para pontos particionados da rede, habilitando apenas em vizinhanças alcançáveis. Esse efeito negativo se mitigou quando foi utilizado um padrão de mobilidade controlado no Experimento D, que evita o particionamento da rede.

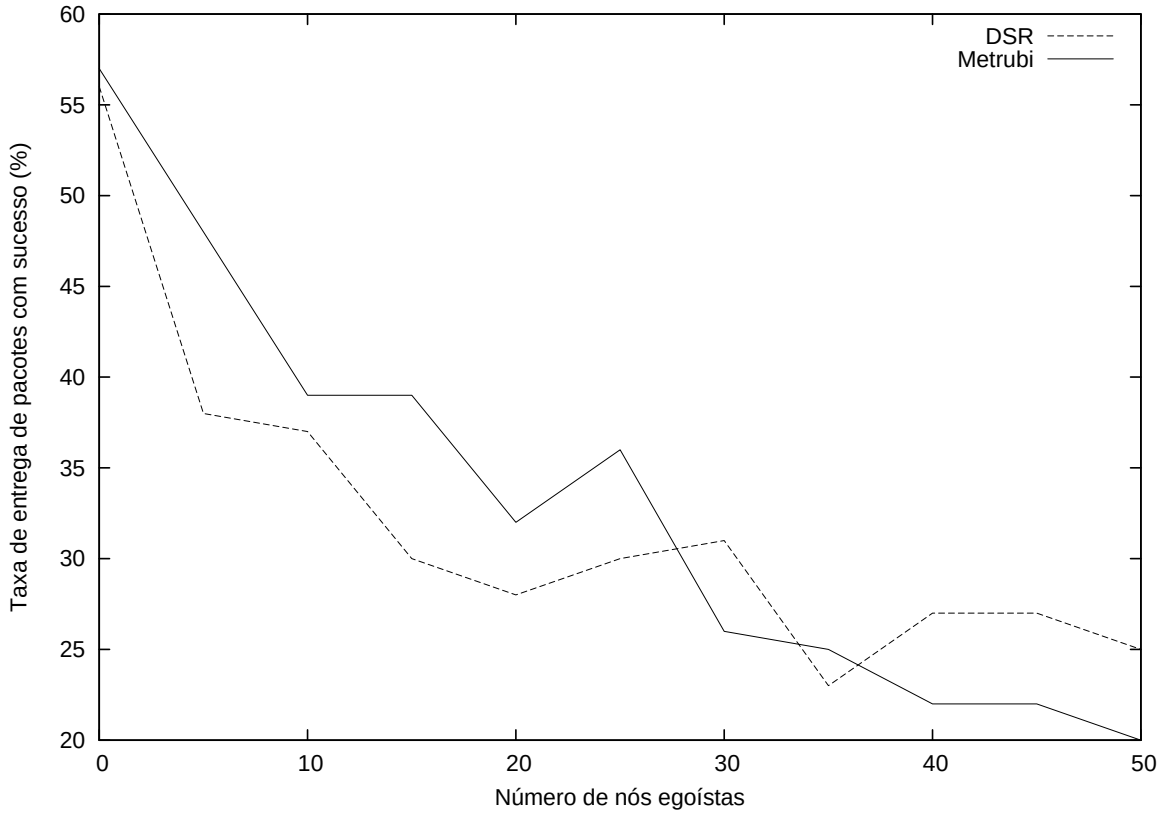

Figura 4.8: Comparação da taxa de entrega de pacotes do DSR padrão com o DSR auxiliado pelo METRUbi com mobilidade aleatória

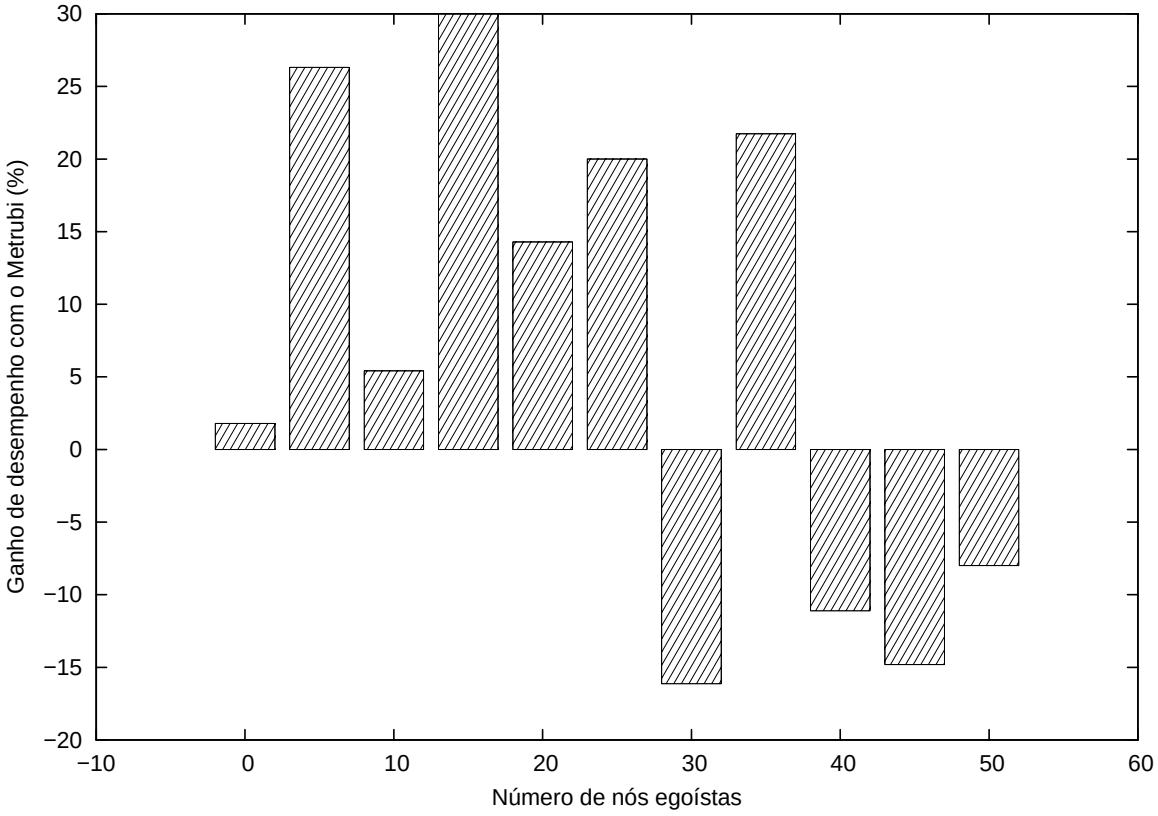

Figura 4.9: Ganho na taxa de entrega proporcionado pelo METRUbi com mobilidade aleatória

Foram obtidas as regras mostradas na Figura 4.10. A única restrição imposta foi em relação ao número de saltos. Ou seja, como não existiam nós egoístas, as métricas de cooperação, atividade e mobilidade se tornaram irrelevantes para o Avaliador de Rotas. Apenas rotas com número de saltos menor ou igual a 4 foram consideradas bem-sucedidas. Os números apresentados à direita de cada regra, são as quantidades de amostras que pertencem a cada classe (i.e., *good* ou *bad*).

```
1 distance <= 4 : good (3960.0)
2 distance > 4 : bad (233.0)
```

Figura 4.10: Regras obtidas nas simulação sem nós egoístas

Como mostrado na Figura 4.11 com 10 nós egoístas, as regras foram mais complexas e, como esperado, o melhor classificador (de menor entropia) foi a cooperação (linha 1). Foi observado que, para rotas de baixa cooperação, só seria admitida comunicação de apenas um salto (linha 3), pois assim não existiriam nós intermediários egoístas. A atividade também comprovou ser um bom classificador para rotas de média cooperação

(linha 7). A mobilidade foi um critério importante para rotas com atividade mediana (linha 9)

```
1  cooperation = high: good (1825.0)
2  cooperation = low:
3  |   distance <= 1 : good (140.0)
4  |   distance > 1 : bad (1810.0)
5  cooperation = middle:
6  |   activity = low: bad (102.0)
7  |   activity = high: good (4.0)
8  |   activity = middle:
9  |   |   mobility = low: good (39.0)
10 |   |   mobility = high: bad (109.0)
11 |   |   mobility = middle:
12 |   |   |   distance <= 2 : good (2.0)
13 |   |   |   distance > 2 : bad (41.0)
```

Figura 4.11: Regras obtidas nas simulação com 10 nós egoístas

Como mostrado na Figura 4.12, o número de saltos se mostrou o melhor classificador (linha 1) em decorrência de uma maior probabilidade de incluir nós egoístas nas rotas *multihop*. O segundo melhor classificador foi a cooperação (linha 4).

```
1  distance <= 1 : good (1322.0)
2  distance > 1 :
3  |   cooperation = low: bad (2401.0)
4  |   cooperation = high: good (238.0)
5  |   cooperation = middle:
6  |   |   mobility = low: bad (23.0)
7  |   |   mobility = high: bad (77.0)
8  |   |   mobility = middle:
9  |   |   |   activity = low: bad (15.0)
10 |   |   |   activity = high: bad (0.0)
11 |   |   |   activity = middle:
12 |   |   |   |   distance > 3 : good (12.0)
13 |   |   |   |   distance <= 3 :
14 |   |   |   |   |   distance <= 2 : good (16.0)
15 |   |   |   |   |   distance > 2 : bad (24.0)
```

Figura 4.12: Regras obtidas nas simulação com 20 nós egoístas

Apenas o número de saltos e a confiança são considerados em ambientes com alta presença de nós egoístas (linhas 1 e 5) como mostrado na Figura 4.13.

```
1 distance <= 1 : good (1189.0)
2 distance > 1 :
3 |   cooperation = low: bad (2301.0)
4 |   cooperation = middle: bad (223.0)
5 |   cooperation = high: good (185.0)
```

Figura 4.13: Regras obtidas nas simulação com 30 nós egoístas

Em ambientes bastante hostis, as regras se tornaram bem rígidas, permitindo apenas rotas com comunicação de um salto, como mostrado na Figura 4.14. Tanto para 40 quanto para 50 nós egoístas, o único classificador foi o número de saltos (linha 1).

```
1 distance <= 1 : good (1236.0/3.9)
2 distance > 1 : bad (3039.0/11.9)
```

Figura 4.14: Regras obtidas nas simulação com 40 e 50 nós egoístas

4.5 EXPERIMENTO D: COMPARAÇÃO DO DESEMPENHO COM O MECANISMO DE AVALIAÇÃO DA CONFIANÇA ATIVADO EM AMBIENTE NÃO COOPERATIVO COM MOBILIDADE CONTROLADA

A Tabela 4.4 reúne os parâmetros usados na simulação. O Experimento D analisou o desempenho do METRUbi em um cenário mais longo com a presença concomitante de nós egoístas e nós em modo *sleep* com velocidades entre 0 e 5 m/s. Foi usado um padrão controlado de mobilidade e posicionamento inicial por meio de arquivo do tipo *trace-file*.

