PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ROTEAMENTO BASEADO EM CAMINHOS DE MAIOR GRAU DE CONECTIVIDADE APLICADO AO OLSR

GUNNAR RAMOS FERMINO

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul

Orientador: Prof. Dr. Fernando Luís Dotti

Porto Alegre
2009

Dados Internacionais de Catalogação na Publicação (CIP)

F359r Fermino, Gunnar Ramos.

Roteamento baseado em caminhos de maior grau de conectividade aplicado ao OLSR / Gunnar Ramos Fermino. – Porto Alegre, 2009.

141 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Fernando Luís Dotti.

1. Informática. 2. OLSR. 3. Roteamento – Redes de Computadores. 4. Protocolos de Aplicação sem Fio (Protocolos de Rede de Computação). I. Dotti, Fernando Luís. II. Título.

CDD 004.62

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Roteamento Baseado em Caminhos de Maior Grau de Conectividade Aplicado ao OLSR**", apresentada por Gunnar Ramos Fermino, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 21/12/09 pela Comissão Examinadora:

Prof. Dr. Fernando Luís Dotti – Orientador — PPGCC/PUCRS

Prof. Dr. Paulo Henrique Lemelle Fernandes - — PPGCC/PUCRS

Dr. Afonso Henrique Corrêa Sales – — Bolsista PNPD FACIN/PUCRS

Homologada em 29/06/10, conforme Ata No. 011 pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

Dedico este trabalho à minha esposa Rosangela Pereira Neves, aos meus filhos Kenyeli e Kalebe. À minha mãe Maria de Lurdes Ramos, ao meu pai Antonio Firmino (*in memorian*) e aos meus queridos avós João Feliciano Ramos e Jovina Gonçalves de Jesus.

AGRADECIMENTOS

Primeiramente agradeço a Deus pela vida e por me permitir alcançar este tão esperado sonho. Obrigado pela saúde e pela família maravilhosa que me deste e por tantas pessoas sem as quais este trabalho não seria possível.

Agradeço ao meu orientador o professor Dr. Fernando Luís Dotti, pelas orientações e por acreditar na possibilidade da realização desta pesquisa. Sem seus esforços e ajuda, certamente esse trabalho não seria possível.

Agradeço a Fundação de Amparo a Pesquisa do Estado de Mato Grosso – FAPEMAT, e a Universidade do Estado de Mato Grosso por custear as despesas com o curso, e pelo apoio financeiro que permitiu maior dedicação à essa pesquisa.

Aos amigos do MINTER pela amizade e pelas ajudas nos momentos de dificuldade no decorrer desse período. Um agradecimento especial ao amigo e companheiro Diógenes Antônio pelos bons momentos, pelas madrugadas de estudo e por seu apoio e por compartilhar seus ricos conhecimentos, que muito ajudaram na realização desse projeto. Também aos amigos Marcos Paulo de Mesquita e Ivan Pires por compartilharem o vosso tempo e conhecimento e principalmente pela vossa amizade.

Faço uma dedicação toda especial a minha esposa Rosângela Pereira Neves, ao meu filhão Kalebe e a minha filha Kenyeli, pela paciência e compreensão, pois fomos privados de estarmos juntos em vários momentos especiais para que esse trabalho fosse realizado. Por serem a razão de todo meu esforço e a motivação para sempre seguir em frente. Amo vocês.

Aos meus sogros, Adonias da Silva e Marlene Alves por todo apoio e ajuda nesse longo período. Muito obrigado a vocês.

À minha mãe, Maria de Lurdes Ramos, pela ajuda, por acreditar nesse filho e por sempre estar presente quando precisei. Aos queridos avós, João Feliciano e Jovina Gonçalves por tudo que fizeram por mim, sem dúvida sem vocês nunca chegaria até aqui. Enfim a todos os familiares por serem a melhor família que eu poderia ter.

ROTEAMENTO BASEADO EM CAMINHOS DE MAIOR GRAU DE CONECTIVIDADE APLICADO AO OLSR

RESUMO

As redes móveis *ad hoc* são caracterizadas pela alta mobilidade, baixa largura de banda devido às limitações do meio de transmissão *wireless* e pela escassez de energia dos nodos, razões pelas quais a topologia dessas redes sofrem alterações imprevisíveis reduzindo diretamente a taxa de entrega de pacotes na rede. Neste trabalho é destacada a atividade de roteamento em cenários de mobilidade, pois construir protocolos de roteamento que garantam boa taxa de entrega de pacotes em tal ambiente é um dos principais desafios em redes *ad hoc,* visto que uma rota eleita como a melhor em um dado momento não existirá em outro. Neste sentido propõe-se um novo critério de seleção de rotas para o protocolo de roteamento OLSR (*Optimized Link State Routing*) a fim de garantir melhores taxas de entrega de pacotes mesmo em ambientes de alta mobilidade. Nesse critério, uma rota para um destino é selecionada se ela é a menor rota de maior grau de conectividade, o que difere do OLSR original que usa apenas o tamanho da rota como critério. O grau de conectividade de uma rota é dado pela soma do grau de cada nodo desse caminho, ou seja, é dado pela soma da quantidade de vizinhos (enlaces) de cada nodo da rota.

Esse critério se beneficia do fato de o OLSR fazer roteamento dinâmico salto-a-salto, o que permite que cada nodo intermediário, com base nas informações mais atuais de sua tabela de rotas, interfira na rota a ser utilizada no roteamento de pacotes, logo se cada nodo intermediário possuir um maior grau de conectividade, em caso de falhas de enlaces por causa da mobilidade, aumenta-se a possibilidade de que mais rapidamente seja utilizado um caminho alternativo até o destino. A versão do OLSR executando o critério proposto, chamamos de OLSR_PD (*OLSR based in Path Degree*), ou seja, é a versão do OLSR que faz roteamento baseado no grau de conectividade do caminho. Para avaliar o desempenho do OLSR_PD foram realizadas simulações em que parâmetros como mobilidade e número de nodos variaram a fim de validar o funcionamento desse protocolo sob as mais variadas situações de mobilidade, tráfego e escalabilidade. Os resultados experimentais das simulações mostraram que o OLSR_PD alcança substancial aumento na taxa de entrega e de repasse de pacotes em relação ao OLSR original, com insignificante aumento no número de pacotes de controle na rede.

Palavras chaves: Redes móveis *ad hoc*, grau de conectividade, redes *wireless,* OLSR.

ROUTING BASED ON PATHS OF GREATER DEGREE OF CONNECTIVITY APPLIED TO OLSR

ABSTRACT

Ad hoc mobile networks are characterized by high mobility, low bandwidth due to the limitations of the wireless transmission, and by the nodes shortage of energy. The topologies of these networks suffer unpredictable changings, reducing the delivery rates in the network. In this composition it is highlighted the routing activity in mobility sceneries, because building routing protocols that guarantee good delivery packets rates in such environment is one of the main challenges in ad hoc networks, since a route, elected as the best at a given moment will not exist in other. This way, it is proposed a new select criterion of routes for the routing protocol OLSR (Optimized Link State Routing) in order to guarantee better delivery packets rates even in high mobility environments. In this criterion, a destination route is selected if it is the least route of bigger connectivity degree, which differs from the original OLSR that uses only the route sizes as criterion. The connectivity degree of a route is given by the sum of the degree of each node of this way, in other words, is given by the sum of quantity of neighbors (links) of each node in the route.

This criterion is beneficed by the fact of the OLSR to make a dynamic hop-by-hop routing, which allows that each intermediary node, based on the most current information of its table routes, interferes in the route to be used in the packets routing, soon, if each intermediary node has a bigger connectivity degree, in case of links failure caused by mobility, increases the possibility of which more quickly an alternative route is used up to the destination. To the version of the OLSR executing the proposed criterion, we call it OLSR_PD (OLSR based on Path Degree), in other words, it is the version of the OLSR what does routing based on the connectivity degree of the route. To evaluate the performance of the OLSR_PD simulations were carried, in which parameters like mobility and number of nodes varied in order to validate the protocol functioning under the most varied situations of mobility, traffic and scalability. The experimental results of the simulations showed that the OLSR_PD reaches substantial increase in the delivery rate and of transfer of packets regarding the original OLSR, with insignificant increase in the number of control packets in the network.

Keywords: ad hoc mobile networks, connectivity degree, wireless.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACK - *Acknowledgment*
ANSN - *Advertised Neighbor Sequence Number*
AODV - *Ad Hoc On-Demand Distance Vector Routing*
AOMDV - *Ad hoc On-demand Multipath Distance Vector Routing*
ARP - *Adress Resolution Protocol*
ARPANET - *Advanced Research Projects Agency Network*
CBK - *Callback*
CBR - *Constant Bit Rate*
CTS - *Clear to Send*
DBF - *Distributed Bellman-Ford*
DSDV - *Destination-Sequenced Distance-Vector Routing Algorithm*
DSR - *Dynamic Source Routing*
DV - *Distance vector*
DYMO - *Dynamic MANET On-Demand Routing ProtocoL*
E2FT - *End-to-End Fault Tolerant Routing*
EXT - *Expected Transmission Count*
FAPEMAT – Fundação de Amparo a Pesquisa do Estado de Mato Grosso
IANA - *Internet Assigned Names Authority*
IC - *Intervalo de Confiança*
IETF MANET - *Internet Engineering Task Force* MANET
IGP – *Internal Gateway Protocol*
IP - *Internet Protocol*
LL - *Link Layer*
LS - *Link State*
MAC - *Media Access Control*
MANET - *Mobile Ad- hoc NETwork*
MID - *Multiple Interface Declaration*
MPR - *Multipoint relaying*
NRTE – Nenhuma rota disponível
NS - *Network Simulator*
OLSR -*Optimized Link State Routing*
OLSR_PD - *OLSR based in Path Degree*
OLSR-LD - OLSR *Link Delay*
OLSR-ML -

OSPF - *Open Shortest Path First*
PUCRS – Pontifícia Universidade Católica do Rio Grande do Sul
QOLSR - *Quality of Service* OLSR
QoS - *Quality of Service*
RFC - *Request for Comments*
RIP - *Routing Information Protocol*
RTS - *Request to Send*
SMR - *Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks*
TC - *Topology Control message*
TCP - *Transport Control Protocol*
TORA – *Temporally Ordered Routing Algorithm*
TEP – Taxa de Entrega de Pacotes
TRS – Taxa de Repasse de pacotes com Sucesso
TTL -*Time-to-Live*
UDP - *User Datagram Protocol*
UM-OLSR - *University of Murcia* – OLSR
UNEMAT – *Universidade do Estado de Mato Grosso*
ZRP - *Zone Routing Protocol*

SUMÁRIO

1. INTRODUÇÃO

As redes *wireless* são redes em que a transmissão de dados é feita por meio de ondas eletromagnéticas que se propagam pela atmosfera ou pelo espaço, sendo uma alternativa às redes cabeadas por serem de fácil instalação nos mais variados ambientes [KUR06]. O avanço das pesquisas nas áreas de telecomunicações e segurança de redes sem fio, somados à redução do tamanho e dos custos de dispositivos e equipamentos para redes *wireless*, têm provocado um rápido aumento do uso das redes *wireless* em todos os seguimentos da sociedade.

Segundo Kurose e Ross [KUR06] as redes sem fio podem ser classificadas em redes infra-estruturadas ou sem infra-estrutura. Nas redes infra-estruturadas, os nodos são associados a uma ou mais estação base (ponto de acesso), que prestam serviços como, atribuição de endereço e roteamento. Neste tipo de rede toda comunicação entre nodos é intermediada pela estação base, não havendo comunicação direta entre os nodos. Redes *wireless* sem infra-estrutura, também conhecidas como redes móveis *ad hoc* ou ainda como MANET (*Mobile Ad-hoc NETwork*), são redes em que os nodos são móveis e capazes de se comunicar diretamente entre si, formando dinâmica e rapidamente uma rede sem fio temporária, sem a presença de estações base centralizadora. Em MANETs cada nodo funciona como roteador e como estação, pois são capazes de descobrir e manter rotas para outros nodos da rede e ainda executar aplicações dos usuários [ROY99].

Redes móveis *ad hoc* possuem algumas limitações em relação às redes fixas, tais como: toda comunicação é feita por um meio não confiável e com altas taxas de erros; baixa largura de banda dos canais de comunicação; nodos com baixa disponibilidade de energia; e topologia de rede altamente dinâmica. Mesmo com essas limitações, por causa da mobilidade, que muitas das vezes é mais que desejável é necessária, e pela rapidez e facilidade de implementação desse tipo de rede, é que essas redes têm sido largamente pesquisadas no meio acadêmico. Pelas suas características, redes MANETs são especialmente indicadas para áreas onde há pouca ou nenhuma infra-estrutura de comunicação ou onde esta é inviável de ser construída. Exemplos significativos de locais onde esse tipo de rede pode ser empregada são: operações de busca e resgate de emergência em lugares de difícil acesso, em situações de desastres como, terremotos, furacões ou inundações; ou ainda em locais onde pessoas com dispositivos móveis sem

fio queiram compartilhar informações rapidamente [COR08] [ROY99].

Em redes MANETs a comunicação entre nodos pode ser feita de maneira direta ou por múltiplos saltos. No primeiro tipo, cada nodo comunica-se apenas com nodos vizinhos que estiverem dentro do alcance de sua transmissão de rádio. Já no segundo, se dois nodos estão fora do alcance do rádio um do outro e precisam se comunicar, todas as mensagens enviadas entre eles devem passar por um ou mais nodos intermediários que atuarão como roteadores. Por causa do limitado alcance do sinal de rádio dos nodos móveis, quase sempre a comunicação entre nodos é feita por “múltiplos saltos” [CAS02].

1.1. Motivação

Em redes MANETs onde a comunicação é feita por múltiplos saltos; os nodos podem se mover a velocidades constantes ou aleatórias; é baixa a largura de banda disponível; e onde há escassez de energia é constante mudanças na topologia da rede e consequentemente a quebra de enlaces (*links*) de comunicação. Segundo Xue e Nahrstedt [XUE03], desenvolver protocolos de roteamento para tal ambiente, que garanta altas taxas de entrega de pacotes com baixo consumo de recursos da rede, mesmo na presença de nodos ou *links* falhos, é um problema dos mais desafiadores em redes *ad hoc*.