Tabela 4.4: Parâmetros da Simulação do Experimento D

Parâmetro	**Valor**
Tempo de simulação	8000 s
Dimensões do terreno	1000 x 1000 m
Número de nós da rede	100
Número de nós egoístas	20
Número de nós sleep	30
Permanência em *sleep*	60 s
Posicionamento inicial	Especificado
Modelo de mobilidade	*Trace-file*
Velocidade dos nós	0 a 5 m/s
Pausa de percurso	20 s
Potência de rádio	6.0 dBm
Ganho da antena	0 dB
Protocolo MAC	802.11
Protocolo de roteamento	DSR [+ Metrubi]
Aplicação de teste	CBR

As Figuras 4.15 e 4.16 demonstram os resultados obtidos no Experimento D. Pode-se perceber que o mecanismo contribuiu para melhoria da taxa de entrega de pacotes em quase todo período de simulação, chegando a ganhos de até 25 %.

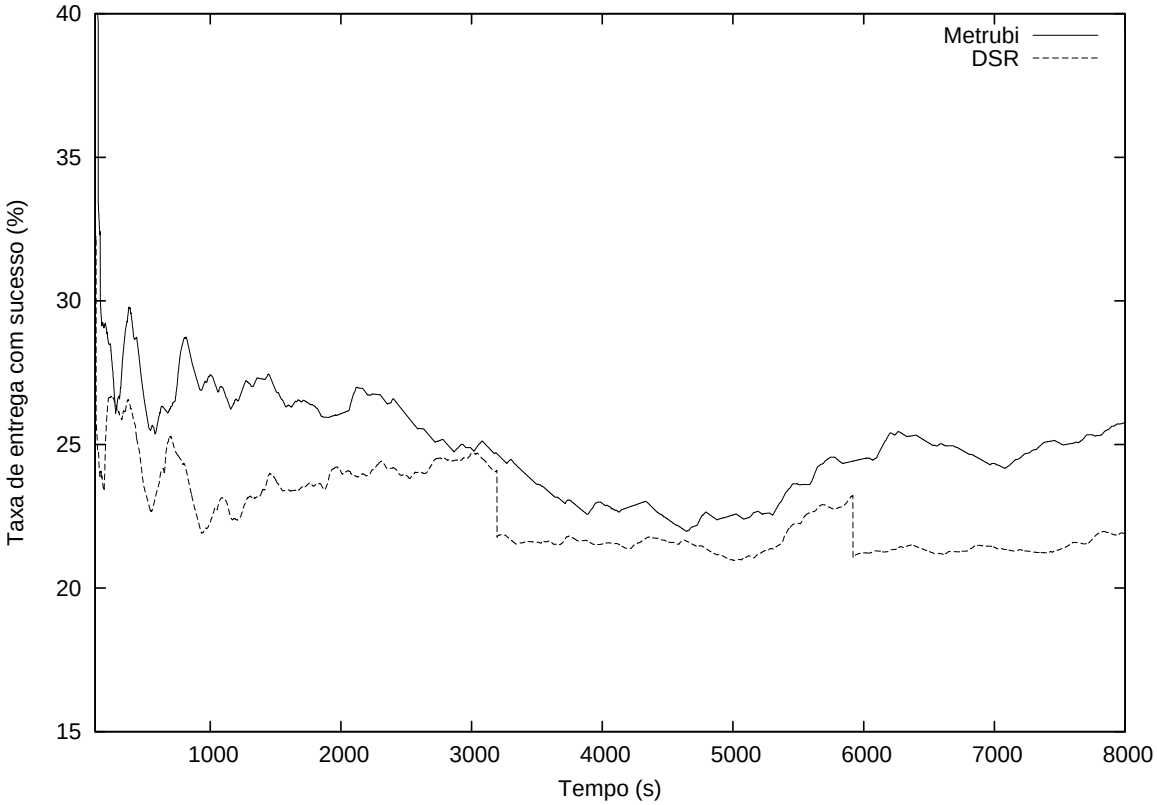

Figura 4.15: Comparação da taxa de entrega de pacotes do DSR padrão com o DSR auxiliado pelo METRUbi com mobilidade controlada

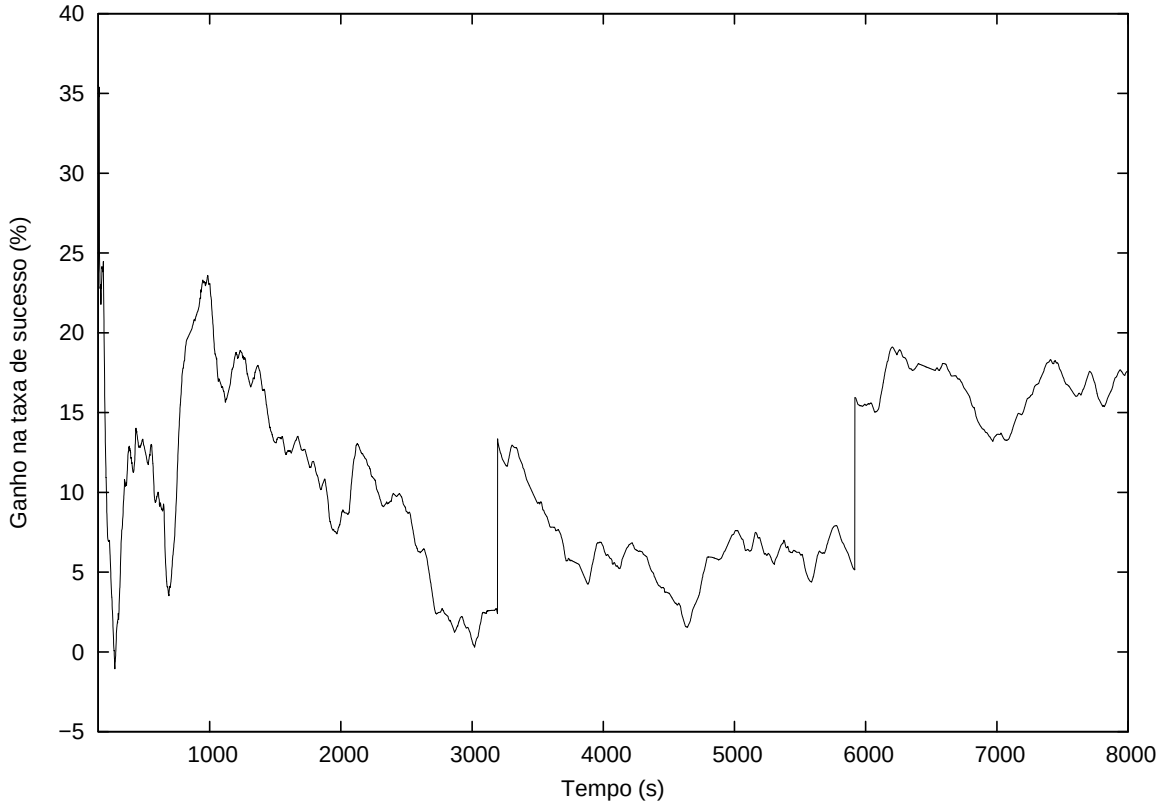

Figura 4.16: Ganho na taxa de entrega proporcionado pelo METRUbi com mobilidade controlada

Ao contrário do Experimento C, não houve perda com a aplicação do METRUbi, pois os particionamentos foram contornados, porém, em determinados instantes também não houve ganho. Vale observar que a taxa de entrega com sucesso no DSR apresenta algumas quedas abruptas, por exemplo, nos instantes de 3000 e 6000 segundos. Esse fenômeno ocorre quando há maior concentração de nós mal comportados nas proximidades, pois o algoritmo nativo do DSR terá maior probabilidade de escolhê-los para compor as rotas. Isso não ocorre com o METRUbi, pois avalia outras métricas que visam eliminar o mal comportamento.

5 CONCLUSÃO E TRABALHOS FUTUROS

Os ambientes de computação ubíqua possuem características desafiadoras, principalmente para a confiança no roteamento, devido a fatores como o dinamismo, a densidade e a heterogeneidade da rede. Essas características interferem no desempenho da comunicação, pois provocam particionamentos na rede e perdas de pacotes. Além disso, por adotarem redes auto-organizadas e autônomas, é essencial que os nós cooperem nas tarefas da rede. Contudo, a escassez de recursos dos dispositivos e a falta de controle central, deixam a rede à mercê de nós mal comportados, degradando o desempenho global da rede. Mais especificamente, podem surgir nós egoístas, que só cooperam na rede para benefício próprio, e nós em modo *sleep*, que se desligam completamente da rede para economizar energia.

Os protocolos de roteamento podem ser melhorados a fim de contornar esses problemas, adicionando maior confiança na comunicação. Em especial, o aprimoramento dos algoritmos de escolhas de rotas tem despertado o interesse de diversas pesquisas. Por exemplo, pode-se adicionar métricas que priorizem rotas mais estáticas e livres de nós mal comportados, evitando, respectivamente, quebras de enlaces e descartes de pacotes. Para isso, os protocolos de roteamento devem assimilar um maior número de variáveis do ambiente e adicionar maior raciocínio nas decisões de roteamento. Em redes dinâmicas, como as dos ambientes de computação ubíqua, as variáveis e os critérios de decisão mudam a todo instante. Logo, os protocolos devem lidar com uma grande quantidade de informações incompletas e voláteis, o que exige constante adaptação dos seus algoritmos.

Esta pesquisa propôs um mecanismo de avaliação da confiança de rotas, chamado de METRUbi, que pode ser adicionado aos protocolos de roteamento a fim de melhorar a chance de sucesso na entrega de pacotes nos ambientes de computação ubíqua. Com o METRUbi habilitado, os nós são munidos com mais raciocínio quanto à tarefa de escolha de rotas, ao contrário das regras clássicas como o SPF (*shortest-path-first*). Em primeira instância, são coletadas as informações de desempenho das rotas e os respectivos valores de métricas, propícios ao bom desempenho na comunicação. Após isso, as informações são submetidas a um sistema de aprendizagem de máquina, baseado em treinamento de árvores de decisão, com a finalidade de obter regras sobre a

avaliação das rotas. Quatro métricas são observadas no treinamento: a mobilidade, a cooperação, a atividade e a distância das rotas. As regras podem, então, ser utilizadas pelos protocolos de roteamento, aumentando a qualidade das rotas. Periodicamente, todo processo é repetido a fim de renovar as regras e garantir a adaptação do mecanismo.

Para conceber o METRUbi, inicialmente foram estudados cenários de computação ubíqua para averiguar os fenômenos que influenciavam no desempenho da rede. Nesse estudo foi demonstrado que a densidade e o dinamismo da rede, bem como os problemas de egoísmo e de modo *sleep*, interferem bastante na taxa de entrega de pacotes. Foram coletados dados de cerca de 2 mil cenários que combinaram diferentes níveis de dinamismo da rede e de presença de nós mal comportados. Com este estudo inicial, foi possível concluir que métricas que mitigassem esses fenômenos certamente beneficiariam as decisões de roteamento.

As métricas sugeridas pelo METRUbi também foram validadas por experimentos a partir da observação de amostras dos resultados de roteamento. Foi possível calcular os valores das métricas das rotas e plotá-los em gráficos de dispersão. Ao submeter tais medidas ao algoritmo de aprendizagem, foi constatado que as métricas sugeridas, principalmente a cooperação e a atividade, foram significativas para a geração de regras de avaliação das rotas. As métricas de mobilidade e distância também foram importantes para avaliar as rotas em ambientes mais hostis.

A fim de comprovar o benefício do METRUbi, foram realizados experimentos em conjunto com o protocolo DSR. O primeiro experimento analisou os resultados com o mecanismo ativado em um ambiente não cooperativo com mobilidade aleatória. Neste primeiro caso, os resultados foram computados com base na média das taxas de entrega de várias simulações. O segundo experimento fez a mesma comparação, porém em um ambiente não cooperativo com mobilidade controlada. Em ambos experimentos, o METRUbi proporcionou melhoras nas taxas globais de entrega de pacotes com sucesso na rede. Contudo, o METRUbi se mostrou limitado ao lidar com redes que apresentem particionamentos antes de um novo período de renovação de regras.

O benefício do METRUbi foi demonstrado, mas, para que seja implementado no mundo real, outros aspectos devem ser estudados e melhorados. Dessa forma, para dar continuidade à pesquisa, são sugeridos os seguintes trabalhos futuros:

- Validar as rotinas do METRUbi em outro protocolo de roteamento (e.g., AODV, OLSR, etc.) e comparar os resultados;
- Variar a estratégia de cada nó, que atualmente é constante, para verificar a adaptação do sistema em relação à reinserção de rotas com nós que deixaram de ser mal comportados;
- Implementar a métrica de posicionamento, que faria a predição das distâncias entre os nós, substituindo, assim, a métrica de mobilidade;
- Fazer experimentos em redes maiores e observar os parâmetros locais dos nós, considerando que existirá maior processamento e consumo de memória dos nós;
- Análise de outros parâmetros de desempenho além da taxa de entrega de pacotes, como a largura de banda e latência da rede;
- Nós maliciosos poderiam comprometer a integridade do mecanismo. Seria interessante fazer um estudo de quais pontos são vulneráveis e como poderiam ser melhorados;
- Verificar os parâmetros da rede nos momentos de ineficácia do METRUbi, a fim de propor técnicas para o reforço do seu algoritmo;
- Implementar um recurso que desabilita o METRUbi para pontos particionados da rede por meio da análise de mensagens de enlaces quebrados ao avaliar as rotas do *cache*;
- Fazer um estudo comparativo com os esquemas CONFIDANT, SORI, OCEAN e CORE a fim de comparar o desempenho da rede em ambientes de computação ubíqua;
- Por fim, reproduzir o mecanismo em ambientes reais a fim de verificar o impacto dos fenômenos de propagação.

Finalmente, como principal contribuição da pesquisa, foi demonstrado que o mecanismo melhorou a taxa de pacotes transmitidos com sucesso na rede nos experimentos. Seu algoritmo se adaptou com o passar do tempo, criando regras mais rígidas em ambientes hostis, a fim de evitar rotas de baixa confiança. O algoritmo de aprendizagem e as métricas sugeridas se mostraram eficazes, pois ampliaram a capacidade de decisão de roteamento dos dispositivos. Sendo assim, o emprego do METRUbi pode beneficiar o processo de roteamento, fornecendo maior desempenho aos ambientes de computação

ubíqua. Os estudos iniciais e as ferramentas desenvolvidas para a automação das simulações são as contribuições secundárias da pesquisa, podendo ser utilizadas para dar maior produtividade a outros trabalhos.

REFERÊNCIAS BIBLIOGRÁFICAS

ADNANE, A.; DE SOUSA JR, R.; BIDAN, C. Trust-based security for the OLSR routing protocol. *Computer Communications*, v. 36, p. 1159–1171, 2013. ISSN 0140-3664.

ADYA, A.; BAHL, P.; PADHYE, J. A multi-radio unification protocol for IEEE 802.11 wireless networks. *First International Conference on Broadband Networks*, p. 344–354, 2004. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1363823>.

AGARWAL, S.; AHUJA, A. Route-lifetime assessment based routing (RABR) protocol for mobile ad-hoc networks. In: *IEEE International Conference on Communications*. [s.n.], 2000. v. 3, p. 1697–1701. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=853783>.

AHUJA, A.; AGARWAL, S.; SINGH, J.; SHOREY, R. Performance of TCP over different routing protocols in mobile ad-hoc networks. *IEEE 51st Vehicular Technology Conference Proceedings*, Ieee, v. 3, p. 2315–2319, 2000. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=851686>.

ALBUQUERQUE, R. d. O. *Uma proposta de um modelo de confiança computacional para grupos em sistemas distribuídos*. Tese (Tese de Doutorado) — Universidade de Brasília, 2008.

AMVAME-NZE, G. *Reconfiguração Dinâmica de Agentes Móveis IPv4 em Redes Sem Fio Ad Hoc*. 148 p. Tese (Tese de Doutorado) — Universidade de Brasília, 2006.

APOSTOLOPOULOS, G.; GUÉRIN, R. Quality of service based routing: A performance perspective. In: *ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*. [s.n.], 1998. p. 17–28. Disponível em: <http://dl.acm.org/citation.cfm?id=285251>.

ARAUJO, R. de. Computação ubíqua: Princípios, tecnologias e desafios. In: *XXI Simpósio Brasileiro de Redes de Computadores*. Sao Carlos: Universidade de São Carlos, 2003. v. 8, p. 11–13.

ASSOCIATION, I. D. *Infrared Data Association Website*. 2012. Disponível em: <http://www.irda.org>.

AUGUSTIN, I.; FERREIRA, G. P.; YAMIN, A. Grade Computacional como Infraestrutura para a Computação Pervasiva/Ubíqua. In: *Escola Regional de Alto Desempenho*. Porto Alegre: Sociedade Brasileira da Computação, 2008. p. 77–118.

AZUMA, R. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, v. 4, n. August, p. 355–385, 1997.

BANSAL, S.; BAKER, M. Observation-based cooperation enforcement in ad hoc networks. *Cornell University Library*, 2003. Disponível em: <http://arxiv.org/abs/cs/0307012>.

BASAGNI, S.; CHLAMTAC, I.; SYROTIUK, V.; WOODWARD, B. A distance routing effect algorithm for mobility (DREAM). In: *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. New York, USA: ACM, 1998. p. 76–84. Disponível em: <http://portal.acm.org/citation.cfm?id=288254>.

BASAGNI, S.; CONTI, M.; GIORDANO, S.; STOJMENOVIC, I. Mobile ad hoc networking. In: IEEE (Ed.). *Mobile Ad Hoc Networking*. 1st. ed. New Jersey,USA: Wiley-IEEE Press, 2004. p. 1–63.

BAUMANN, R.; HEIMLICHER, S.; STRASSER, M.; WEIBEL, A. *A Survey on Routing Metrics - TIK Report 262*. ETH-Zentrum, Switzerland, 2007.