Protocolos de roteamento são softwares que operam na camada de rede dos nodos roteadores de uma rede (dispositivos móveis em MANETs), os quais são responsáveis por escolher quais rotas serão utilizadas na comunicação entre dois nodos, com base nas informações das tabelas de roteamento de cada nodo [KUR06]. À medida que ocorrem mudanças na topologia e *links* de comunicação se tornam falhos, as tabelas de roteamento dos nodos ficam desatualizadas por um período de tempo. Pacotes em trânsito nesse período poderão ser descartados, visto que *links* foram quebrados e as tabelas de rotas dos nodos intermediários ainda não foram atualizadas. O intervalo de tempo entre a ocorrência de alterações na rede até o reconhecimento das mudanças e a atualização das tabelas de rotas é chamado de “latência de convergência” [LAB00].

Os protocolos de roteamento, quanto à construção das tabelas de roteamento, podem ser classificados em três categorias: pró-ativos, reativos e híbridos. Os pró-ativos mantêm em todos os nodos da rede uma tabela de rotas para todos os destinos

conhecidos da rede e a cada mudança na topologia ou de tempos em tempos todas as tabelas de rotas em todos os nodos são atualizadas, começando pelos nodos mais próximos às alterações (falhas), os quais informarão aos demais sobre as mudanças. Protocolos pró-ativos têm como vantagem, o fato de que sempre que uma rota for requisitada estará disponível, todavia ao custo de consumir grande largura de banda com mensagens de controle para atualização de tabelas, o que tende a se agravar com o aumento da mobilidade dos nodos. Os reativos constroem rotas apenas sob demanda, ou seja, apenas quando são requisitadas, porém, dependendo do tamanho da rede, a procura por rotas sob demanda pode acarretar demoras imprevisíveis. Todavia consome menos recursos da rede com a troca periódica de mensagens de controle. Já os protocolos híbridos são construídos contendo características dos pró-ativos e dos reativos [ABO04] [ROY99]. Essas classes de protocolos serão melhor discutidas nas seções do Capítulo 2.

Tanto os protocolos reativos quanto os pró-ativos sofrem degradação na largura de banda e apresentam altas taxas de perdas de pacotes em situações de alta mobilidade dos nodos, todavia, segundo Abolhasan et al. [ABO04] e Royer e Toh [ROY99], os pró-ativos têm desempenho inferior aos reativos em cenários de alta mobilidade e alta densidade de nodos. Várias técnicas e estudos têm sido apresentados a fim de melhorar o desempenho dos protocolos pró-ativos. Uma técnica apresentada no OLSR (*Optimized Link State Routing*) [JAC01] [JAC03] introduz uma nova estratégia de propagação de mensagens de controle na rede, chamada MPR (*multipoint relaying*), que consiste em reduzir ao máximo o número de nodos responsáveis por retransmitir as mensagens de controle na rede. Em Abolhasan et al. [ABO04], foram analisados vários protocolos pró-ativos e o OLSR é o que mostrou melhor escalabilidade e controle de *overhead* em cenários de mobilidade.

Segundo Jacquet et al. [JAC01], o OLSR é um protocolo de roteamento robusto, visto que se recupera rapidamente de falhas provocadas por causa da mobilidade dos nodos e reage bem à latência de convergência provocada pelas mudanças na topologia, pois o roteamento de pacotes é feito de maneira dinâmica salto-a-salto, o que permite que em caso de falhas em rotas, nodos imediatamente anteriores à falha possam interferir na rota de pacotes já em trânsito com base nas informações mais atuais de sua tabela de rotas. Neste sentido, a priorização de rotas de maior grau de conectividade, como proposto neste trabalho, deve aumentar a probabilidade de entrega de pacotes em trânsito em caso de mudança topológica.

Há na literatura várias propostas, como encontradas em Badis et al. [BAD04], Couto et al.[COU03] e Cordeiro et al. [COR07], as quais visam melhorar o desempenho do OLSR em diferentes cenários, empregando critérios de QoS (*Quality of Service*) na seleção dos conjuntos MPR ou na seleção de rotas. Porém, para o critério de seleção de rotas, não foi encontrada proposta tal como sugerida neste trabalho.

1.2. Objetivos

O objetivo deste trabalho é apresentar e avaliar uma extensão ao protocolo de roteamento OLSR, propondo um novo critério para seleção de rotas a esse protocolo. Nesse critério, em cada entrada na tabela de roteamento, o próximo nodo (próximo salto) em direção a um destino é selecionado se ele faz parte da menor rota de maior grau de conectividade, ou seja, primeiro é verificado se a rota encontrada é o menor caminho até o destino. Depois, se existir mais de um caminho de mesmo comprimento é escolhido aquele que possui maior grau de conectividade, o que difere do OLSR original que usa apenas o menor caminho como critério. À versão do OLSR executando esse critério nomeamos como OLSR_PD (OLSR *based in Path Degree*) ou OLSR baseado no grau de conectividade do caminho.

Segundo Drozdek [DRO08], em teoria dos grafos o grau de um vértice (nodo), *d(v),* equivale ao número de arestas *(links)* incidentes nele, logo o grau do caminho pode ser encontrado pela soma do grau de todos os vértices do caminho (vede “grau do caminho” Seção 2.5). Neste trabalho o “grau do caminho” é dado pela soma do grau de todos os nodos intermediários mais o grau do nodo destino. No caso do grafo representado na Figura 1, onde os vértices representam os nodos, os arcos representam os *links* de comunicação e os números abaixo de cada vértice, o grau do vértice. Considere agora, que o nodo origem *A* e o destino *H* precisam se comunicar, e que a rota escolhida passa pelo caminho *c=(A-C-E-H),* neste trabalho, o grau desse caminho é dado por *d(A+C+E+H)={d(C)+d(E)+d(H)}*, ou seja, a soma do grau de todos os vértices intermediários entre origem *A* e o destino *H*, mais o grau do destino *H,* logo o *d(A+C+E+H)=9.*

Esse critério se beneficia do fato de o OLSR fazer roteamento salto-a-salto, visto que os nodos intermediários podem interferir na rota com base em informações mais atuais de suas tabelas de rotas. A ideia básica é que, caso ocorram falhas na rota em uso

e *links* dessa rota se tornem indisponíveis, o nodo imediatamente anterior a falha, sendo um nodo da rota de maior grau, deve ter maior probabilidade, se comparado a nodos de outras rotas, de calcular uma rota alternativa para o destino. Paralelamente a isso, tendo detectado a falha, o nodo deve enviar uma mensagem TC (*Topology Control*) a outros nodos, alterando a topologia da rede. Nodos anteriores a este, na rota de dados, receberão a mensagem TC e, por fazerem parte da rota de maior grau, tem maior chance de calcular uma rota alternativa para o destino, se comparados a nodos de outras rotas. Assim, utilizando este critério, espera-se que os nodos da rota escolhida tenham maior probabilidade de encontrar rotas alternativas para um dado destinatário de dados, de tal forma que, o descarte de pacotes, devido à mudança topológica durante o tráfego de dados, seja minimizado.

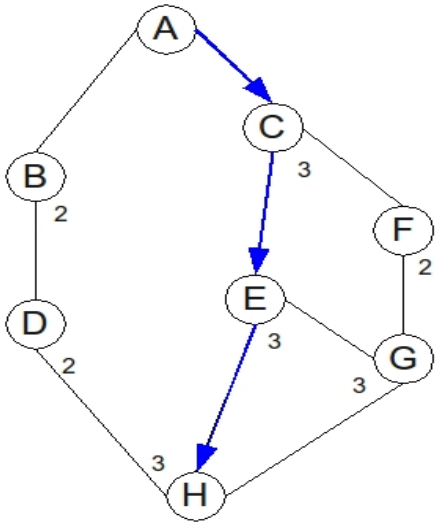

Figura 1 - Grau dos nodos de um grafo

Como resultado da adoção desse critério esperamos que em cenários de mobilidade e constantes mudanças de topologia, seja reduzida a taxa de pacotes descartados, bem como o atraso fim-a-fim e consequentemente aumentando a taxa de entrega de pacotes com pouco ou nenhum aumento no *overhead* na rede, quando comparado à versão original do OLSR.

Para avaliar o desempenho do OLSR_PD, este protocolo fora implementado no NS-2 (*Network Simulator*) [FAL09], e foram realizadas simulações, em que parâmetros como mobilidade e número de nodos variavam a fim de validar o funcionamento desse protocolo sob as mais variadas situações de mobilidade, tráfego e escalabilidade. Os

resultados experimentais das simulações também são apresentados.

1.3. Organização do trabalho

Este trabalho é dividido em seis capítulos, estando organizado da seguinte forma: no Capítulo 1 é feita a introdução, onde são apresentados os objetivos e a motivação para a implementação do protocolo proposto. No Capítulo 2 são apresentados os conceitos fundamentais para o trabalho, como classificações dos protocolos de roteamentos e trabalhos relacionados. No Capítulo 3 é descrito o funcionamento do protocolo de roteamento OLSR e suas principais partes. No Capítulo 4 é descrito a proposta e o processo de implementação do OLSR_PD. No Capítulo 5 são apresentadas as métricas utilizadas para avaliação do desempenho dos protocolos OLSR e OLSR_PD, bem como os resultados obtidos. Por fim no Capítulo 6 são apresentadas as conclusões e propostas para trabalhos futuros.

2. ROTEAMENTO EM REDES DE COMPUTADORES

Neste capítulo são introduzidos alguns dos principais conceitos e trabalhos relacionados sobre roteamento em redes de computadores e principalmente conceitos utilizados no roteamento em redes MANETs. Este capítulo está organizado da seguinte forma: na Seção 2.1 são apresentados os principais algoritmos de roteamento utilizados em redes de computadores; na Seção 2.2 são apresentados os principais conceitos relacionados às MANETs e os principais protocolos de roteamento utilizados nessas redes; na Seção 2.3 são abordados conceitos sobre roteamento tolerante a falhas em redes *ad hoc*; e na última seção, 2.4, são apresentados conceitos e os principais protocolos de roteamento que implementam mecanismos de qualidade de serviço QoS; na seção 2.5 são apresentadas definições para os principais termos utilizados na implementação neste trabalho.

2.1. Algoritmos de roteamento em redes de computadores

A principal finalidade das redes de computadores é possibilitar a comunicação entre computadores e o compartilhamento de recursos. O melhor exemplo de redes é a própria Internet, a qual pode ser definida como um conjunto de redes de diferentes formatos, tamanhos, topologias e arquiteturas. Cada rede na Internet é formada por um ou mais roteadores, os quais interligam essas redes graças à criação de protocolos comuns, como o IP (*Internet Protocol*), que permite que computadores de diferentes redes possam ser localizados, estabelecendo conexões e transmissões de dados entre si.

A comunicação entre computadores na Internet, geralmente é feita em múltiplos saltos, visto que, para que pacotes de dados de uma aplicação rodando em um computador origem alcance um destino, esses são repassados várias vezes pelos grandes roteadores da Internet até que um desses roteadores conheça o endereço da rede do computador destino. A escolha do caminho pelo qual os pacotes serão encaminhados e a definição das estruturas de dados utilizadas no roteamento são realizadas pelos chamados Algoritmos de Roteamento, os quais constituem um dos principais elementos do projeto da camada de redes.

Em uma definição dada por Tanembaun [TAN03], algoritmos de roteamento “é a

parte do software da camada de rede responsável pela decisão sobre a linha de saída a ser usada na transmissão do pacote de entrada", ou seja, são algoritmos, partes de protocolos de roteamento, cuja principal finalidade é calcular e manter atualizadas nos roteadores, as melhores rotas para encaminhar pacotes em direção aos demais destinos.

Segundo Tanembaun [TAN03] certas propriedades são desejáveis em um algoritmo de roteamento dentre as quais destaca-se: correção, simplicidade, robustez, estabilidade, equidade e otimização. A correção e simplicidade são auto-explicáveis; já robustez refere-se à capacidade do algoritmo de roteamento, mesmo em ambientes onde *hosts* (computadores), roteadores e *links* se tornem falhos repentinamente e que as topologias mudem muitas vezes, este deve aceitar as alterações na topologia e no tráfego sem exigir que as tarefas dos demais computadores sejam interrompidas e que a rede seja reinicializada sempre que um roteador falhar. A estabilidade trata da capacidade do algoritmo alcançar o equilíbrio e permanecer nesse estado. A equidade refere-se ao tratamento igualitário a todos os pares origem-destino que desejam comunicar-se na rede, e a otimização refere-se à utilização da banda de tal forma que a eficiência global seja máxima sem, no entanto, comprometer a equidade.

De todas as propriedades citadas, neste trabalho será dado destaque à robustez, visto que essa é uma das características do protocolo de roteamento OLSR que, por ser pró-ativo e fazer roteamento dinâmico, atualiza sua tabela de rotas à medida que falhas e alterações da topologia acontecem. Também a proposta do novo critério de seleção de rotas ao OLSR visa garantir maior robustez, visto que, rotas cujos nodos possuam o maior grau de conectividade entre seus pares, aumenta a possibilidade de que em caso de falha na rota principal, a partir daquele ponto, mais rapidamente uma rota alternativa seja encontrada.

Em Tanembaun [TAN03] é destacada duas categorias principais de algoritmos de roteamento: algoritmos *não adaptativos* e *adaptativos*. Os não adaptativos desconsideram medidas ou estimativas do tráfego e da topologia atual para construir sua tabela de roteamento, visto que, rotas para todos os destinos conhecidos já são calculadas *off-line* e transferidas para os roteadores quando a rede é inicializada, por isso esse procedimento é também conhecido como **roteamento estático.** Já os adaptativos constroem e mudam suas decisões de roteamento, com base nas informações de topologia atual, e eventualmente em informações sobre o tráfego e distância das rotas, por isso esse procedimento também é conhecido como **roteamento dinâmico.** A seguir destacamos

alguns dos principais algoritmos de roteamento conhecidos.

2.1.1. O roteamento pelo caminho mais curto

O algoritmo do caminho mais curto, em suas várias formas, é um dos mais utilizados em protocolos de roteamento. Segundo Tanembaun [TAN03] dado um grafo de uma sub-rede, com cada nodo do grafo representando um roteador e cada arco indicando uma linha de comunicação (*link*), para escolher um caminho entre um par de roteadores, o algoritmo simplesmente encontra o caminho mais curto entre eles no grafo.

O caminho mais curto geralmente é o caminho com menor número de saltos (*hops*), ou seja, o número de arestas entre um nodo origem e um destino. Logo segundo esse critério, os caminhos *A-C-D* e *A-B-D* da Figura 2 são iguais em distância. Outra métrica utilizada é a distância geográfica, ou seja, cada arco é avaliado em termos de metros. Se, por acaso, a Figura 2 possuísse uma escala, o caminho entre *A-C-D* seria visivelmente mais longo. Outros algoritmos rotulam os arcos atribuindo a estes valores, que segundo Tanembaun [TAN03] “podem ser calculados como uma função da distância, da largura de banda, do tráfego médio, do custo da comunicação, do comprimento médio da fila, do retardo médio entre outros”.

Existem vários algoritmos utilizados para calcular o caminho mais curto entre dois nodos de um grafo, dentre eles destacam-se o algoritmo de *Dijkstra [DIJ59]* e o de busca em largura. Neste trabalho é considerado como caminho mais curto aquele que possuir o menor número de saltos e para implementar o algoritmo do caminho mais curto é utilizado o algoritmo de busca em largura.

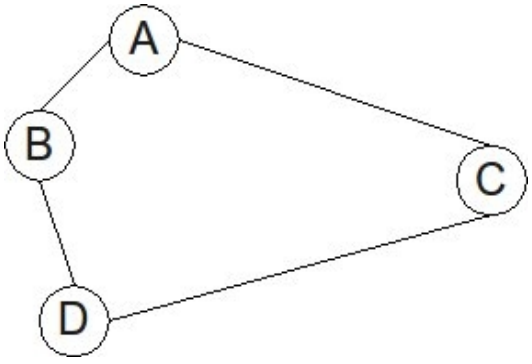

Figura 2 - Grafo de Rede - Caminho mais curto

2.1.2. Roteamento com o algoritmo de vetor de distância

O algoritmo do vetor de distância, mais comumente chamado de Algoritmo de *Bellman-Ford ou Distributed Bellman-Ford* (DBF) *[BEL57]* e algoritmo de *Ford-Fulkerson [FOR62]* foi o algoritmo utilizado na precursora da Internet, a ARPANET (*Advanced Research Projects Agency Network*), e ainda hoje é utilizado no protocolo RIP *(Routing Information Protocol) [MAL08]*. Nesse algoritmo, cada roteador cria e mantém uma tabela contendo a melhor distância conhecida até cada destino da rede e determina qual aresta (próximo roteador) que deverá ser utilizado para alcançá-los. Periodicamente cada nodo envia toda sua tabela de rotas, apenas para seus vizinhos a 1 salto. Ao receber as tabelas de um vizinho, cada nodo atualiza sua tabela de rotas comparando as informações dessa mensagem com as informações de sua própria tabela, se encontrar rotas menores ou novos roteadores de destino, atualiza sua tabela e armazena de onde veio a informação [TAN03].

Esse algoritmo apresenta como vantagem sua simplicidade e eficiência computacional, devido a sua característica distribuída. Todavia apresenta uma baixa convergência na presença de muitas mudanças de topologia, e tende a criar *loops,* principalmente em condições não estáveis, como constantes alterações na topologia da rede, como ocorre na MANETs.

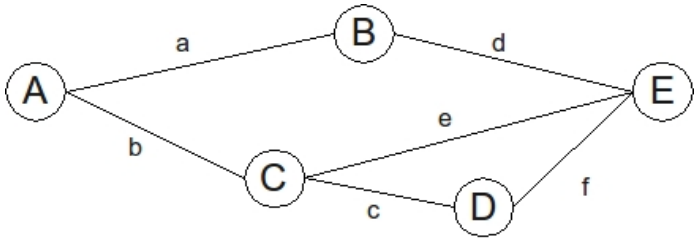

Figura 3 - Grafo representando rede executando o algoritmo Vetor de Distância

Na tabela de roteamento construída em cada roteador, cada entrada de rota é indexada pelo endereço dos outros roteadores da sub-rede. Além da coluna indexadora, esse vetor é composto pelo campo que guarda a aresta de saída a ser utilizada para alcançar o destino, e um campo que guarda o custo para alcançar o destino, que pode registrar a estimativa do tempo ou a distância até esse destino. Na Tabela 1 é

apresentada a tabela de rotas criada pelo roteador *A* do conjunto de redes, representados na Figura 3. A coluna “destino” registra o endereço de cada destino conhecido do roteador A; a coluna “enlace” define qual a aresta de saída para se alcançar o destino; e por fim o campo “custo” recebe o custo para alcançar cada destino em número de saltos.

Tabela 1 - Exemplo da tabela de rotas do algoritmo Vetor de Distâncias criada no roteador A da Figura 3

Destino	Enlace	Custo
B	a	1
C	b	1
D	b	2
E	a	2

2.1.3. Roteamento com o algoritmo do estado do enlace (*Link State*)

Devido a algumas limitações do algoritmo do Vetor de distância, tais como: não levar em consideração largura de banda para calcular o custo de seleção de rotas; e por consumir muito tempo para convergir, ou seja, calcular todas suas rotas, criando *loops* que ficou conhecido como "problema da contagem até o infinito", então em 1979 o DBF deixou de ser utilizado na ARPANET, quando foi substituído pelo algoritmo de roteamento baseado no estado do enlace LS (*Link State*).

Cada roteador, executando esse algoritmo, ao detectar uma alteração no estado do seus enlaces com algum roteador vizinho, então é criado um pacote que levará a informação, apenas do(s) *link(s)* que sofreram alterações, para todos os outros roteadores da rede, diferente do DBF que envia toda a tabela de rota apenas para seus vizinhos. Ao receber esses pacotes, cada roteador atualiza suas informações sobre a topologia dessa rede. Segundo Tanembaun [TAN03], o algoritmo do estado do enlace apresenta convergência mais rápida que o DBF, evita *loops* e ainda suporta mais de uma rota e métrica de seleção de rotas. O funcionamento básico desse algoritmo pode ser resumido em cinco passos [TAN03]:

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

O principal protocolo de roteamento representante do algoritmo do estado do enlace é o OSPF (*Open Shortest Path First*) [MOY89] e foi o protocolo criado para substituir o RIP na ARPANET. Diferentemente do RIP, cada roteador OSPF guarda em sua tabela de roteamento, informações sobre todos os *links* da rede e seu estado atual de forma que possua uma exata representação de toda topologia da rede e dessa forma é possível que cada roteador escolha a melhor (menor) rota para alcançar cada destino da rede. As informações disponíveis nessa tabela são: o endereço identificador da interface de rede do nodo, o número do enlace e a distância ou custo para alcançar o destino.

Segundo Tanembaun [TAN03], atualmente o OSPF é um dos protocolos de roteamento mais empregados na Internet, pois é suportado na maioria dos roteadores e servidores da internet, funcionando bem em redes de pequeno e grande porte.

O OSPF é um protocolo de roteamento projetado para fazer distinção entre *hosts* e roteadores na Internet, o que permite que sejam armazenadas entradas de rotas apenas para os roteadores, reduzindo a quantidade de entradas de rotas, de forma que, se for preciso alcançar algum *host,* basta localizar o roteador (*Gateway*) daquela sub-rede e este saberá localizar o *host.* Já o RIP, fora projetado como um protocolo de roteamento interno (IGP – *Internal Gateway Protocol*), de forma que, esse protocolo só faz roteamento entre *hosts* de uma mesma sub-rede.

Dos protocolos apresentados nessa seção, tanto os algoritmos “vetor de distância” como o “estado do enlace” em suas versões originais não são adequados para implementar roteamento em redes MANETs, devido à mobilidade e limitações de recursos dessas redes. Existem vários algoritmos de roteamento para redes *ad hoc* que exploram características desses dois algoritmos de roteamento, às vezes combinando-os a fim de adequá-los às peculiaridades dessas redes [PER01] [JAC01] [PER03]. Na próxima seção são discutidas as principais características e os principais protocolos de roteamento usados em MANETs.

2.2. Roteamento em redes MANET (*Mobile Ad hoc Network*)

Devido à limitação do alcance do sinal de rádio a comunicação em redes *ad hoc* geralmente é feita em *links* de múltiplos saltos, onde nodos intermediários são utilizados como roteadores na comunicação. Para que esses nodos atuem como roteadores, estes

devem possuir algum protocolo de roteamento que funcione de forma eficiente nesse ambiente. Um protocolo de roteamento é um software que é executado no roteador, o qual se baseia em algum algoritmo ou mecanismo de roteamento.

A atividade de roteamento envolve a determinação de rotas e o repasse de pacotes de dados. Essa função é executada na camada de redes do modelo TCP/IP. Neste trabalho a expressão “pacotes de dados” será usada referindo-se a pacotes enviados pela aplicação, em contraste a “pacotes de controle ou pacotes de roteamento”, que são pacotes gerados para criar e manter a atividade de roteamento.

Segundo Cordeiro [COR07], para se construir protocolos de roteamento para MANETs os seguintes requisitos são desejáveis: prover mecanismos de escolha do melhor caminho para envio e recebimento de pacotes; efetuar roteamento com a menor sobrecarga de rede possível; ser independente da tecnologia da rede; ser capaz de lidar de maneira tolerante a falhas e consistente com as mudanças de topologia, falhas de equipamento e diferentes cargas de tráfego.

Algoritmos tradicionais de roteamento para redes fixas ou mesmo para Internet, como o vetor de distância DV (*Distance Vector*) e do estado do enlace (*Link state*) não são adequados para o ambiente de uma MANET [ABO04]. Para superar este problema, vários protocolos de roteamento têm sido propostos para MANETs. Segundo Abolhasan et al. [ABO04] e Royer e Toh [ROY99], esses protocolos podem ser classificados em três grupos diferentes: *table-driven* ou pró-ativos; *on-demand* ou reativos; e os híbridos, os quais agregam características tanto dos protocolos pró-ativos como dos reativos.

Os protocolos de roteamento também podem ser classificados quanto à técnica de roteamento e podem ser divididos em duas categorias: os que fazem roteamento na origem (*source routing*) e roteamento salto-a-salto (*hop-by-hop*) [ABO04].

O roteamento baseado na origem, é uma técnica em que o nodo origem determina toda a rota por onde os pacotes deverão passar até encontrar o destino, ou seja, o pacote enviado carrega o endereço completo do nodo origem até o nodo destino, restando para os nodos intermediários apenas analisar no cabeçalho o endereço do próximo salto em direção ao destino e repassar o pacote. Essa técnica tem como vantagem o fato de os nodos intermediários não precisar armazenar rotas para todos os destinos. Como desvantagens: não são adequados para grandes redes, onde o número de nodos intermediários é grande e onde ocorrem frequentes falhas de rota; e quanto maior a rede,

maior será o tamanho do cabeçalho dos pacotes, visto que terá de carregar os endereços de todos os nodos intermediários na rota.

No roteamento salto-a-salto, cada pacote carrega apenas o endereço do destino e o endereço do próximo nodo em direção ao destino. Logo cada nodo intermediário deverá usar sua tabela de roteamento para repassar cada pacote em direção ao destino. As vantagens dessa estratégia é que as rotas são sempre atualizadas à medida que ocorrem mudanças na topologia da rede, logo os nodos intermediários repassam os pacotes com base em informações mais recentes e por melhores rotas. Como desvantagem, cada nodo deve manter uma tabela com rotas para todos os destinos conhecidos da rede, o que gera constante tráfego na rede, visto que para manter atualizadas as tabelas de rotas para cada destino é constante o envio de mensagens de controle enviadas por cada nodo para seus vizinhos.

2.2.1. Protocolos pró-ativos

Em uma rede executando um protocolo de roteamento pró-ativo, cada nodo mantém uma ou mais tabelas de rotas para todos os nodos da rede, independentemente do uso ou necessidade dessas rotas. Para manter suas tabelas de rotas atualizadas, de tempos em tempos ou quando ocorrem mudanças na vizinhança, cada nodo propaga para todos os seus vizinhos, informações de sua tabela de rotas. Ao receber essas mensagens os outros nodos atualizam suas tabelas de rotas, e mais adiante propagam essas alterações para os seus vizinhos, até que todos os nodos da rede tenham suas tabelas atualizadas. Dessa forma, quando uma das rotas for requisitada ela pode ser usada imediatamente.

Vários protocolos pró-ativos têm sido propostos para redes *ad hoc*. Em [ABO04] [ROY99] Abolhasan et al. [ABO04] e Royer e Toh [ROY99], são analisados vários desses protocolos, dos quais os dois mais difundidos e estudados são: o DSDV (*Destination-Sequenced Distance-Vector Routing Algorithm*) [PER01] e o OLSR (*Optimized Link State Routing*) [JAC01] [JAC03].

Segundo Abolhasan et al. [ABO04], os protocolos roteamento pró-ativos não são indicados para redes grandes, onde o número de nodos é variável ou ainda quando esses são muito móveis, visto consumirem uma quantia significativa de largura de banda da

rede com mensagens de atualização de topologia da rede. Desses protocolos o que se mostrou melhor escalável é OLSR, o qual diminui o tráfego na rede, reduzindo o número de nodos que reenviam mensagens de atualização, pois cada nodo utiliza-se da técnica MPR (*multipoint relaying*), que consiste em eleger apenas um grupo de nodos que irão repassar suas mensagens de controle. Por ter apresentado o melhor desempenho e escalabilidade entre os pró-ativos, e a fim de melhorar o desempenho desse protocolo em ambientes de alta mobilidade e falhas dos nodos é que escolhemos esse protocolo para que fosse feita a adaptação do novo critério de seleção de rotas proposto neste trabalho. O OLSR será descrito em detalhes no Capítulo 3.