BEIJAR, N. Zone routing protocol (ZRP). *Networking Laboratory Helsinki University of Technology Finland*, Citeseer, v. 9, n. 4, p. 427–438, 2002. Disponível em: <http://gicl.cs.drexel.edu/people/regli/Classes/CS680/Papers/Ad Hoc/Routing/zrp.pdf>.

BHARGHAVAN, V.; DEMERS, A. MACAW: a media access protocol for wireless LAN's. In: ACM (Ed.). *Proceedings of the conference on Communications architectures, protocols and applications - SIGCOMM 94*. New York, USA: [s.n.], 1994. p. 212–225. Disponível em: <http://dl.acm.org/citation.cfm?id=190314.190334>.

BLAZE, M.; FEIGENBAUM, J.; IONNIDIS, J.; KEROMYTIS, A. D. The role of trust management in distributed systems security. In: *Secure internet programming: security issues for mobile and distributed objects*. Berlin: Lecture Notes in Computer Science, 1996. p. 185–210.

BOHN, J. *Reliability: from Distributed Systems to Ubicomp*. Zurich, 2002. 1–17 p. Disponível em: <http://www.vs.inf.ethz.ch/edu/WS0102/UI/slides/ui_08reliable.pdf>.

BOUKERCHE, A.; REN, Y. A trust-based security system for ubiquitous and pervasive computing environments. *Computer Communications*, Elsevier B.V., v. 31, n. 18, p. 4343–4351, dez. 2008. ISSN 01403664. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0140366408003009 http://www.sciencedirect.com/science/article/pii/S0140366408003009>.

BRANDAO, J. E. M. d. S.; FRAGA, J. d. S. Abordagens e Emprego de Agentes Móveis em Detecção de Intrusão. In: *VI Simposio Segurança em Informática (SSI'2004)*. São José dos Campos: [s.n.], 2004.

BROOKS, K. The context quintet: narrative elements applied to context awareness. In: MOTOROLA HUMAN INTERFACE LABS. *Human Computer Interaction International Proceedings*. Crete,Greece: Erlbaum Associates Inc, 2003. v. 2003. Disponível em: <http://xenia.media.mit.edu/ brooks/storybiz/Brooks-context_quintet.pdf>.

BUCHEGGER, S.; Le Boudec, J. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks). In: *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*. Citeseer, 2002. p. 226–236. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.9062>.

BUCHEGGER, S.; Le Boudec, J. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In: *Proceedings of WiOpt'03 Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*. [S.l.]: Citeseer, 2003. v. 4, n. 2, p. 4–1.

CAPKUN, S.; HAMDI, M.; HUBAUX, J. GPS-free positioning in mobile ad hoc networks. *Cluster Computing*, Springer, v. 5, n. 2, p. 157–167, 2002. Disponível em: <http://www.springerlink.com/index/XP3J7RA35HYFV474.pdf>.

CHIANG, C.; WU, H.; LIU, W.; GERLA, M. Routing in clustered multihop, mobile wireless networks with fading channel. In: *Proceedings of IEEE SICON*. Los Angeles, USA: [s.n.], 1997. v. 97, p. 197–211. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.50.8359>.

CHO, J.; SWAMI, A.; CHEN, R. A survey on trust management for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, v. 13, n. 4, p. 562–583, 2011. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5604602>.

CORDASCO, J.; WETZEL, S. Cryptographic Versus Trust-based Methods for MANET Routing Security. *Electronic Notes in Theoretical Computer Science*, v. 197, n. 2, p. 131–140, fev. 2008. ISSN 15710661. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1571066108000613>.

CORSON, S. RFC 2501: Mobile Ad hoc Networking (MANET). In: *IETF working group*. Internet Engineering Task Force, 1999. Disponível em: <http://www.faqs.org/rfcs/rfc2501.html>.

COUTO, D.; AGUAYO, D.; BICKET, J.; MORRIS, R. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, v. 11, n. 4, p. 419–434, 2005. Disponível em: <http://www.springerlink.com/index/X1L047934QU8H56U.pdf>.

DE SOUSA JR, R. T.; ADNANE, A.; BIDAN, C.; ME, L. On the Vulnerabilities and Protections of the OLSR ad hoc Routing Protocol from the point of view of Trust. *IEEE Latin America Transactions*, v. 7, n. 5, p. 594–602, set. 2009. ISSN 1548-0992. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5361198>.

DEUS, F. E. G. de. *Mecanismo de Otimização para Sobrevivência em WLAN: Estudo de Caso em Rede IEEE 802.11*. 140 p. Tese (Tese de Doutorado) — Universidade de Brasília, 2006.

DEY, A. Understanding and using context. *Personal and ubiquitous computing*, v. 5, n. 1, p. 4–7, 2001. Disponível em: <http://www.springerlink.com/index/1d9grxkjvquhpwkw.pdf>.

DJENOURI, D.; BADACHE, N. Cross-layer approach to detect data packet droppers in mobile ad-hoc networks. *Self-Organizing Systems*, Springer, v. 4124, p. 163–176, 2006. Disponível em: <http://www.springerlink.com/index/F91684M082376711.pdf>.

DRAVES, R.; PADHYE, J.; ZILL, B. Comparison of routing metrics for static multi-hop wireless networks. In: *SIGCOMM 04 Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, USA: [s.n.], 2004. p. 133–144. Disponível em: <http://dl.acm.org/citation.cfm?id=1015467.1015483>.

DUBE, R.; RAIS, C. Signal stability-based adaptive routing (SSA) for ad hoc mobile networks. *Personal Communications, IEEE*, v. 4, n. 1, p. 36–45, 1997. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=575990>.

DUBE, R.; RAIS, C.; WANG, K.; TRIPATHI, S. Signal stability-based adaptive routing (SSA) for ad hoc mobile networks. *IEEE Personal Communications*, IEEE, v. 4, n. 1, p. 36–45, 1997. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=575990>.

FARINACCI, D.; LIU, C.; DEERING, S.; ESTRIN, D. *RFC 2362: Protocol independent multicast-sparse mode (PIM-SM): Protocol specification*. Internet Engineering Task Force, 1998. 1–66 p. Disponível em: <http://www.cise.ufl.edu/ helmy/papers/PIM-SM-Spec-97protocol.pdf http://tools.ietf.org/html/rfc2362.txt>.

FEENEY, L.; NILSSON, M. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications*. Ieee, 2001. p. 1548–1557. ISBN 0-7803-7016-3. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=916651>.

FRIDAY, A.; DAVIES, N.; WALLBANK, N.; CATTERALL, E.; PINK, S. Supporting service discovery, querying and interaction in ubiquitous computing environments. *Wireless Networks*, Springer, v. 10, n. 6, p. 631–641, 2004. Disponível em: <http://www.springerlink.com/index/L0836837J6570831.pdf>.

GAMBETTA, D. Trust: Making and breaking cooperative relations. In: *Trust as a Commodity*. New York, USA: Blackwell, 1990. p. 47–72. Disponível em: <http://www.citeulike.org/group/756/article/522989>.

GAMBETTA, D. Can we trust trust. In: *Trust: Making and breaking cooperative relations*. [s.n.], 2000. p. 213–237. Disponível em: <http://reference.kfupm.edu.sa/content/c/w/can_we_trust_trust_110761.pdf>.

GEER, D. Nanotechnology: the growing impact of shrinking computers. *Pervasive Computing, IEEE*, IEEE, v. 5, n. 1, p. 7–11, 2006. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1593564>.

GERLA, M.; CHEN, L.; LEE, Y.; ZHOU, B. Dealing with node mobility in ad hoc wireless network. *Formal Methods for Mobile Computing*, v. 3465, p. 69–106, 2005. Disponível em: <http://www.springerlink.com/index/V6FGJVWW6KNL91QH.pdf>.

GERLA, M.; PEI, G.; LEE, S.; CHIANG, C. On-Demand Multicast Routing Protocol (ODMRP) for mobile Ad-Hoc Networks. In: *WAM Laboratory*. [s.n.], 1998. Disponível em: <http://www.ietf.org/proceedings/98dec/slides/manet-lee-98dec.pdf>.

GIORDANO, S. Mobile ad hoc networks. In: *Handbook of wireless networks and mobile computing*. Wiley Online Library, 2002. p. 15. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/0471224561.ch15/summary>.

GLOMOSIM. *About GloMoSim*. 1999. Disponível em: <http://pcl.cs.ucla.edu/projects/glomosim/>.

GOFF, T.; ABU-GHAZALEH, N. Preemptive routing in ad hoc networks. *Journal of Parallel and Distributed Computing*, v. 63, n. 2, p. 123–140, 2003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S074373150200059X>.

GROUP, I. M. W. *MANET Charter, 2000*. [S.l.], 2000. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Manet+Charter#3>.

GUERRA, G. N. *Universidade de Brasília Modelo de Reputação e Ontologia Aplicados à Rede Social Científica do ObserveUnB*. Tese (Dissertação de Mestrado) — Universidade de Brasília, 2012.

HARRISON, B.; FISHKIN, K.; GUJAR, A. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In: *CHI '98 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. [s.n.], 1998. p. 17–24. Disponível em: <http://dl.acm.org/citation.cfm?id=274647>.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Infrastructure for pervasive computing: Challenges. 2001. Disponível em: <http://www.gta.ufrj.br/seminarios/CPE826/doc/ip.pdf>.

IANA. *Assigned Internet Protocol Numbers*. Internet Assign Numbers Authority, 2012. Disponível em: <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml>.

IEEE WPAN TG1. *IEEE 802.15.1 WPAN TG1*. 2002. Disponível em: <http://www.ieee802.org/15/pub/TG1.html>.

IWATA, A.; CHIANG, C.; PEI, G. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 17, n. 8, p. 1369–1379, 1999. ISSN 0733-8716. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=779920>.