2.2.2. Protocolos reativos

Protocolos reativos têm suas atividades de roteamento iniciadas pelo nodo de origem, ou seja, rotas são criadas apenas quando um nodo de origem deseja enviar dados para um destino qualquer. Por essas características esses protocolos também são classificados como *on-demand*, cujas principais atividades consistem no processo de descobrir rotas e manter rotas. A descoberta da rota se inicia quando um nodo origem precisa de uma rota para um destino, então ele inunda a rede com mensagens de requisição de rotas (*rout_request*). Cada nodo intermediário ao receber a mensagem, se ele não for o nodo destino apenas atualiza seus dados com base nas informações do pacote de requisição de rotas, coloca sua identificação no pacote e repassa as mensagens. Ao encontrar o destino este retorna uma mensagem com a rota descoberta (*route_reply*) através dos nodos intermediários até chegar ao nodo de origem. Quando a rota origem-destino é encontrada o nodo de origem passa a utilizá-la até que esta se torne indisponível ou por um período de tempo também conhecido como tempo de vida TTL (*Time-to-Live*).

Segundo Royer e Toh [ROY99], protocolos de roteamento reativos sob demanda não gastam recursos da rede com rotas desnecessárias, porém o processo de descoberta da rota é mais demorado e até imprevisível.

Em protocolos reativos o processo de manutenção de rota consiste em apagar rotas falhas, ou reiniciar o processo de descoberta de rota em caso de mudança de topologia. A manutenção de rotas depende muito do modelo de detecção de falha disponível nas camadas inferiores.

Vários são os protocolos reativos propostos na literatura, sendo os mais difundidos, segundo [ABO04][ROY99]Abolhasan et al. [ABO04] e Royer e Toh [ROY99], o AODV (*Ad Hoc On-Demand Distance Vector Routing*) [PER03] e o DSR (*Dynamic Source Routing*) [JOH96]. Em experimentos realizados em Perkins et al. [PER01] e Villela e Duarte [VIL03], esses dois protocolos se destacam dentre os protocolos reativos por apresentarem desempenho consideravelmente maior sob diversos aspectos, tais como: maior taxa de entrega de pacotes, e menor sobrecarga de roteamento e média de atraso fim-a-fim.

Segundo Chakeres e Perkins [CHA08], o IETF MANET *Work Group* [MAC97] a partir de experiências adquiridas e com base nas melhores práticas em protocolos de roteamento para redes *ad hoc* tem proposto um protocolo reativo chamado DYMO (*Dynamic* MANET *On-Demand Routing Protocol*) [CHA08]. DYMO é resultado da combinação das melhores características de dois protocolos anteriores a ele o DSR e AODV. Nenhum desses protocolos serão detalhados nesse trabalho, visto não contribuírem para os objetivos do mesmo.

2.2.3. Protocolos híbridos

Segundo Abolhasan et al. [ABO04], protocolos de roteamento híbridos possuem tanto as características dos protocolos pró-ativos como dos reativos. Estes protocolos foram projetados para permitir maior escalabilidade da rede. Para tanto, nodos que estão no alcance uns dos outros são vistos como um grupo. O processo de manutenção de rotas dentro de um grupo é feito por algoritmos de roteamento pró-ativos evitando sobrecarregar toda rede com mensagens de descobertas de rotas. Quando um destino está em outros grupos então é utilizado um processo reativo para descobrimento de rotas. Muitos dos protocolos híbridos propostos são baseados em zonas ou áreas, ou seja, cada grupo de nodos que estão ao alcance do outro a apenas 1 salto são considerados uma zona e os outros grupos como árvores ou grupos de nodos. Vários protocolos híbridos têm sido propostos na literatura, sendo um dos principais e mais difundido o ZRP (*Zone Routing Protocol*) [HAS08]. Em Abolhasan et al. [ABO04], vários protocolos híbridos são analisados, e segundo esse autor os protocolos de roteamento híbridos têm o potencial de prover maior escalabilidade que protocolos puramente pró-ativos ou reativos. Isto porque eles tendem reduzir o número de nodos que reencaminham requisições de rotas.

2.3. Roteamento tolerante a falhas em MANETs

A grande maioria dos algoritmos de roteamento, propostos para MANETs, são projetados para trabalhar em ambientes de comportamento ideal onde nodos e *links* não falham. Segundo Xue e Nahrstedt [XUE03], tais protocolos não são indicados para redes onde é alto o índice de falhas, pois apresentam altas taxas de perdas de pacotes. Em simulações feitas por Xue e Nahrstedt [XUE03] o DSR, um dos protocolos mais difundido em MANETs, quando submetido a testes em ambientes com 20% de falhas de rotas, apresentou perda de 30% dos pacotes enviados, índice intolerável por algumas aplicações.

A principal problemática quando se propõe construir um protocolo de roteamento tolerante a falhas é garantir alta taxa de entrega de pacotes, mesmo na presença de falhas na rede sem, no entanto consumir grandes quantidades de largura de banda e recursos da rede. Visando atender a tais objetivos, várias propostas de protocolos de roteamento tolerante a falhas têm se utilizado da redundância de pacotes, os quais são enviados por múltiplos caminhos simultaneamente entre os pares, origem e destino, ou ainda o uso de rotas alternativas que são guardadas para o caso de ocorrer falhas na rota principal. Tais protocolos têm alcançado altas taxas de entrega de pacotes, todavia muitos destes apresentam um grande consumo de recursos da rede por causa da duplicação de pacotes enviados por múltiplos caminhos entre a origem e o destino [XUE03].

2.3.1. Roteamento tolerante a falhas por múltiplos caminhos

Conforme Xue e Nahrstedt [XUE03], a redundância é um dos principais recursos utilizados para tornar um sistema tolerante a falhas. Redes móveis *ad hoc* são altamente redundantes, pois entre um nodo origem e destino pode existir vários caminhos diferentes. Tal redundância em redes *ad hoc* facilita a implementação de protocolos de roteamento tolerante a falhas.

Existem basicamente dois tipos de protocolos de roteamento tolerante a falhas que fazem roteamento por múltiplos caminhos. O primeiro se utiliza tanto da redundância de pacotes como de rotas, pois para cada pacote gerado é enviado uma cópia por todos os caminhos redundantes entre origem e destino acarretando em alto consumo da largura de banda disponível. O segundo apenas cria e guarda as rotas redundantes na tabela de

roteamento, utilizando apenas uma por vez, caso a primeira venha a falhar, outras rotas já estarão disponíveis. Geralmente esses protocolos fazem roteamento na origem (*source routing*), visto que o nodo origem é quem tem de calcular e guardar rotas redundantes para um dado destino.

Segundo Oomm e Misra [OOM06], vários protocolos baseados no roteamento por múltiplos caminhos tem sido criados. Esses protocolos são fortes em tolerância à falhas, todavia, a maioria desses algoritmos introduzem uma quantia desnecessária de *overhead* na rede. Segundo Kim et al. [KIM06], os principais algoritmos de roteamento baseados em múltiplos caminhos para redes *ad hoc* são: o *End-to-End Fault Tolerant Routing* (E2FT) [XUE03] o qual segundo Oomm e Misra [OOM06], é capaz de significativa redução no *overhead* com envio de múltiplos pacotes, enquanto garante uma satisfatória taxa de entrega de pacotes; o *Ad hoc On-demand Multipath Distance Vector Routing (*AOMDV) [MAR02] é um dos mais notáveis protocolos de roteamento por múltiplos caminhos em pesquisas em redes *ad hoc*, o qual se baseia nos conceitos do AODV. Para alcançar múltiplas rotas ele aceita múltiplos requisições de rotas e mantém uma tabela de múltiplos caminhos para todos os destinos; o *Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks* (SMR) [LEE01] tal como o AOMDV emprega mecanismos de descoberta de caminhos disjuntos para prover rotas livres de falhas.

2.3.2. Roteamento dinâmico, tolerante a falhas baseado na avaliação do fluxo máximo

Em Shoroeder e Junior [SCH07], é proposto um algoritmo de roteamento dinâmico, tolerante a falhas para funcionar na Internet. Nessa proposta o principal critério de roteamento é a seleção de rotas robustas. Rotas robustas no sentido de rotas em que os nodos intermediários possuam um alto grau de conectividade, pois em caso da ocorrência de falhas em parte do caminho (nodos ou *links*), é possível encontrar um caminho alternativo que parte do ponto em que a falha foi conhecida em direção ao nodo destino, o que faz esse algoritmo tolerante a falhas.

Segundo Shoroeder e Junior [SCH07], esse algoritmo é dinâmico, no sentido que, cada nodo ao receber uma mensagem a ser roteada escolhe apenas a próxima aresta da rota, permitindo que nodos intermediários da rota possam escolher um caminho melhor que os conhecidos pelo roteador de origem da mensagem. Esse roteamento explora o fato de que nodos mais próximos a um evento de alteração de estado de nodo ou *link,*

recebam a informação do evento antes dos demais nodos da rede. Esse conceito pode ser estendido também às informações de congestionamento, caso em que uma aresta pode ser considerada temporariamente falha.

Para que esse algoritmo funcione é necessário que cada nodo possua uma representação local da topologia da rede, mesmo que algumas entradas sejam desatualizadas esse algoritmo funcionará por ser tolerante a falhas. Essa representação é feita através de uma estrutura de grafos simétricos, com um conjunto de vértices (nodos) e de arestas (*links*). Essa estrutura é atualizada periodicamente através de troca de mensagens. Como exemplo da representação da rede em cada nodo, suponhamos a rede ilustrada na Figura 4. Em cada nodo haverá uma representação da topologia com os seguintes conjuntos: o conjunto de nodos V = {*s, a, b, c, d, e, f, t*} e o conjunto de arestas E = {*(s, a), (s, b), (a, c), (a, e), (a, s), (b, f), (b, s), (c, a), (c, d), (d, c), (d, t), (e, a), (e, t), (f, b), (f, t), (t, d), (t, e), (t, f)*}.

Para calcular rotas em que os nodos intermediários possuam maior redundância de *links* em direção ao destino, é utilizado o algoritmo do fluxo máximo proposto por Ford Fulkerson [FOR62]. Segundo Shoroeder e Junior [SCH07], esse algoritmo “é utilizado para avaliar a redundância de caminhos, visto que, quanto maior o fluxo máximo, maior a quantidade de caminhos disjuntos até o destino e, por consequência maior é o número de atalhos que podem ser utilizados em caso de falha”. Por exemplo, na rede representada na Figura 4, se o caminho escolhido pelo nodo origem *s,* para chegar ao destino *t,* se iniciasse a partir da aresta *(s, a)*, em caso de falha da aresta *(a, e)*, o nodo *a* possui rotas alternativas, como a aresta *(a, c)*, o qual ainda possui um caminho até o destino.

Esse algoritmo recebe como entrada um par de nodos da rede, correspondentes a origem e ao destino de uma mensagem a ser enviada. O algoritmo então é executado em cada nodo da rede, iniciando pelo nodo de origem, escolhendo o próximo nodo da rota dentre os seus nodos vizinhos. Quando a mensagem chega a um nodo escolhido, esse executa o mesmo algoritmo para escolher o nodo seguinte, e assim, até que o destino seja alcançado [SCH07].

Conforme Shoroeder e Junior [SCH07], na escolha de um caminho também é utilizado um critério secundário, o caminho de menor comprimento. Logo para avaliação de uma aresta, é utilizada aquela, cujo caminho seja o menor caminho de maior redundância até o destino que passa por aquela aresta. Por exemplo, na rede representada pela Figura 4, caso *s* precise enviar uma mensagem para *t*, a aresta *(s, a)*

será avaliada. Observa-se que, por essa aresta possuir duas rotas que leva ao destino *t*, cada uma com comprimentos diferentes. Neste caso, para a avaliação da aresta *(s, a)* será utilizado o menor caminho que passa por essa aresta que é o caminho *(s, a, e, t)*.

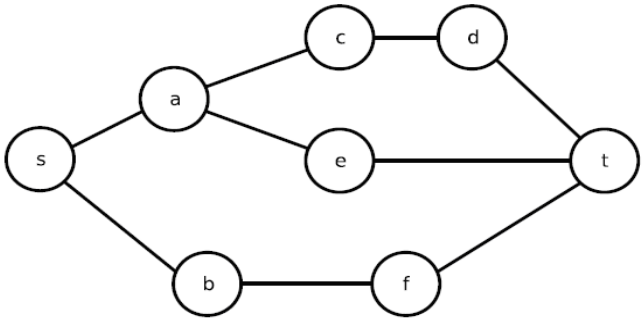

Figura 4 - Representação da topologia em forma de grafo – Fonte: [SCH07]

Para manter atualizas as tabelas de topologia em cada nodo periodicamente são enviadas mensagens de atualização, na presença ou não de alterações de topologia da rede. Cada mensagem contém um conjunto de triplas: *uma aresta; a situação ou estado da aresta* (funcional ou falho); e um contador para ordenação dos eventos *(time stamp).* Um nodo ao receber uma mensagem de atualização inclui ou substitui informações de roteamento em tabela de rotas. Se um nodo observa que durante um intervalo de tempo ele não recebe nenhuma mensagem por uma aresta, então aquela aresta é considerada como falha, e essa informação é enviada para os outros nodos. Caso um nodo receba uma informação de uma aresta antes inexistente, este insere essa informação na sua tabela de topologia e a aresta passa a ser considerada funcional. Provas do funcionamento desse algoritmo são apresentadas em Shoroeder e Junior [SCH07].

2.4. Roteamento baseado em QoS (*Quality of Service*)

Em redes de computadores um dos principais critérios para seleção de rotas utilizados por protocolos de roteamento é o número de saltos. Ocorre que para algumas aplicações do usuário apenas esse critério pode não garantir o funcionamento satisfatório de aplicações multimídias e de tempo real, visto que esses serviços necessitam de garantia de que alguns parâmetros como largura de banda, atraso e perdas de pacotes

estejam dentro de limites bem definidos. Segundo Leguay [LEG06], para tais requisitos dá-se o nome de Qualidade de Serviço (QoS), que pode ser definida como a capacidade da rede prover serviço de encaminhamento de dados de forma consistente e previsível.

Em redes de computadores, QoS pode ser entendida como a capacidade da rede, através dos mecanismos de reserva de largura de banda e priorização de tráfego, fornecer garantias de que determinados fluxos de tráfego irão ter tratamento diferenciado.