JACQUET, P. RFC-3626: Optimized link state routing protocol (OLSR). In: RFC 3626, OCTOBER. *IETF working group*. Internet Engineering Task Force, 2003. p. 1–75. Disponível em: <http://en.scientificcommons.org/42911053 http://tools.ietf.org/html/rfc3626.txt>.

JAQUET, P.; MüHLETHALER, P.; CLAUSEN, T. Optimized link state routing protocol for ad hoc networks. In: *IEEE INMIC 2001 Multi Topic Con-*

ference. Technology for the 21st Century. [s.n.], 2001. p. 62–68. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=995315>.

JAYASURIYA, A.; PERREAU, S.; DADEJ, A.; GORDON, S. Hidden vs. exposed terminal problem in ad hoc networks. In: *Proceedings of the Australian Telecommunication Networks and Applications Conference*. Sidney,Australia: [s.n.], 2004. Disponível em: <http://sandilands.info/sgordon/doc/jayasuriya2004-hidden.pdf>.

JOA-NG, M. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, v. 17, n. 8, p. 1415–1425, 1999. ISSN 07338716. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=779923>.

JOHNSON, D.; HU, Y.; MALTZ, D. RFC 4728: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. In: *IETF working group*. Internet Engineering Task Force, 2001. Disponível em: <http://www.ietf.org/rfc/rfc4728.txt>.

JOHNSON, D.; MALTZ, D.; BROCH, J. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad hoc networking*, Citeseer, v. 5, p. 139–172, 2001. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.5263>.

JOSHI, A.; FININ, T.; KAGAL, L.; PARKER, J.; PATWARDHAN, A. Security policies and trust in ubiquitous computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, v. 366, n. 1881, p. 3769–80, out. 2008. ISSN 1364-503X. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/18672450>.

JUBIN, J.; TORNOW, J. The DARPA packet radio network protocols. *Proceedings of the IEEE*, IEEE, v. 75, n. 1, p. 21–32, 1987. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1457970>.

KARN, P. MACA- a new channel access method for packet radio. In: *Computer Networking Conference*. [s.n.], 1990. p. 134–140. Disponível em: <http://radio-network-simulation.googlecode.com/svn/trunk/papers/P.Karn MACA A new Channel.pdf>.

KHANNA, A.; ZINKY, J. The revised ARPANET routing metric. *SIGCOMM '89 Symposium proceedings on Communications architectures & protocols*, v. 19, n. 4, p. 45–56, 1989. Disponível em: <http://dl.acm.org/citation.cfm?id=75252>.

KHOSLA, P. SORI: a secure and objective reputation-based incentive scheme for ad-hoc networks. *2004 IEEE Wireless Communications and*

Networking Conference, Ieee, v. 2, p. 825–830, 2004. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1311293>.

KHOSRAVI, E.; TYSON, B. Increasing security in mobile ad hoc networks by incentives to cooperate and secure routing. In: *Security and Management*. Los Angeles, USA: Southern University, 2006. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.4851>.

KIRUTHIKA, R.; UMARANI, R. An Exploration of Count-to-Infinity Problem in Networks. *International Journal of Engineering Science and Technology*, v. 2, n. 12, p. 7155–7159, 2010. Disponível em: <http://www.ijest.info/docs/IJEST10-02-12-065.pdf http://journals.indexcopernicus.com/abstracted.php?icid=940680>.

KNOBLAUCH, R.; PIETRUCHA, M.; NITZBURG, M. Field studies of pedestrian walking speed and start-up time. *Transportation Research Record: Journal of the Transportation Research Board*, Trans Res Board, v. 1538, p. 27–38, 1996. Disponível em: <http://trb.metapress.com/index/K340133161573026.pdf>.

KO, Y.; VAIDYA, N. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, Kluwer Academic Publishers, v. 6, n. 4, p. 307–321, 2000. Disponível em: <http://portal.acm.org/citation.cfm?id=352164>.

KOGURE, K.; HAGITA, N.; SUMI, Y.; KUWAHARA, N.; ISHIGURO, H. Toward ubiquitous intelligent robotics. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. IEEE, 2003. v. 2, p. 1826–1831. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1248909>.

KRUKOW, K.; NIELSEN, M.; SASSONE, V. Trust models in ubiquitous computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, v. 366, n. 1881, p. 3781–93, out. 2008. ISSN 1364-503X. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/18678555>.

LEE, S.-J.; BELDING-ROYER, E. M.; PERKINS, C. E. Ad hoc on-demand distance-vector routing scalability. *ACM SIGMOBILE Mobile Computing and Communications Review*, v. 6, n. 3, p. 94, jun. 2002. ISSN 15591662. Disponível em: <http://portal.acm.org/citation.cfm?doid=581291.581306>.

LIU, C.; KAISER, J. A survey of mobile ad hoc network routing protocols. *Analysis*, Citeseer, Magdeburg, v. 1, n. 2003, p. 125–136, 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.7768 http://www.minema.di.fc.ul.pt/reports/report_routing-protocols-survey-final.pdf>.

LUNDBERG, J. Routing security in ad hoc networks. *Helsinki University of Technology*, p. 1–12, 2000. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.144.7589>.

MACKER, J.; PARK, V.; CORSON, M. Mobile and wireless internet services: Putting the pieces together. *IEEE Communications Magazine*, v. 39, n. 6, p. 148–155, 2001. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=925683>.

MALTZ, D.; BROCH, J.; JOHNSON, D. *Experiences designing and building a multihop wireless ad hoc network testbed*. [S.l.], 1999. 99–116 p.

MAN, L. A. N.; SOCIETY, C. IEEE Std 802.11 - 2007, IEEE Standard for Information Technology - Telecommunications and information exchange between systems-LANs and MANs. In: *Control*. New York, USA: IEEE Computer Society, 2007. cap. Part 11.

MARIAS, G. F.; GEORGIADIS, P.; FLITZANIS, D.; MANDALAS, K. Cooperation enforcement schemes for MANETs: a survey. *Wireless Communications and Mobile Computing*, v. 6, n. 3, p. 319–332, maio 2006. ISSN 1530-8669. Disponível em: <http://doi.wiley.com/10.1002/wcm.398>.

MARTI, S.; GIULI, T.; LAI, K.; BAKER, M. Mitigating routing misbehavior in mobile ad hoc networks. In: *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000. p. 255–265. Disponível em: <http://portal.acm.org/citation.cfm?id=345955>.

MITCHELL, T. M. *Machine Learning*. McGraw-Hill, 1997. Disponível em: <http://www.springerlink.com/index/H537516835XW154P.pdf>.

MOLVA, R.; MICHIARDI, P. *Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks*. Sophia Antipolis, France, 2001. 107 p. Disponível em: <http://www.eurecom.edu/ nsteam/Papers/michiardi_adhoc_core.pdf>.

MOY, J. *RFC 1584: Multicast extensions to OSPF*. [S.l.]: Internet Engineering Task Force, 1994.

MUI, L.; MOHTASHEMI, M. Ratings in distributed systems: A bayesian approach. *Workshop on Information Technologies and Systems*, 2001. Disponível em: <http://bitsavers.trailing-edge.com/pdf/mit/lcs/tm/MIT-LCS-TM-617.pdf>.

MUI, L.; MOHTASHEMI, M.; HALBERSTADT, A. A computational model of trust and reputation for e-business. In: *35th Annual Hawaii International Conference on System Sciences*. Wahshington, DC.: IEEE Computer Society, 2002. v. 7, p. 188.

MURTHY, S.; GARCIA-LUNA-ACEVES, J. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, Springer, v. 1, n. 2, p. 183–197, 1996. ISSN 1383-469X. Disponível em: <http://www.springerlink.com/index/X0811063175870L4.pdf>.

NIEMELA, E.; LATVAKOSKI, J. Survey of requirements and solutions for ubiquitous software. In: *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*. ACM, 2004. p. 71–78. Disponível em: <http://dl.acm.org/citation.cfm?id=1052391>.

NIKAEIN, N.; BONNET, C. Harp-hybrid ad hoc routing protocol. *Proceedings of International Symposium on Telecommunications*, v. 1, n. 3, 2001. Disponível em: <http://www.cs.cornell.edu/courses\cs615\2002fa/615/harp.pdf>.

NUEVO, J. A comprehensible glomosim tutorial. *INRS- Universite du Quebec.*, p. 1–34, 2004.

PARISSIDIS, G.; KARALIOPOULOS, M. Routing metrics for wireless mesh networks. In: *Computer Communications and Network - Guide to Wireless Mesh Networks*. [s.n.], 2009. p. 199–230. Disponível em: <http://www.springerlink.com/index/g65k17442618m12r.pdf>.

PAUL, K. Communication-aware mobile hosts in ad-hoc wireless network. In: *IEEE International Conference on Personal Wireless Communication,*. [s.n.], 1999. p. 83–87. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=759590>.

PERKINS, C. E.; BHAGWAT, P. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *ACM SIGCOMM Computer Communication Review*, v. 24, n. 4, p. 234–244, out. 1994. ISSN 01464833. Disponível em: <http://portal.acm.org/citation.cfm?doid=190809.190336>.

PIRZADA, A. A.; MCDONALD, C.; DATTA, A. Performance comparison of trust-based reactive routing protocols. *IEEE Transactions on Mobile Computing*, v. 5, n. 6, p. 695–710, jun. 2006. ISSN 1536-1233. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1624341>.

PUNNOOSE, R.; NIKITIN, P. Optimizing wireless network protocols using real-time predictive propagation modeling. In: *RAWCON 99. 1999 IEEE Radio and Wireless Conference, 1999.* [s.n.], 1999. p. 39–44. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=810925>.