É importante destacar que a implementação de protocolos de roteamento que atendam requisitos de QoS tendem a consumir mais largura de banda da rede e recursos de processamento dos roteadores. Para que informações sobre a qualidade dos *links* (vazão, atraso etc...) cheguem a todos os nodos de uma rede, novas mensagens precisam ser trocadas entre os nodos, consumindo assim largura de banda adicional. Para descobrir a rota que possui a melhor QoS, se faz necessário analisar todas as rotas possíveis entre os nodos origem e destino, o que gera processamento extra nos roteadores [CRA98].

A complexidade de fazer roteamento baseado em QoS em MANETs é exatamente não consumir recursos, como largura de banda e tempo de processamento, visto largura de banda e fonte de alimentação (energia elétrica) são as principais limitações dessas redes. Adicional a essas limitações, em redes *ad hoc* os nodos são móveis, fazendo com que a QoS de rotas mudem em tempos imprevisíveis. Na Seção 3.5 são apresentadas algumas características de alguns protocolos de roteamento para MANETs que implementam QoS.

2.5. Definições Preliminares

Nesta seção são apresentadas as principais definições utilizadas para descrever o critério proposto neste trabalho.

1. **Grafo:** É possível definir grafo como um par $G = (V,E)$, onde V é um conjunto não vazio e finito de vértices, e um conjunto E de arestas (arcos), que pode ser vazio, é uma relação entre dois vértices pertencentes a V. Dado v_1, v_2 pertencentes a V, e uma aresta entre estes vértices $\{v_1,v_2\}$ esse é um grafo simples, ou um grafo não orientado se $\{v_1,v_2\}=\{v_2,v_1\}$ como representado na Figura 5 e é um grafo direcionado ou grafo orientado, ou ainda apenas **dígrafo,** se cada aresta é denotada por (v_1,v_2) de forma que

$(v_1,v_2) \neq (v_2,v_1)$, essa diferença é representada pela seta em uma ou ambas as extremidades do arco das arestas (Figura 6) [DRO08]. Para representar arestas entre v_1,v_2 essa distinção será desprezada e uma aresta entre dois vértices, será referenciada apenas como (v_1,v_2). Neste trabalho grafos serão utilizados para representar as topologias das redes, onde os vértices representam os nodos (roteadores) e as arestas os *links* ou enlaces de comunicação entre um par de nodos.

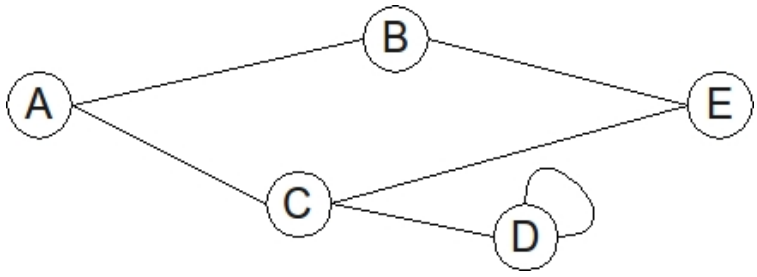

Figura 5 - Exemplo de grafo não orientado

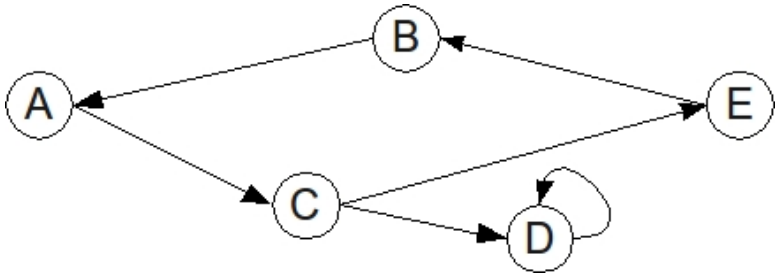

Figura 6 - Exemplo de grafo orientado (dígrafo)

2. **Adjacência:** Um vértice v_1 é adjacente ao vértice v_2, ambos pertencentes a V, se existe uma aresta entre eles, ou seja, se a ponta inicial do arco está em v_1 é a ponta final em v_2. Em uma relação de adjacência pode acontecer de v_1 ser adjacente a v_2, sem que v_2 seja adjacente a v_1. Tomando como exemplo a Figura 5 o vértice *C* possui 3 vizinhos adjacentes: os vértices *A, D* e *E.*

3. **Grau de um vértice:** Em teoria dos grafos o grau de um vértice *v, d(v),* equivale ao número de arestas *(links)* incidentes ao vértice *v.* Os *loops* são contados duas vezes, por exemplo, o vértice *D* do grafo da Figura 5 possui grau igual a *d(D)=3.* Em dígrafos existem os conceitos de grau de entrada, que representa o número de arestas que

chegam a um vértice *v*, e o grau de saída, que é dado pelo número de arestas que partem de um vértice *v.* Logo em um dígrafo, o grau de um vértice é dado pela soma dos graus de entrada e saída, como se observa no vértice *C* do grafo da Figura 6, cuja aresta de entrada é *(A,C)* e as arestas de saída são *(C,E)* e *(C,D),* logo *d(C)=3*.

4. **Caminho em um grafo:** Um caminho *c* de v_1 a v_n é uma sequência de arestas $(v_1,v_2), (v_2,v_3), \ldots,(v_{n-1},v_n)$ e é denotado como $c=(v_1, v_2, v_3, \ldots, v_{n-1}, v_n)$. O comprimento de um caminho é dado pelo número de arestas que o caminho usa [DRO08]. Nesse trabalho o comprimento de um caminho, ora será referenciado como a distância ou como número de saltos para se alcançar um destino. Já o termo rota será utilizado como sinônimo para caminho, pois é o mais comumente utilizado no contexto de redes de computadores para designar o caminho pelo qual um dado nodo origem envia uma mensagem a um destino.

5. **Grau de um caminho:** O grau do caminho *c* é denotado pela soma do grau de todos os vértices do caminho $d(c)=d(v_1)+ d(v_2)+d(v_3)+_{\ldots}d(v_{n-1})+d(v_n)$. Neste trabalho o "grau do caminho" é dado pela soma do grau de todos os nodos intermediários aos nodos origem e destino, mais o grau do nodo destino $d(c)=d(v_2)+d(v_3)+_{\ldots}d(v_{n-1})+d(v_n),$ o grau do nodo origem não foi considerado nos cálculos, por não influenciar nos resultados e para não adicionar mais atividade de processamento no protocolo. Com base na Figura 5, o grau do caminho entre a origem *A* e o destino *E* seguindo pelo caminho *c=(A,C) (C,E)* é igual a *d(A+C+E)=d(C)+d(E),* como *d(C)=3* e *d(E)=2,* então *d(A+C+E)=5*.

3. OPTIMIZED LINK STATE ROUTING (OLSR)

O OLSR (*Optimized Link State Routing*) [JAC01] [JAC03] é um protocolo que executa o roteamento salto-a-salto, pró-ativo, orientado a tabelas de roteamento, o qual é uma versão otimizada do protocolo puramente baseado no estado do enlace *(Link State*). Como em todos os algoritmos baseados no estado do enlace, cada nodo mantém atualizadas as informações sobre a topologia da rede através da troca periódica de mensagens de controle. Segundo Abolhasan [ABO04], esse protocolo tem se destacado entre os pró-ativos, porque reduz significativamente o número de retransmissões de mensagens de controle na rede. Para isso utiliza-se de duas técnicas: primeiro apenas um grupo de nodos é selecionado para propagar mensagens de controle na rede, técnica esta chamada MPR (*Multipoint Relaying*), ou seja, múltiplos pontos de retransmissão. Cada nodo possui o seu grupo de nodos MPR para difundir suas mensagens de controle; segundo, porque diminui o tamanho dos pacotes de controle difundidos na rede, visto que cada nodo MPR informa apenas o endereço dos nodos que o selecionou como MPR. Segundo Jacquet et al. [JAC01], por essas características, este protocolo é adequado para operar em grandes e densas redes *ad hoc* suportando centenas de nodos.

Por causa da natureza proativa do OLSR, enquanto um nodo permanecer ativo, deve manter rotas para todos os destinos conhecidos da rede. A rota ótima, segundo o critério de seleção de rotas do OLSR, é aquela de menor comprimento em termo de número de saltos.

No OLSR cada pacote de controle enviado possui um número de sequência, o qual é incrementado a cada pacote enviado, tornando possível a distinção entre informações mais velhas das recentes. Logo não importa a ordem de chegada dos pacotes, o OLSR usará sempre as informações mais recentes. Segundo Jacquet et al. [JAC01] mensagens de controle são trocadas entre nodos através de comunicação não segura usando o protocolo da camada de transporte UDP (*User Datagram Protocol*) e a porta 698 a qual tem sido designada pela IANA (*Internet Assigned Names Authority*) exclusivamente para ser usada pelo OLSR. Pequenas perdas de mensagens de controle, devido a não garantia de entrega de pacotes do UDP, são perfeitamente suportadas pelo OLSR, visto que as informações dos pacotes perdidos são logo compensadas por novas mensagens enviadas periodicamente.

O OLSR utiliza dois tipos básicos de mensagens de controle: mensagem HELLO e mensagem de controle de topologia (TC – *Topology Control*). Uma mensagem HELLO é enviada por cada nodo apenas para os seus vizinhos a 1 salto, a qual é utilizada para fornecer informações sobre o estado dos *links* desse nodo com seus vizinhos. As informações das mensagens HELLO são utilizadas para que cada nodo construa seu conjunto MPR e a lista de nodos que o tem selecionado como MPR (MPR *selector*). Já as mensagens TC são enviadas por um nodo para todos os outros nodos da rede contendo informações da lista de MPR *selector* daquele nodo. As mensagens TC só podem ser retransmitidas (repassadas) pelos nodos MPRs [JAC01]. Esses conceitos serão discutidos em mais detalhes nas próximas seções.

O OLSR adota um formato unificado de pacote para transmitir qualquer mensagem de controle ou de dado relacionado ao protocolo (Figura 7). A finalidade de usar um formato único é facilitar possíveis extensões do protocolo sem perder compatibilidade com as definições básicas do protocolo. Em um único pacote básico do OLSR é possível transportar vários tipos de mensagens até que o tamanho do pacote alcance o máximo permitido pela rede.

Como se observa na parte superior da Figura 7 existe uma sequência de números em que cada dígito correspondente a 1 *bit* os quais têm por finalidade indicar o tamanho de cada campo. Este pacote é dividido em duas partes, conforme segue-se [JAC03]:

1. Cabeçalho do pacote que é composto dos campos:

    - *Packet Length* - é o comprimento do pacote que é dado em *bytes;*
    - *Packet sequence number* - é número de sequência do pacote que é o identificador único do pacote, o qual é incrementado em 1 a cada novo pacote gerado.

2. Cabeçalho da mensagem que é composto dos campos:

    - *Message type* - tipo da mensagem que identifica que tipo de mensagem é encontrada no campo MESSAGE;
    - *Vtime* - campo que registra o tempo de validade da mensagem;
    - *Message size* - contém o tamanho da mensagem em *bytes*;
    - *Originator address* - carrega o endereço da principal interface de rede do nodo que originou a mensagem;

- *Time to Live* – o campo tempo de vida da mensagem recebe um número inteiro que representa a quantidade máxima de vezes que o pacote pode ser repassado, esse valor é decrementado em 1 a cada salto até chegar a 0 (zero) quando a mensagem deve ser descartada;
- *Hop count* - contador de salto registra por quantos nodos essa mensagem já passou. A cada salto, esse número é incrementado em 1;
- *Message sequence number* - é número de sequência da mensagem que é incrementado a cada nova mensagem partindo daquele nodo.

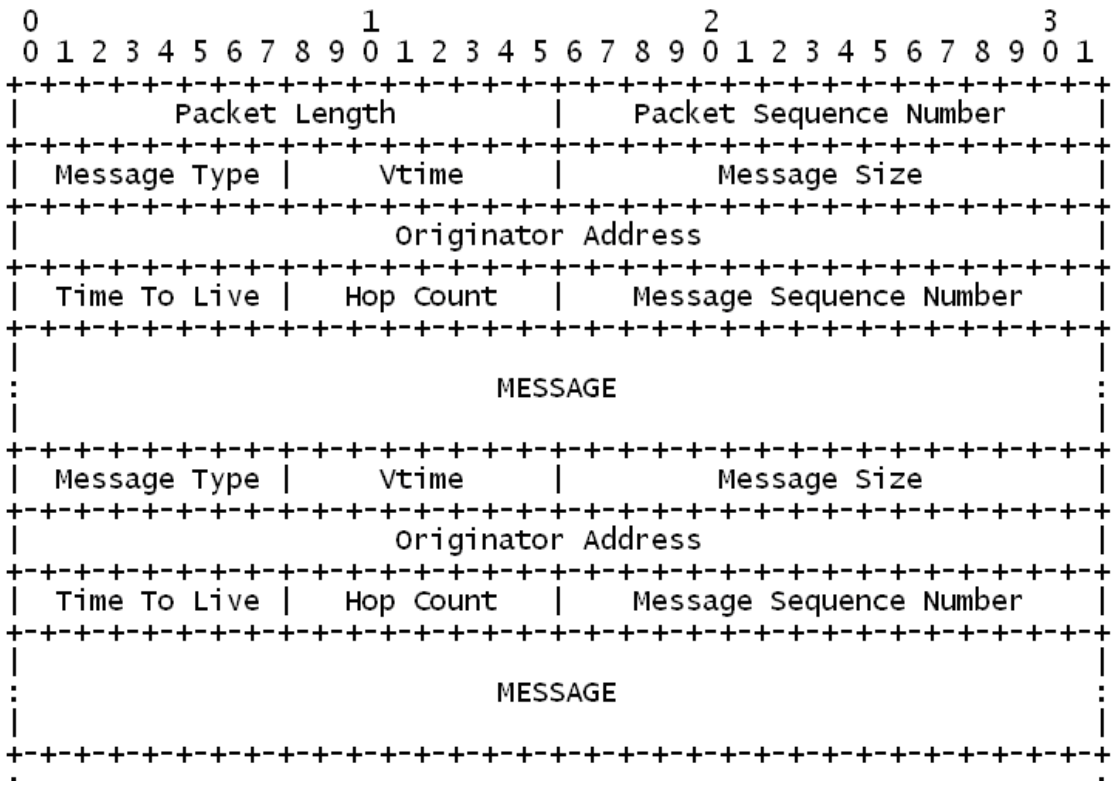

Figura 7 - Formato unificado do pacote básico do OLSR – Fonte: [JAC01]

Um mesmo pacote pode carregar vários tipos de mensagens, como HELLO e TC. Maiores detalhes sobre cada campo do pacote básico pode ser obtido em Jacquet et al. [JAC01].