PUTTINI, R. S. *Um Modelo de Segurança para Redes Móveis Ad Hoc*. Tese (Tese de Doutorado) — Universidade de Brasília, 2004.

QUINLAN, J. *C4. 5: programs for machine learning*. San Francisco,USA: Morgan Kaufmann, 1993.

ROYER, E. Multicast ad hoc on-demand distance vector (MAODV) routing. *IETF, Intemet Draft*, 2000.

ROYER, E.; TOH, C. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, IEEE, v. 6, n. 2, p. 46–55, 1999. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=760423>.

RUSSELL, S.; NORVIG, P. *Artificial intelligence: a modern approach*. New Jersey: Prentice-Hall Inc, 1995.

SABATER, J.; SIERRA, C. Reputation and social network analysis in multi-agent systems. In: *AAMAS '02 Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. [s.n.], 2002. p. 475–482. Disponível em: <http://dl.acm.org/citation.cfm?id=544854>.

SAHA, D.; MUKHERJEE, A. Pervasive computing: a paradigm for the 21st century. *Computer*, v. 36, n. 3, p. 25–31, 2003. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1185214>.

SAKAMURA, K. Challenges in the age of ubiquitous computing: a case study of T-Engine, an open development platform for embedded systems. In: *Proceedings of the 28th international conference on Software engineering*. ACM, 2006. p. 713–720. Disponível em: <http://dl.acm.org/citation.cfm?id=1134399>.

SATYANARAYANAN, M. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, IEEE, v. 8, n. 4, p. 10–17, 2001. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=943998>.

SCHILIT, B.; ADAMS, N. Context-aware computing applications. In: *WMCSA 1994. First Workshop on Mobile Computing Systems and Applications.* [s.n.], 1994. p. 85–90. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4624429>.

SEREDYNSKI, M.; BOUVRY, P.; KLOPOTEK, M. A. Evolving Strategy-based Cooperation in Wireless Mobile Ad Hoc Networks. In: *Review Literature And Arts Of The Americas*. [S.l.]: World Scientific Review, 2008.

SERIQUE JR, L. F. S.; DE SOUSA JR, R. T.; VASCONCELOS, V. d. A. Evolução da Cooperação em Redes Ad Hoc usando Algorítmos Genéticos na Seleção de Rotas Aptas. In: *International Information and Telecommunication Technologies Symposium*. [S.l.: s.n.], 2009.

SERIQUE, L. F.; SOUSA, R. T. de. Evaluating Trust in Ad Hoc Network Routing by Induction of Decision Trees. *Revista IEEE America Latina*, v. 10, n. 1, 2012.

SERIQUE, L. F. S.; SOUSA, R. T. de. Um Framework para o Desenvolvimento de Sistemas de Detecção de Intrusão baseado em Múltiplos Agentes. In: *International Information and Telecommunication Technologies Symposium*. [S.l.: s.n.], 2005. p. 1–2.

SERIQUE, L. F. S.; SOUSA, R. T. de. Avaliando a Confiança do Roteamento em Redes Ad Hoc por Indução de Árvores de Decisão. In: *10th International Information and Telecommunication Technologies Conference*. [S.l.: s.n.], 2011. p. 218–229.

SHI, H. *Communication and Computing for Distributed Multimedia Systems*. MA, USA: Artech House Inc., 2001. Disponível em: <http://dl.acm.org/citation.cfm?id=248734 https://scpe.org/index.php/scpe/article/view/84>.

SIVAKUMAR, R.; SINHA, P.; BHARGHAVAN, V. CEDAR: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 17, n. 8, p. 1454–1465, 1999. ISSN 0733-8716. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=779926>.

STAJANO, F.; ANDERSON, R. The resurrecting duckling: Security issues for ad-hoc wireless networks. *Security Protocols*, Springer, v. 1796, p. 172–182, 2000. Disponível em: <http://www.springerlink.com/index/ru2015q381304428.pdf>.

STALLINGS, W. *Cryptography and network security, principles and practices*. Prentice-Hall Inc, 2007. Disponível em: <http://dl.acm.org/citation.cfm?id=1209579>.

SUN, J. Mobile ad hoc networking: an essential technology for pervasive computing. In: *Proc. International Conferences on Info-tech and Info-net -ICII 2001*. [s.n.], 2001. v. 3, p. 316–321. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=983076>.

SUNDARARAJAN, T.; SHANMUGAM, A. Performance Analysis of Selfish Node Aware Routing Protocol for Mobile Ad Hoc Networks. In: *The International Con-*

gress for global Science and Technology. [s.n.], 2009. v. 9, n. 1, p. 1. Disponível em: <http://www.icgst.com/cnir/Volume9/Issue1/CNIR-V9-I1.pdf#page=5>.

THAKARE, A. N. Performance Analysis of AODV & DSR Routing Protocol in Mobile Ad hoc Networks. *IJCA Special Issue on MANETs*, v. 4, p. 211–218, 2010.

THEODORAKOPOULOS, G.; BARAS, J. Trust evaluation in ad-hoc networks. In: *Proceedings of the 3rd ACM workshop on Wireless security*. ACM, 2004. p. 1–10. Disponível em: <http://portal.acm.org/citation.cfm?id=1023648>.

TOH, C. Associativity-based routing for ad hoc mobile networks. *Wireless Personal Communications*, Springer, v. 4, n. 2, p. 103–139, 1997. Disponível em: <http://www.springerlink.com/index/X0516P1X801H7494.pdf>.

TOH, C. Associativity-based routing for ad hoc mobile networks. *Wireless Personal Communications*, v. 4, n. 2, p. 103–139, 1997. Disponível em: <http://www.springerlink.com/index/X0516P1X801H7494.pdf>.

UCLA. *UCLA Parsec Programming Language*. 1999. Disponível em: <http://pcl.cs.ucla.edu/projects/parsec/>.

WAITZMAN, D.; PARTRIDGE, C. *RFC 1075: Distance Vector Multicast Routing Protocol*. [S.l.]: Internet Engineering Task Force, 1988.

WANG, F.; WANG, F.; HUANG, B.; YANG, L. T. COSR: A Reputation-Based Secure Route Protocol in MANET. *EURASIP Journal on Wireless Communications and Networking - Special issue on multimedia communications over next generation wireless networks*, v. 2010, p. 1–10, 2010. ISSN 1687-1472. Disponível em: <http://www.hindawi.com/journals/wcn/2010/258935/>.

WANG, Z.; CROWCROFT, J. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, v. 14, n. 7, p. 1228–1234, 1996. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=536364>.

WANT, R.; PERING, T. System challenges for ubiquitous & pervasive computing. *Proceedings of the 27th international conference on Software engineering - ICSE '05*, ACM Press, New York, New York, USA, p. 9–14, 2005. Disponível em: <http://portal.acm.org/citation.cfm?doid=1062455.1062463>.

WEISER, M. The world is not a desktop. *Interactions*, ACM, v. 1, n. 1, p. 7–8, 1994. Disponível em: <http://doi.ieeecomputersociety.org/10.1145/174800.174801>.

WEISER, M. *Ubiquitous Computing*. 1996. Disponível em: <http://sandbox.xerox.com/ubicomp/>.

WEISER, M. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review - Special issue dedicated to Mark Weiser*, p. 3–11, 1999. Disponível em: <http://wiki.daimi.au.dk/pca/_files/weiser-orig.pdf>.

WEISER, M.; BROWN, J. The coming age of calm technology. In: *Beyond calculation*. New York, USA: Copernicus New York, 1997. p. 75–85. Disponível em: <http://www.cs.ucsb.edu/ ebelding/courses/284/papers/calm.pdf http://homes.di.unimi.it/ boccignone/GiuseppeBoccignone_webpage/IUM2_files/weiser-calm.pdf>.

WU, C.; TAY, Y. AMRIS: A multicast protocol for ad hoc wireless networks. In: *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*. IEEE, 1999. v. 1, n. c, p. 25–29. ISBN 0780355385. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=822636>.

XIE, J.; TALPADE, R.; MCAULEY, A.; LIU, M. AMRoute: ad hoc multicast routing protocol. *Mobile Networks and Applications*, Kluwer Academic Publishers, v. 7, n. 6, p. 429–439, 2002. ISSN 1383-469X. Disponível em: <http://portal.acm.org/citation.cfm?id=603299>.

YASSER, K.; HESHAM, A.; ELMAHDI, N.; ALLOLA, S.; AHMAD, H. Optimizing Mobile Agents Migration Based on Decision Tree Learning. In: *Proceedings of the World Academy of Science Engineering and Technology, PWASET*. [S.l.: s.n.], 2007. p. 564–570.

YAU, S. S.; KARIM, F. An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments. *Real-Time Systems*, v. 26, n. 1, p. 29–61, jan. 2004. ISSN 0922-6443.

ZHAO, J.; GOVINDAN, R. Understanding packet delivery performance in dense wireless sensor networks. In: *SenSys '03 Proceedings of the 1st international conference on Embedded networked sensor systems*. [s.n.], 2003. p. 1–13. Disponível em: <http://dl.acm.org/citation.cfm?id=958491.958493>.

APÊNDICES

A Script para Automação das Simulações

```
1  ##################################################################################################################
2  # Universidade de Brasília - UnB  * Departamento de Engenharia Elétrica - ENE
3  # Programa de Pós-graduação em Engenharia Elétrica - PPGEE
4  # Script para automação de simulações em Glomosim
5  # Autores:
6  #  Luiz Fernando S. Serique Junior <serique@gmail.com>
7  #  Rafael Timóteo de Sousa Junior <desousa@unb.br>
8  # Atualizado em: 13/10/2011
9  ##################################################################################################################
10 #/bin/bash
11
12 CONFIG="config.in"
13 PROTOCOL="DSR"
14 OUTPUT="glomo_metrubi_dsr.dat"
15
16 sed -i "s/^ROUTING-PROTOCOL.*/ROUTING-PROTOCOL $PROTOCOL/g" $CONFIG
17 rm $OUTPUT
18
19 for DENSITY in 50 100 150 200 250 300
20 do
21   sed -i "s/NUMBER-OF-NODES .*/NUMBER-OF-NODES $DENSITY/g" $CONFIG
22
23   for MOBILITY in 0 1 2 3 4 5 6 7 8 9 10
24   do
25     if [ $MOBILITY = 0 ]; then
26
27       sed -i "s/^#*MOBILITY NONE/MOBILITY NONE/g" $CONFIG
28       sed -i "s/#*MOBILITY RANDOM-WAYPOINT/#MOBILITY RANDOM-WAYPOINT/g" $CONFIG
29       sed -i "s/#*MOBILITY-WP-PAUSE.*/#MOBILITY-WP-PAUSE/g" $CONFIG
30       sed -i "s/#*MOBILITY-WP-MIN-SPEED.*/#MOBILITY-WP-MIN-SPEED/g" $CONFIG
31       sed -i "s/#*MOBILITY-WP-MAX-SPEED.*/#MOBILITY-WP-MAX-SPEED/g" $CONFIG
32     else
33       sed -i "s/#*MOBILITY NONE/#MOBILITY NONE/g" $CONFIG
34       sed -i "s/#*MOBILITY RANDOM-WAYPOINT/MOBILITY RANDOM-WAYPOINT/g" $CONFIG
35       sed -i "s/#*MOBILITY-WP-PAUSE.*/MOBILITY-WP-PAUSE 20/g" $CONFIG
36       sed -i "s/#*MOBILITY-WP-MIN-SPEED.*/MOBILITY-WP-MIN-SPEED $MOBILITY/g" $CONF
37       sed -i "s/#*MOBILITY-WP-MAX-SPEED.*/MOBILITY-WP-MAX-SPEED $MOBILITY/g" $CONFIG
38     fi
39
40     for SELFISH in 0 10 20 30 40 50
41     do
42       sed -i "s/NUMBER-OF-SELFISH-NODES .*/NUMBER-OF-SELFISH-NODES $SELFISH/g" $CONFIG
43
44       for SLEEP in 0 10 20 30 40 50
45       do
46
47         sed -i "s/NUMBER-OF-SLEEPY-NODES .*/NUMBER-OF-SLEEPY-NODES $SLEEP/g" $CONFIG
48
49         let C=0
50         let PKT_TM=0
51         let DLY_TM=0
52         let BRK_TM=0
53         let DRP_TM=0
54         let REQ_TM=0
55         let REP_TM=0
56         let ERR_TM=0
57
58         for SEED in 3 9 17 22 31 44 58 65 77 88
59         do
60           let C=$C+1
61           sed -i "s/SEED .*/SEED $SEED/g" $CONFIG
62
63           echo "Simulando PROTOCOL=$PROTOCOL, DENSITY=$DENSITY, MOBILITY=$MOBILITY, SELFISH=$SELFISH, SLEEP=$SLEEP SEED=$SEED .."
64
65           ./glomosim $CONFIG > /dev/null
```

```
66
67          let PKT_S=‘cat glomo.stat  | grep ’packets sent’ | cut -f4 -d’:’ | awk ’{s+=$1} END {print s}’‘
68          let PKT_R=‘cat glomo.stat  | grep ’packets received’ | cut -f4 -d’:’ | awk ’{s+=$1} END {print s}’‘
69
70          BRK=‘cat glomo.stat  | grep ’Link Breaks’ | cut -f2 -d’=’ | awk ’{s+=$1} END {print s}’‘
71          DLY=‘cat glomo.stat  | grep ’end-to-end’ | cut -f4 -d’:’ | awk ’{s+=$1} END {print s}’‘
72          DRP=‘cat glomo.stat  | grep ’Dropped Packets’ | cut -f2 -d’=’ | awk ’{s+=$1} END {print s}’‘
73          REQ=‘cat glomo.stat  | grep ’Requests Txed’ | cut -f2 -d’=’ | awk ’{s+=$1} END {print s}’‘
74          REP=‘cat glomo.stat  | grep ’Replies Txed’ | cut -f2 -d’=’ | awk ’{s+=$1} END {print s}’‘
75          ERR=‘cat glomo.stat  | grep ’Errors Txed’ | cut -f2 -d’=’ | awk ’{s+=$1} END {print s}’‘
76
77          PKT_T=‘echo "scale=2; ($PKT_R/$PKT_S)*100" | bc -l‘
78          PKT_TM=‘echo "scale=2; ($PKT_TM+$PKT_T)" | bc -l‘
79          DLY_TM=‘echo "scale=2; ($DLY_TM+$DLY)" | bc -l‘
80          BRK_TM=‘echo "scale=2; ($BRK_TM+$BRK)" | bc -l‘
81          DRP_TM=‘echo "scale=2; ($DRP_TM+$DRP)" | bc -l‘
82          REQ_TM=‘echo "scale=2; ($REQ_TM+$REQ)" | bc -l‘
83          REP_TM=‘echo "scale=2; ($REP_TM+$REP)" | bc -l‘
84          ERR_TM=‘echo "scale=2; ($ERR_TM+$ERR)" | bc -l‘
85          echo "    Taxa de pacotes com sucesso: $PKT_T"
86          echo "    Atraso na transmissão: $DLY"
87          echo "    Quebras de link: $BRK"
88          echo "    Pacotes descartados: $DRP"
89          echo "    Requests enviados: $REQ"
90          echo "    Replies enviados: $REP"
91          echo "    Errors enviados: $ERR"
92        done
93
94        PKT_TM=‘echo "scale=2; ($PKT_TM/$C)" | bc -l‘
95        DLY_TM=‘echo "scale=2; ($DLY_TM/$C)" | bc -l‘
96        BRK_TM=‘echo "scale=2; ($BRK_TM/$C)" | bc -l‘
97        DRP_TM=‘echo "scale=2; ($DRP_TM/$C)" | bc -l‘
98        REQ_TM=‘echo "scale=2; ($REQ_TM/$C)" | bc -l‘
99        REP_TM=‘echo "scale=2; ($REP_TM/$C)" | bc -l‘
100       ERR_TM=‘echo "scale=2; ($ERR_TM/$C)" | bc -l‘
101
102       echo "    ========================================     "
103       echo "    Taxa média de pacotes com sucesso: $PKT_TM"
104       echo "    Atraso médio de transmissão: $DLY_TM"
105       echo "    Quebra média de links: $BRK_TM"
106       echo "    Média de pacotes descartados: $DRP_TM"
107       echo "    Média de requests enviados: $REQ_TM"
108       echo "    Média de replies enviados: $REP_TM"
109       echo "    Média de errors enviados: $ERR_TM"
110       echo "$DENSITY $MOBILITY $SELFISH $SLEEP $PKT_TM $DLY_TM $BRK_TM $DRP_TM $REQ_TM $REP_TM $ERR_TM" >> $OUTPUT
111
112     done
113   done
114 done
115 done
116
```

B Rotina de Cálculo das Métricas de Confiança

```
1  ##################################################################################################################
2  # Universidade de Brasília - UnB  * Departamento de Engenharia Elétrica - ENE
3  # Programa de Pós-graduação em Engenharia Elétrica - PPGEE
4  # Rotina MetrubiGetRouteTrust()
5  # Autores:
6  #  Luiz Fernando S. Serique Junior <serique@gmail.com>
7  #  Rafael Timóteo de Sousa Junior <desousa@unb.br>
8  # Atualizado em: 13/10/2011
9  ##################################################################################################################
10
11 int MetrubiGetRouteTrust( NODE_ADDR *path, int numHops, double **trustRegister, int numNodes ){
12
13   int nodeAddr, i;
14   int iPathAct = 0;
15   int iPathRep = 1;
16   int iPathMob = 0;
17   int iPathDst = numHops;
18   int routeTrust = 0;
19   double avgAct = 0;
20   double nodeAct = 0;
21   double pathAct = 0;
22   double nodeRep = 1;
23   double pathRep = 1;
24   double nodeSpeed = 0;
25   double pathMob = 0;
26   double lowBoundAct = 0;
27   double highBoundAct = 0;
28   double maxNodeAct = 0;
29   double minNodeAct = 0;
30   double tmpAct = 1;
31
32   //find minNodeAct, maxNodeAct and avgAct in trust register
33   for (i = 0; i < numNodes; i++){
34     tmpAct = trustRegister[i][NODE_PKTS_FORWARDED];
35     if ( tmpAct > maxNodeAct ){ maxNodeAct = tmpAct; }
36     if ( tmpAct < minNodeAct ){ minNodeAct = tmpAct; }
37     avgAct += trustRegister[i][NODE_PKTS_FORWARDED];
38   }
39
40   avgAct = avgAct / numNodes;
41   lowBoundAct = minNodeAct + ( (avgAct - minNodeAct)/2 );
42   highBoundAct = avgAct + ( (maxNodeAct - avgAct)/2 );
43
44   for (i = 0; i < numHops ; i++){
45
46     nodeAddr = path[i];
47     nodeAct = ( trustRegister[nodeAddr][NODE_PKTS_FORWARDED]) ;
48     nodeRep = trustRegister[nodeAddr][NODE_PKTS_FORWARDED] /
49       (trustRegister[nodeAddr][NODE_PKTS_FORWARDED] + trustRegister[nodeAddr][NODE_PKTS_DROPPED]);
50     nodeSpeed = trustRegister[nodeAddr][NODE_MOBILITY];
51     pathAct += nodeAct;
52     pathRep *= nodeRep;
53     pathMob += nodeSpeed;
54   }
55
56   pathAct = pathAct / numHops;
57   pathMob = pathMob / numHops;
58
59   //discretization of trust metrics
60
61   if (pathAct <= lowBoundAct) {
62     iPathAct = 1;
63   } else if (pathAct > lowBoundAct && pathAct <= highBoundAct){
64     iPathAct = 2;
65   } else if (pathAct > highBoundAct){
```

```
66      iPathAct = 3;
67    }
68
69    if (pathMob >= 0 && pathMob < 1){
70      iPathMob = 1;
71    } else if (pathMob >= 1 && pathMob < 2){
72      iPathMob = 2;
73    } else if (pathMob >= 2 ) {
74      iPathMob = 3;
75    }
76
77    if (pathRep < 0.4){
78      iPathRep = 1;
79    } else if (pathRep >= 0.4 && pathRep < 0.8) {
80      iPathRep = 2;
81    } else if (pathRep >= 0.8){
82      iPathRep = 3;
83    }
84
85    //submit trust metrics to c45 classifier
86    //return 1 for good routes or 0 for bad ones
87    return MetrubiEvalRouteInDTRules( iPathAct, iPathRep, iPathMob, iPathDst );
88
89  }
90
```