3.1. Multipoint Relays (MPR)

A técnica utilizada pelo OLSR para reduzir o número de pacotes de controle

repassados na rede, chama-se MPR (*Multipoint Relaying*) que diz que cada nodo deve selecionar, dentre seus vizinhos a 1 salto, o menor número de nodos que irão repassar suas mensagens de controle. O principal critério para que um nodo selecione seu conjunto MPR é que esse grupo de nodos alcance todos os nodos a 2 saltos de distância do nodo origem.

Para escolher o conjunto de nodos MPRs, são utilizadas as informações dos nodos vizinhos alcançáveis a 1 salto contidas nas mensagens do HELLO. Segundo Jacquet et al. [JAC01] e Qayyum et al. [QAY02], o conjunto MPR do nodo *S*, é chamado MPR(*S*). Um nodo só é escolhido como MPR(*S*) se ele possuir um *link* bidirecional para *S* e alcançar o maior número de nodos vizinhos a 2 saltos do nodo *S*. É importante destacar que quanto menor o número de nodos no conjunto MPR(*S*) mais eficientemente esse protocolo funcionará, pois apenas os nodos MPR(*S*) podem reencaminhar mensagens de controle geradas em *S*. Quanto aos demais nodos vizinhos a 1 salto de *S*, mas que não fazem parte do conjunto MPR(*S*), apenas leem e processam as informações que estão nos pacotes, todavia não podem repassá-las, como se observa na Figura 9.

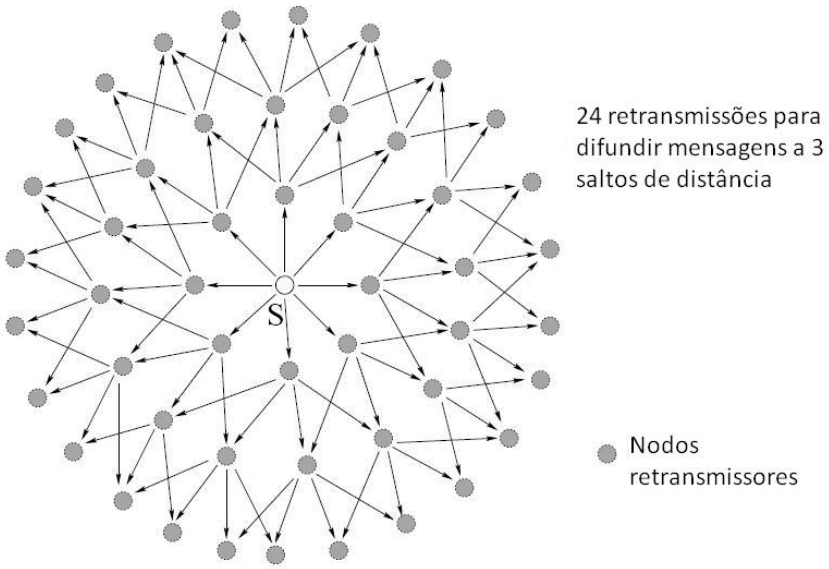

Figura 8 - Difusão normal – Fonte: baseado em [QAY02]

No OLSR apenas os nodos selecionados como MPR são utilizados como nodos intermediários/roteadores para a formação das rotas. Para implementar este esquema, com base em informações do tipo de *link* da mensagem HELLO, cada nodo mantém uma lista dos vizinhos a 1 salto que o tem selecionado ele como MPR (*MPR Selector*) e periodicamente espalha essa lista por toda a rede. Ao receber essas informações, cada

nodo calcula e atualiza suas rotas para cada destino conhecido na rede. Logo, em uma rede de nodos OLSR, rotas sempre serão uma sequência de saltos através de MPRs até um determinado destino [JAC01] [QAY02].

Na Figura 8 é apresentado o processo de envio de mensagens de controle por difusão de pacotes sem MPR. Nesse exemplo para alcançar nodos a 3 saltos do nodo *S* foram necessárias 24 nodos retransmissores. Já na Figura 9 é apresentado o processo de envio por difusão usando MPR, observa-se significativa redução do número de retransmissões de 24 para apenas 11 e alcançando todos os nodos a 3 saltos. Em ambas as ilustrações a mensagem de controle a ser difundida parte do nodo *S* no centro.

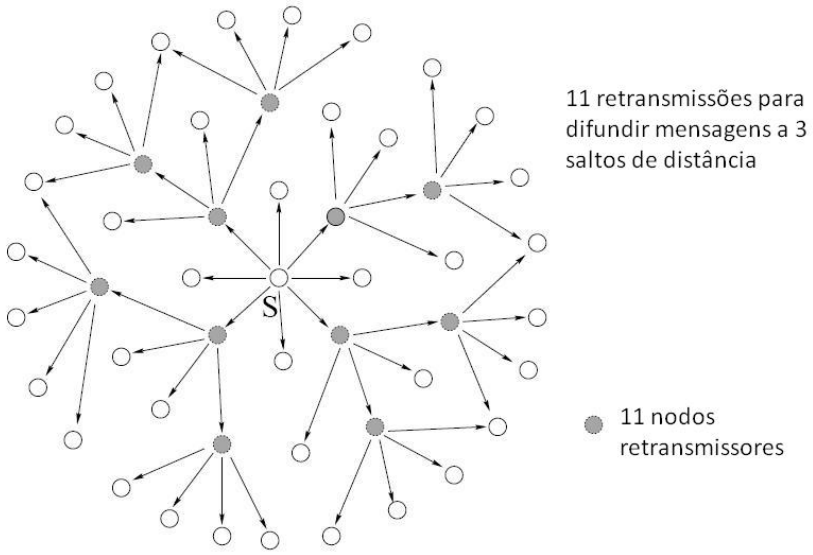

Figura 9 - Difusão com MPR – Fonte: baseado em [QAY02]

Ao receber novas mensagens HELLO, o conjunto MPR de um nodo é recalculado sempre que:

- mudanças na vizinhança a 1 salto são detectadas. Quebra de *link* bidirecional, ou quando um novo vizinho com *link* bidirecional é detectado;
- mudanças nos vizinhos a 2 saltos com *links* bidirecionais são detectadas.

Um dos principais critérios para seleção do conjunto de nodos MPR é a alcançabilidade, ou seja, a capacidade de alcançar o maior número de nodos a 2 saltos do nodo *S*. Ao escolher tais nodos busca-se diminuir o conjunto MPR ao mínimo de nodos suficientes para alcançar os benefícios do MPR. Pode ocorrer, no entanto, do conjunto

MPR(*S*) coincidir com o conjunto de todos os vizinhos a 1 salto. Situação que ocorre toda vez que a rede é inicializada. Mais detalhes sobre o cálculo do MPR de um nodo, e provas do funcionamento desse algoritmo são encontrados em [JAC02].

3.2. Mensagens de controle do OSLR

O OLSR utiliza dois tipos básicos de mensagens de controle para criar e manter as informações de rotas da rede sempre atualizadas. A mensagem mais básica do OLSR é a mensagem HELLO que cada nodo envia apenas para seus vizinhos a 1 salto, a qual é utilizada para detecção do estado dos *links* com vizinhos a 1 salto. Para construção e divulgação de informações da tabela de rotas de cada nodo, são enviadas periodicamente mensagens de controle de topologia (TC) as quais são propagadas para todos os nodos da rede pelos nodos MPRs.

3.2.1. Detecção de nodos vizinhos – mensagem HELLO

Para que um nodo possa escolher seu conjunto MPR, primeiro ele precisa detectar seus vizinhos a 1 salto. No OLSR, cada nodo detecta seus vizinhos enviando periodicamente mensagens de HELLO para todos os nodos ao alcance de sua transmissão de rádio. Esses vizinhos por sua vez processam as mensagens, mas não as repassam, pois o valor do campo *"Time to Live"* do cabeçalho da mensagem HELLO é igual a 1 e com apenas um salto o tempo de vida da mensagem expira não podendo mais ser repassada adiante (Figura 7). Na implementação do OLSR utilizada nesse trabalho [ROS08] o intervalo de tempo padrão para envio de mensagens de HELLO é de dois segundos;

Segundo Jacquet et al. [JAC01] [JAC03], para que nodos de *links* unidirecionais não sejam escolhidos como MPR, todos os *links* devem ser checados em ambas as direções. Por exemplo, se o nodo *A* recebe uma mensagem HELLO de *B*, então o nodo *A* marca em sua tabela de rotas o *link* para o nodo *B* como sendo assíncrono (unidirecional). Ao criar uma mensagem HELLO, o nodo *A* informa nela que possui um *link* assíncrono para o nodo *B*. O nodo *B* ao receber a mensagem, verifica que *A* pode escutá-lo, logo marca em sua tabela que possui um *link* síncrono (bidirecional) com o nodo *A*. Na próxima

mensagem de HELLO enviada por *B* é incluído na mensagem a informação de que *B* possui um *link* bidirecional com *A*. Ao receber essa mensagem o nodo *A* reconhece que pode ser ouvido pelo nodo B, como também pode ouvi-lo, logo marca o *link* com *B* como sendo síncrono.

Na Figura 10 é apresentado o formato básico de uma mensagem HELLO. No cabeçalho da mensagem são encontrados os seguintes campos:

- *Reserved* - que recebe por padrão uma sequência de zeros "0000000000000";
- *Htime* - especifica o intervalo de tempo em que cada mensagem de HELLO é enviada;
- *Willingness* - registra o grau de disposição de cada nodo para repassar pacotes em nome de outros nodos. Esse campo recebe valores entre 0 e 7 significando respectivamente “nunca disponível” e “sempre disponível”. Na implementação do OLSR utilizada neste trabalho, em todos os nodos esse campo recebe o valor padrão “3”, significando “disposição normal” em repassar pacotes.

No cabeçalho das informações sobre cada vizinho a 1 salto, existem os seguintes campos:

- *Link Code* - que é subdividido em dois outros campos (Figura 11), recebendo duas informações básicas: o campo tipo do nodo vizinho (*neighbor type*) que pode receber os valores “síncrono”, “MPR”, ou “não é mais vizinho” e tipo de *link* (*Link type*), que pode receber os valores: “assíncrono”, “síncrono” ou “desconhecido”;
- *Neighbor interface address* - Campos que armazenam o endereço da interface de rede do nodo vizinho, se o nodo vizinho tiver mais que uma interface de rede os endereços são inseridos nesse campo.

Com base nas informações das mensagens de HELLO várias bases de informações são criadas, como: tabela de múltiplas interfaces, tabelas de *links* bidirecionais, tabelas de vizinhos a 1 e 2 saltos; conjunto de MPR e conjunto de MPR *Selectors.* Todas esses repositórios de informações serão detalhados na Seção 3.3.

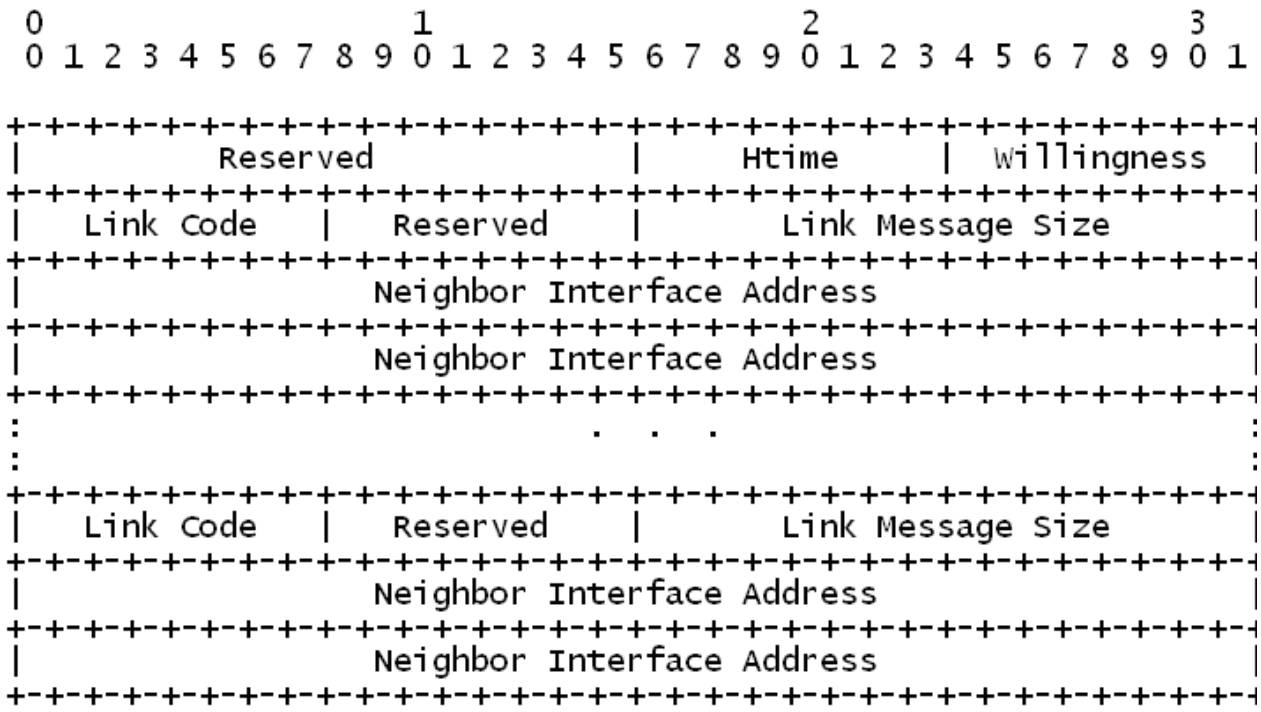

Figura 10 - Formato do cabeçalho de uma mensagem de HELLO – Fonte: [JAC03]

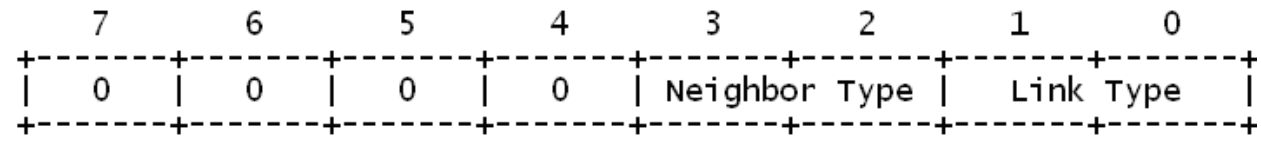

Figura 11 - Subdivisões do campo *Link Code* da mensagem de HELLO – Fonte: [JAC03]

3.2.2. Mensagens de Controle de Topologia (TC - *Topology Control*)

Para que cada nodo construa uma tabela com informações da topologia da rede, todo nodo que foi selecionado como MPR e apenas os nodos MPRs, devem enviar (por *broadcast*), periodicamente mensagens de controle de topologia (TC) para todos os nodos da rede, contendo informações dos *links* para todos os nodos de seu conjunto MPR *Selector*.

Na Figura 12 é apresentado o formato básico de uma mensagem TC a qual é divida nos seguintes campos:

- ANSN (*Advertised Neighbor Sequence Number*) – esse campo recebe um número de sequência relacionado ao conjunto de MPR *Selector* anunciado. Toda vez que ocorrer uma alteração no conjunto MPR *Selector* do nodo que gerou a mensagem TC esse campo é incrementado em 1. Esse número de sequência é utilizado para evitar *loops* infinitos de repasse de uma mensagem TC, e para indicar se a mensagem é a mais nova, se não for a mais nova ou já tiver sido processada a mensagem é descartada sem processamento algum;
- *Reserved* - que recebe por padrão uma sequência de zeros "0000000000000";
- *Advertised Neighbor Main Address* – esse campo recebe o endereço da interface de rede principal dos vizinhos anunciados, os quais fazem parte do conjunto MPR *Selector* daquele nodo [JAC03].

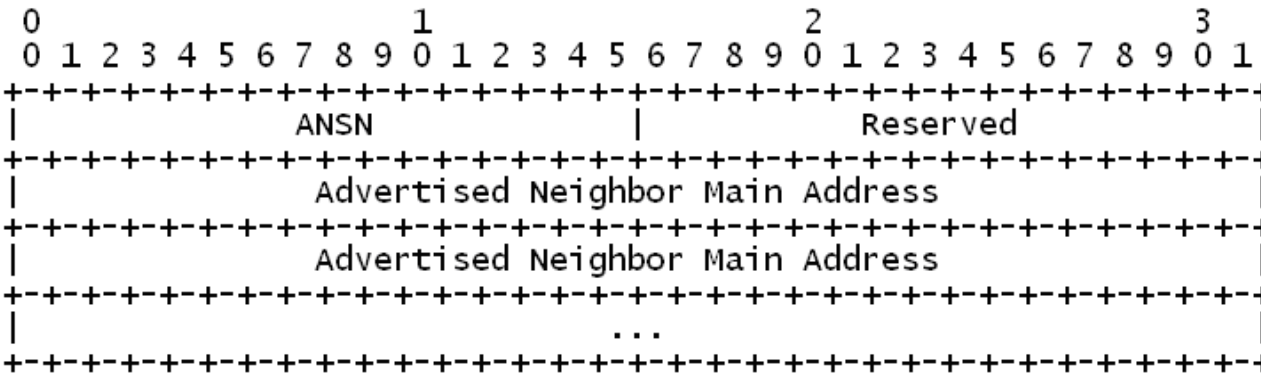

Figura 12 - Formato de mensagens de Controle de Topologia (TC) – Fonte: [JAC03]

As mensagens TC são enviadas em intervalos fixos, todavia se ocorrer alguma alteração no conjunto MPR *Selector* do nodo emissor da mensagem TC, a próxima mensagem é enviada antes do tempo programado. Todas as próximas mensagens TC serão enviadas no tempo padrão para envio de mensagem TC, até ocorrer outra mudança no conjunto MPR *Selector*. Na implementação do OLSR [ROS08], o intervalo de tempo padrão para envio de mensagens TC é de cinco segundos [JAC03].

Toda vez que uma mensagem TC é recebida, cada nodo da rede cria ou atualiza uma tabela de topologia (Subseção 3.3.7), onde são registradas as informações sobre a topologia da rede, as quais serão utilizadas para o cálculo da tabela de roteamento. Ao receber uma mensagem TC, cada nodo verifica se já existe alguma entrada com as

mesmas informações em sua tabela de topologia, caso exista a mensagem é descartada e não é repassada adiante; se não existir ou se o número de sequência da mensagem TC for mais recente, então as informações mais recentes são inseridas na tabela e a entrada antiga é removida. Após a mensagem é repassada adiante, se o nodo for MPR do remetente da mensagem TC.

3.2.3. Declaração de Múltiplas Interfaces (MID)

Em redes MANETs pode ocorrer que alguns nodos OLSR disponham de mais de uma interface de rede. Nesses casos os nodos que possuem múltiplas interfaces devem gerar periodicamente mensagens MID (*Multiple Interface Declaration*) que serão enviadas por *broadcast* para todos os nodos da rede utilizando o mecanismo do MPR. Na Figura 13 é apresentado o formato de uma mensagem MID [JAC03].

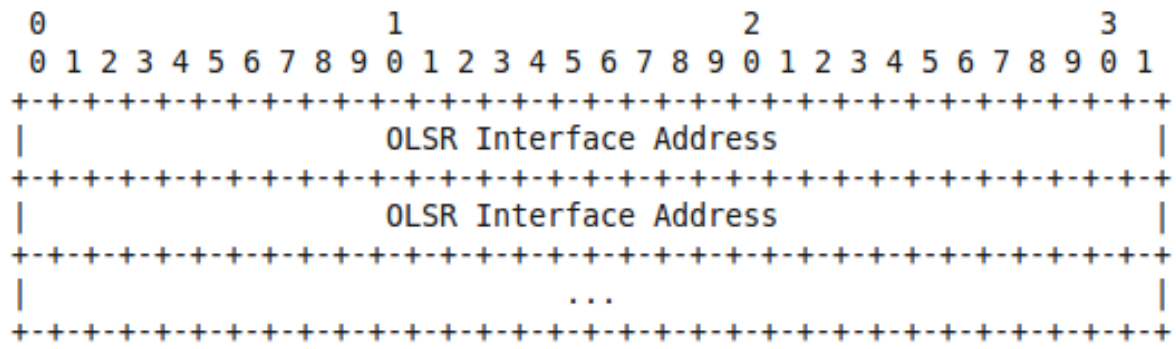

Figura 13 - Formato da mensagem “declaração de múltiplas interfaces - MID” – Fonte: [JAC03]

No OLSR cada nodo é identificado pelo endereço de rede principal. Em nodos com apenas uma interface de rede, o endereço principal é o endereço da única interface de rede. Já em redes com múltiplas interfaces é eleita uma interface como sendo a principal e as demais são relacionadas a essa interface.

O campo *OLSR Interface Address* da Figura 13 recebe todos os múltiplos endereços do nodo que criou a mensagem MID, excluindo o endereço da interface principal que já vem indicado no campo *Originator Address* do pacote básico do OLSR (Figura 7). As informações contidas nessa mensagem são guardadas em uma base de informações (Subseção 3.3.1) e utilizadas no cálculo da tabela de roteamento.

Neste trabalho mensagens MID não serão geradas, visto que os nodos não foram configurados para operar com múltiplas interfaces de redes, por não contribuir para os objetivos propostos nesse trabalho, todavia em trabalhos futuros pode-se avaliar o desempenho do OLSR_PD quando trabalhando com nodos de múltiplas interfaces.

3.3. Repositórios de informações do OLSR

Para que nodos OLSR construam e mantenham as tabelas de roteamento para todos os nodos da rede, as informações obtidas através da troca de mensagens HELLO e TC são armazenadas e organizadas em várias bases de informações, as quais serão descritas nas próximas subseções.

3.3.1. Base de informações das múltiplas interfaces de rede

Cada nodo OLSR guarda uma tabela onde são registradas as múltiplas interfaces de redes para cada destino da rede. Cada registro nessa tabela é chamado de "tupla de associação de interface" e é composto pelos seguintes campos [JAC03]:

l_iface_addr - l_main_addr - l_time

O campo *l_iface_addr* é um dos múltiplos endereços de interface de rede de um nodo; o *l_main_addr* é o endereço da principal interface de rede do nodo; e o *l_time* especifica a hora que esta tupla expira e deve ser removida. Em nodos com mais de uma interface de rede, a interface principal aparece em várias tuplas referenciando as placas adicionais.

3.3.2. Base de informações do conjunto de *links*

Em cada nodo OLSR deve ser mantida uma base de informações dos *links* de comunicação entre esse nodo e todos os seus vizinhos a 1 salto. O mecanismo utilizado para detectar os *links* entre nodos são as mensagens de HELLO. A cada recebimento de uma mensagem HELLO um nodo cria ou atualiza sua tabela de *links* contendo tuplas (registros) com os seguintes campos:

L_local_iface_addr - L_neighbor_iface_addr - L_SYM_time - L_ASYM_time - L_time

O campo *L_local_iface_addr* é o endereço da interface de rede local, ou seja, do próprio nodo que calcula o conjunto de *links*; o campo *L_neighbor_iface_addr* é o endereço da interface de rede do nodo vizinho; o campo *L _SYM_time* registra até quando (em redes reais, o tempo no formato hh:mm:ss) o *link* pode ser considerado simétrico; o campo *L_ASYM_time* registra até quando o *link* pode ser considerado assimétrico; por último, o campo *L_time* é o tempo de vida do registro na base de informações, que ao expirar indica que o registro deve ser removido [JAC03].

3.3.3. Base de informações de vizinhos a 1 salto

Após receber uma mensagem HELLO cada nodo deve processá-la e construir ou atualizar uma base de informações de seus vizinhos a 1 salto. Essas informações são armazenadas em tuplas contendo os seguintes campos:

N_neighbor_main_addr - N_status - N_willingness

O campo *N_neighbor_main_addr* é o endereço da principal interface de rede do vizinho; *N_status* guarda o estado do *link* com o vizinho, se “simétrico”, “assimétrico” ou “perdido”; o campo *N_willingness* recebe valores inteiros entre 0 e 7, e representa o grau de disposição do nodo vizinho em repassar mensagens em nome de outros nodos, significando respectivamente “nunca disponível para retransmitir” e “sempre disponível para retransmitir” [JAC03] em nome de outros nodos.

O campo *N_status* tem seu estado alterado com base nas informações dos campos *L_SYM_time* e L_ASYM_time da base de informações de *links* (Subseção 3.3.2). Se o tempo especificado em *L_SYM_time* ainda não expirou, *N_status* é declarado “simétrico”. Se o tempo especificado em *L_SYM_time* expirou, então o estado de *N_status* é declarado “assimétrico”. Se ambos *L_SYM_time* e *L_ASYM_time* são expirados, o *N_status* é declarado “perdido”, indicando que o *link* com o vizinho não existe mais [JAC03].

3.3.4. Base de informações de vizinhos a 2 saltos

Ao receber mensagens HELLO, cada nodo deve processá-la e construir ou atualizar sua tabela de vizinhos a 2 saltos. Essa tabela de vizinhos a 2 saltos é formada pelo conjunto de nodos que possuem *link* simétrico com um vizinho simétrico a 1 salto. Essas informações são guardadas em tuplas contendo os seguintes campos:

N_neighbor_main_addr - N_2hop_addr - N_time

O campo *N_neighbor_main_addr* é o endereço da principal interface de rede do vizinho a 1 salto que enviou a mensagem HELLO; O campo *N_2hop_addr* é o endereço da interface de rede principal de um vizinho a 2 saltos de distância, que possui um *link* simétrico com o *N_neighbor_main_addr;* e o campo *N_time* registra a hora em que essa tupla expira e deve ser removida [JAC03].

Para construir a tabela de vizinhos a 2 saltos, após receber uma mensagem HELLO as seguintes restrições são observadas:

- se já existe uma entrada (tupla) para um dado vizinho a 2 saltos, então é verificado se o tempo de validade da nova mensagem é maior que *N_time* então a tupla é atualizada, senão a nova entrada é ignorada;
- para inserir ou atualizar uma tupla é verificado se o endereço do remetente da mensagem consta como algum endereço principal do *L_neighbor_iface_addr* da base de informações do conjunto de *links* (Subseção 3.3.2) cujo tempo de validade do campo *L_SYM_time* não tenha expirado, ou seja, que exista ainda um *link* simétrico entre os dois nodos;
- se o vizinho a 2 saltos informado na mensagem HELLO é também um vizinho a 1 salto, essa entrada é ignorada, pois esse nodo não é um vizinho a dois saltos.

3.3.5. Conjunto de nodos MPR

Como apresentado na Seção 3.1, o conjunto MPR de um nodo *S* é denotado por MPR(*S*) e equivale ao menor número possível de nodos, capazes de alcançar todos os nodos a 2 saltos do nodo *S.*

Após executar o algoritmo que calcula o MPR, cada nodo constrói uma tabela contendo o seu conjunto de nodos MPR, ou seja, são escolhidos quais nodos que poderão repassar suas mensagens TC, os quais, consequentemente, pertencem ao grupo dos nodos que podem ser escolhidos como próximo salto na construção da tabela de roteamento desse nodo.