C Rotina de Ordenação do Avaliador de Rotas

```
1  ##################################################################################################################
2  # Universidade de Brasília - UnB  * Departamento de Engenharia Elétrica - ENE
3  # Programa de Pós-graduação em Engenharia Elétrica - PPGEE
4  # Modificação da rotina RoutingTdsrInsertRCInOrder() para avaliar a confiança
5  # Autores:
6  #  Luiz Fernando S. Serique Junior <serique@gmail.com>
7  #  Rafael Timóteo de Sousa Junior <desousa@unb.br>
8  # Atualizado em: 13/10/2011
9  ##################################################################################################################
10
11 DSR_RouteCacheEntry *RoutingTdsrInsertRCInOrder(NODE_ADDR destAddr,
12                                                 int hopCount,
13                                                 NODE_ADDR *path,
14                                                 TDSR_RouteCacheEntry *old,
15                                                 TDSR_RouteCacheEntry *last)
16 {
17     TDSR_RouteCacheEntry *newOne;
18     int i;
19     int trust = MetrubiGetRouteTrust( path, hopCount );
20
21     if (old == NULL)
22     {
23         newOne = (TDSR_RouteCacheEntry *)pc_malloc(sizeof(TDSR_RouteCacheEntry));
24         assert(newOne != NULL);
25
26         newOne->destAddr = destAddr;
27         newOne->hopCount = hopCount;
28   newOne->trust = trust;
29
30         for (i = 0; i < TDSR_MAX_SR_LEN; i++)
31         {
32             newOne->path[i] = path[i];
33         }
34         newOne->prev = last;
35         newOne->next = NULL;
36     }
37     //else if ((old->destAddr > destAddr) || ((old->destAddr == destAddr) && (old->hopCount > hopCount)))
38     else if ((old->destAddr > destAddr) || ((old->destAddr == destAddr) && (old->hopCount > hopCount) && (old->trust == trust)) ||
39    ( (old->destAddr == destAddr) && (old->trust < trust) && (old->hopCount == hopCount)) )
40     {
41         newOne = (TDSR_RouteCacheEntry *)pc_malloc(sizeof(TDSR_RouteCacheEntry));
42         assert(newOne != NULL);
43
44         newOne->destAddr = destAddr;
45         newOne->hopCount = hopCount;
46   newOne->trust = trust;
47
48         for (i = 0; i < TDSR_MAX_SR_LEN; i++)
49         {
50             newOne->path[i] = path[i];
51         }
52
53         newOne->prev = old->prev;
54
55         if (old->prev != NULL)
56         {
57             old->prev->next = newOne;
58         }
59         old->prev = newOne;
60         newOne->next = old;
61
62     }
63     else if ((old->destAddr == destAddr) && (old->hopCount == hopCount) && (old->trust == trust) )
64     //else if ((old->destAddr == destAddr) && (old->trust == trust))
65     {
```

```
66          newOne = (TDSR_RouteCacheEntry *)pc_malloc(sizeof(TDSR_RouteCacheEntry));
67          assert(newOne != NULL);
68
69          newOne->destAddr = destAddr;
70          newOne->hopCount = old->hopCount;
71    newOne->trust = old->trust;
72          old->hopCount = hopCount;
73          old->trust = trust;
74
75          for (i = 0; i < TDSR_MAX_SR_LEN; i++)
76          {
77              newOne->path[i] = old->path[i];
78          }
79          for (i = 0; i < TDSR_MAX_SR_LEN; i++)
80          {
81              old->path[i] = path[i];
82          }
83
84          newOne->prev = old->prev;
85          if (old->prev != NULL)
86          {
87              old->prev->next = newOne;
88          }
89          old->prev = newOne;
90          newOne->next = old;
91      }
92      else
93      {
94          newOne = old;
95          newOne->next = RoutingTdsrInsertRCInOrder(destAddr,
96                                                    hopCount,
97                                                    path,
98                                                    old->next,
99                                                    newOne);
100     }
101
102     return(newOne);
103 }
```

D Mapeamento das Alterações no Glomosim

include/api.h

```
struct glomo_node_str {
    /* Common Information  about each node. */
    /* This field represents the simulation id of the node. It is used
       only for simulation purposes and should not be used by the protocol
       code at any layer. For the network address of the node use the next
       field, which is called nodeAddr. */
    unsigned       id;

    NODE_ADDR      nodeAddr;      /* the network address of the node */
    unsigned short seed[3];       /* seed for random number generator */
    unsigned short initialSeedValue[3]; /* First seed value for a node */
    long           numNodes;      /* number of nodes in the simulation */

    GlomoCoordinates position;

    SplayTree splayTree;

    GlomoPartition *partitionData;

    GlomoMobility  mobilityData;

    /* Information about partition nodes */
    GlomoNode      *prevNodeData;
    GlomoNode      *nextNodeData;

    /* Layer specific information about each node. */
    GlomoProp      *propData; /* propagation information */
    GlomoRadio*     radioData[MAX_NUM_RADIOS]; /* radio layer information   */
    int             numberRadios;
    GlomoMac*       macData[MAX_NUM_INTERFACES];  /* mac layer information */
    int             numberInterfaces;
    GlomoNetwork    networkData;   /* network layer information */
    GlomoTransport  transportData; /* transport layer information */
    GlomoApp        appData;       /* application layer information */

    // For Parallel Lookahead Calculation
    int eotCalculatorBackPtrIndex;

    int selfish; //selfish behaviour
    int sleepy;  //sleep mode
};
```

main/nodes.pc

```
int DriverGenerateRandomNodes(GlomoNodePositionInfo *nodeData, int nodeNum, int selfishNodeNum, int sleepyNodeNum,
  GlomoCoordinates terrainDimensions, unsigned short seed[3])
{
    int i;

    for (i = 0; i < nodeNum; i++) {
        nodeData[i].nodeAddr = i;
        nodeData[i].position.x = pc_erand(seed) * terrainDimensions.x;
        nodeData[i].position.y = pc_erand(seed) * terrainDimensions.y;
```

```
        nodeData[i].selfish = 0;
        nodeData[i].sleepy = 0;

        if (i<selfishNodeNum)
        {
         nodeData[i].selfish = 1;
        }

        if (i > (nodeNum - sleepyNodeNum) )
        {
         nodeData[i].sleepy = 1;
        }
    }

    return i;
}
```

network/nwip.pc

```
static
void RoutePacketAndSendToMac(GlomoNode *node, Message *msg) {
   GlomoNetworkIp *ipLayer = (GlomoNetworkIp *)node->networkData.networkVar;
   IpHeaderType *ipHeader = (IpHeaderType *) msg->packet;

   NODE_ADDR path[TDSR_MAX_SR_LEN + 1];
   int length;
   int current;
   int nodeArea;

   //
   // Check if its a broadcast.
   //

   if (ipHeader->ip_dst == ANY_DEST) {
      NetworkIpSendPacketToMacLayer(node, msg, DEFAULT_INTERFACE, ANY_DEST);
   } else {
      // First Try to route with the routing protocol supplied routing
      // function.  If the function doesn't exists or
      // fails to find a route, then try standard lookup table
      // or source routing.

      BOOL PacketWasRouted = FALSE;
      if (ipLayer->routerFunction != NULL) {
         (ipLayer->routerFunction)(
            node, msg, ipHeader->ip_dst, &PacketWasRouted);
      }//if//

      if (!PacketWasRouted) {
         if (IpHeaderHasSourceRoute(ipHeader)) {

                ExtractIpSourceAndRecordedRoute(msg, path, &length, &current);

                if ( (ipHeader->ip_src != node->nodeAddr && node->selfish == 1) ) {
                        RoutingInsertDecisionTreeStats( ipHeader->ip_src , path, length, node, 0 );
                        trustRegister[node->nodeAddr][NODE_PKTS_DROPPED]++;
                        return;
                }

                trustRegister[node->nodeAddr][NODE_PKTS_FORWARDED]++;
                SourceRouteThePacket(node, msg);

         } else {

            RouteThePacketUsingLookupTable(node, msg);
         }//if//
      }//if//
   }//if//
}//RoutePacketAndSendToMac//

```

network/nwip.pc

```
1 void RadioAccnoiseLayer(GlomoNode *node, const int radioNum, Message *msg) {
2     GlomoRadio* thisRadio = node->radioData[radioNum];
3     GlomoRadioAccnoise* accnoise = (GlomoRadioAccnoise *)thisRadio->radioVar;
4
5     if ( node->sleepy == 1 && ( (simclock() % (2*sleepTurnTime)) >= sleepTurnTime ) ){
6         return;
7     }
8
```