Após um nodo qualquer *S* calcular o seu conjunto de nodos MPR, inicialmente nenhum dos vizinhos de *S* possui a informação se pertence ao conjunto MPR*(S).* Para que os vizinhos de *S* adquiram essa informação, ao criar uma mensagem HELLO o nodo *S* insere no campo *neighbor type* (Figura 11) qual o tipo de *link* para um determinado vizinho *X*, que pode assumir três possíveis valores “síncrono”, “MPR” ou “não mais vizinho”. Cada nodo vizinho *X*, ao receber essa mensagem HELLO do nodo *S,* verifica qual o tipo de *link* informado no campo *neighbor type,* se este for “MPR” significa que o nodo *S* tem selecionado *X* como seu MPR, logo o nodo *X* insere o endereço do nodo *S* no conjunto de nodos que o selecionaram como MPR (Subseção 3.3.6), conjunto este posteriormente enviado por *X* nas suas mensagens TC.

3.3.6. Conjunto de nodos que selecionaram um nodo com seu MPR

Como cada nodo *S* precisa escolher o seu conjunto MPR(*S*)*,* também todo nodo *S* precisa guardar informações de quais nodos selecionaram *S* como seu MPR [JAC03]. Essas informações são guardadas em tabelas chamadas de conjunto de selecionadores de MPR (MPR *Selector*), cujas tuplas contêm os seguintes campos:

MS_main_addr - MS_time

Na tabela construída em *S* o campo *MS_main_addr* guarda o endereço da principal interface de rede do vizinho que selecionou *S* como seu MPR; já o campo *MS_time* equivale ao tempo de validade do *link* para o nodo que escolheu *S* como MPR. Esse campo recebe o tempo contido no campo *Vtime* do cabeçalho da mensagem HELLO (Figura 7). É importante destacar que cada nodo envia periodicamente suas tabelas MPR *Selector* para todos os nodos da rede através de suas mensagens de TC, de forma que cada nodo possa ter informações suficientes para construir sua tabela de topologia.

3.3.7. Base de informação da topologia da rede

Para possibilitar a construção ou atualização de uma base com informações da topologia da rede, cada nodo MPR envia mensagens TC (Subseção 3.2.2) contendo informações do conjunto de nodos que o selecionaram como seu MPR (Subseção 3.3.6). Ao receber e processar uma mensagem TC cada nodo forma sua tabela de topologia com a seguinte estrutura:

T_dest_addr – T_last_addr – T_time – T_seq

O campo *T_dest_addr* recebe o endereço de um destino que é um nodo que selecionou o originador da mensagem TC como MPR; *T_last_addr* é o endereço do nodo MPR que originou a mensagem TC ou ainda um nodo capaz de alcançar o destino; o *T_seq* é o número de sequência (campo *ANSN)* da mensagem TC originada de um dado nodo MPR; e o campo *T_time* registra o tempo de validade da tupla na tabela de topologia [JAC03].

Na tabela de topologia construída em cada nodo OLSR, não são registrados todos os *links* que compõem a rede, mas apenas aqueles que passam por nodos MPRs (Seção 3.1), isso porque o principal objetivo da adoção do MPR no OLSR é reduzir o número de retransmissões de mensagens de controle na rede. Segundo Jacquet et al. [JAC01], esse não é um problema para o OLSR, mas uma de suas vantagens, pois consegue reduzir o *overhead*, o tempo de processamento de mensagens de controle de topologia e ainda assim permite que todos os nodos da rede sejam alcançados toda vez que uma rota for requisitada. Provas do funcionamento do MPR são encontradas em [JAC01] [JAC02].

3.4. Cálculo da tabela de roteamento

Em redes executando o OLSR, cada nodo constrói e mantem uma tabela de roteamento que o permite enviar ou reencaminhar pacotes para todos os destinos conhecidos na rede. Para construir ou atualizar a tabela de roteamento, um nodo OLSR utiliza informações dos seguintes repositórios: conjunto de *links*, conjunto de vizinhos a 1 salto, conjunto de vizinhos a 2 saltos, tabela de topologia, e base de informações de múltiplas interfaces de rede (Seção 3.3). Caso ocorra alterações, como inserção ou exclusão de tuplas em qualquer desses repositórios, toda tabela de roteamento é

recalculada a fim de manter atualizadas rotas para todos os destinos da rede. O ato de recalcular a tabela de roteamento não cria ou dispara quaisquer mensagem adicional a ser transmitida pela rede ou na vizinhança, visto que são utilizadas apenas informações de tabelas locais. Cada entrada de rota é registrada na tabela de roteamento no seguinte formato [JAC03]:

1. *R_dest_addr - R_next_addr - R_dist - R_iface_addr*
2. *R_dest_addr - R_next_addr - R_dist - R_iface_addr*

Por fazer roteamento salto-a-salto, para cada destino da rede, o OLSR registra na tabela de rotas apenas o próximo salto na rota em direção ao destino. Em uma entrada de rota na tabela de roteamento o campo *R_dest_addr* equivale a um dos destinos da rede, o qual está a uma distância igual a *R_dist* (em número de salto) do nodo local; para cada destino *R_dest_addr* é guardada apenas uma entrada na tabela de roteamento, logo o número de entradas na tabela de roteamento será igual ao número total de nodos da rede menos um (o nodo local que executa o cálculo). O *R_next_addr* é o endereço da principal da interface de rede de um vizinho simétrico, que obrigatoriamente é um nodo MPR do nodo local, sendo portanto, o nodo escolhido como próximo salto na rota em direção ao destino *R_dest_addr.* Já *R_iface_addr* é o endereço da interface de rede do nodo local, pela qual é possível alcançar o nodo vizinho simétrico *R_next_addr.* O campo *R_iface_addr* é utilizado, porque um mesmo nodo pode possuir várias interfaces de rede, e é necessário saber por quais delas é possível alcançar o *R_next_addr*. Como neste trabalho todos os nodos foram configurados com apenas uma interface de rede, *R_iface_addr* sempre recebe o endereço da principal interface do nodo local [JAC03].

Segundo Jacquet [JAC03], para construir e manter uma tabela de roteamento em um nodo *X*, um algoritmo que encontra o “menor caminho” entre dois nodos é executado sobre um grafo direcionado contendo o arco *X->Y,* onde *Y* é qualquer vizinho simétrico de *X;* o arco *Y->Z,* onde *Y* é um nodo vizinho a 1 salto (Subseção 3.3.3), cujo campo “*willingness*” seja diferente de “nunca disponível”, e desde que, exista uma entrada na tabela de vizinhos a 2 saltos com *Y* como *N_neighbor_main_addr* e *Z* como *N_2hop_addr* (Subseção 3.3.4)*;* e o arco *U->V*, onde existe uma entrada na base de informações sobre a topologia da rede (Subseção 3.3.7) com *V* como *T_dest_add*r e *U* como *T_last_addr*. Tanto o algoritmo proposto em Jacquet et al. [JAC03] como o implementado na versão do OLSR para o NS-2 [ROS08] utilizam o algoritmo de “busca em largura” sobre um grafo, pois esse algoritmo descobre todos os vértices à distância *h*

a partir de *X,* antes de descobrir quaisquer vértices à distância *h*+1.

Em Jacquet et al. [JAC03], são encontrados os procedimentos para calcular ou recalcular as tabelas de roteamento em cada nodo executando o OLSR, os quais transcrevemos a seguir: (tradução nossa)

1 Todas as entradas da tabela de rotas são removidas.

2 As novas entradas na tabela de roteamento são registradas iniciando com vizinhos simétricos a 1 salto (h=1) como nodos destinos.

- Para cada registro na tabela de vizinhos a 1 salto, cujo *link* seja simétrico (*N_status = SYM*), e cujo tempo de validade do *link* seja maior que a hora atual (L_time >= hora atual) uma nova entrada de rota é registrada na tabela de roteamento, onde destino (*R_dest_addr*) e o endereço do próximo salto (R-next_addr)são ambos iniciados com o endereço do vizinho analisado e a distância (*R_dist*) é iniciada com 1 salto:

  R_dest_addr = *L_neighbor_iface_addr*, da tupla do *link* analisado;
  R_next_addr = *L_neighbor_iface_addr*, da tupla do *link* analisado;
  R_dist = 1;
  R_iface_addr = L_local_iface_addr da tupla do *link* analisado;

3 Para cada nodo a 2 saltos, que não seja também um nodo vizinho a 1 salto, ou o próprio nodo que executa o cálculo, e tal que exista pelo menos uma entrada na "tabela de vizinhos a 2 saltos" em que *N_neighbor_main_addr* corresponde a pelo menos um nodo vizinho a 1 salto cujo campo *willingness* seja diferente de "nunca disponível" então a tupla recebe:

  R_dest_addr = O endereço principal do vizinho a 2 saltos (*N_2hop_addr*);
  R_next_addr = O *R_next_addr* da entrada na tabela de roteamento em que:
  R_dest_addr == N_neighbor_main_addr da tupla de vizinho a 2 salto em análise;
  R_dist = 2;
  R_iface_addr = O R_iface_addr da entrada na tabela de roteamento em que:
  R_dest_addr == N_neighbor_main_addr da tupla de vizinho a 2 saltos;

4 As novas entradas de rotas para os nodos destinos a *h*+1 saltos são registrados na tabela de roteamento. O seguinte procedimento deve ser executado para cada valor de *h*, inciando com *h*=2 e aumentando ele em 1 a cada vez. A execução parará se nenhuma entrada é encontrada em uma iteração.

– Para cada entrada na TABELA DE TOPOLOGIA, se o seus T_DEST_ADDR não corresponde a R_DEST_ADDR de qualquer entrada de rotas na tabela de roteamento, E seu T_LAST_ADDR corresponde a R_DEST_ADDR de uma entrada de rota, cujo R_DIST seja igual a h (saltos), então uma nova entrada de rota deve ser registrada na tabela de roteamento (se já não existe) onde:

R_dest_addr = *T_dest_addr* da tupla da tabela de topologia em análise;

R_next_addr = *R_next_addr* da entrada de rota registrada onde: *R_dest_addr == T_last_addr*;

R_dist = h+1;

R_iface_addr = *R_iface_addr* da entrada de rota registrada onde: *R_dest_addr == T_last_addr*.

5 Em caso de haver múltiplas interfaces de rede, para cada entrada na base de múltiplas interfaces de rede(subseção 3.3.1), onde exista uma entrada de rota tal que *R_dest_addr == I_main_addr* (da base de múltiplas interfaces de rede) "e" não existir entrada de rota onde *R_dest_addr == I_iface_addr* então uma entrada de rota é criada na tabela de roteamento com:

R_dest_addr = *I_iface_addr* da entrada da base de múltiplas interfaces de rede;

R_next_addr = *R_next_addr* (da entrada de rota onde *R_dest_addr == I_main_addr*)

R_dist = *R_dist* (da entrada de rota onde *R_dest_addr == I_main_addr*)

R_iface_addr = *R_iface_addr* (da entrada de rota onde *R_dest_addr == I_main_addr*).

Como se observa no algoritmo anterior, por fazer busca em largura, o OLSR calcula rotas primeiro para destinos a 1 salto de distância (*h=1*), depois a 2 saltos (*h=2*) e então os destinos a *h+1* saltos, até que seja construída rota para o último destino mais distante da rede.

Tabela 2 - Tabela de roteamento do nodo 9 da rede com 10 nodos OLSR da Figura 14

R_dest_addr	*R_next_addr*	*R_iface_addr*	*R_dist*
0	**5**	9	4
1	**5**	9	3
2	**5**	9	3
3	**5**	9	2
4	**6**	9	2
5	**5**	9	1
6	**6**	9	1
7	**6**	9	2
8	***8***	***9***	***1***

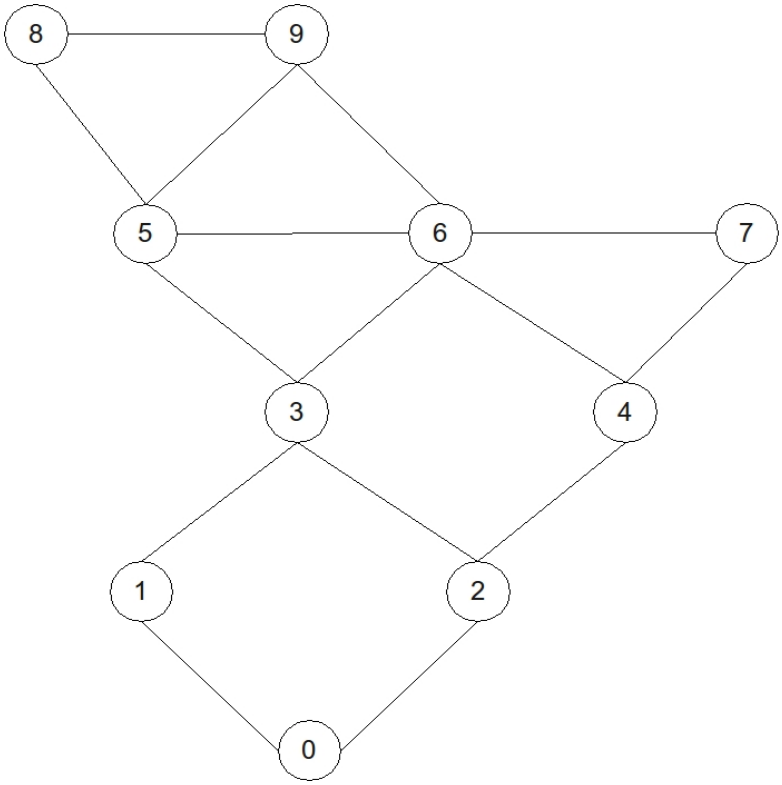

Figura 14 - Estrutura em grafo de uma rede com 10 nodos

Na Tabela 2, é apresentada, como exemplo, uma tabela de roteamento do nodo 9 calculada pelo OLSR para uma rede com 10 nodos contendo a estrutura apresentada na Figura 14. No Apêndice A é apresentado o algoritmo (pseudocódigo) e as estruturas de dados básicas utilizadas no cálculo da tabela de roteamento no OLSR.

3.5. Extensões do protocolo OLSR

Conforme análise feita em Abolhasan et al. [ABO04], o OLSR tem se destacado dentre vários protocolos de roteamento pró-ativos propostos, por manter o tempo todo uma rota para cada destino conhecido na rede e ainda reduzir significativamente o número de mensagens de controle na rede por utilizar apenas nodos selecionados como MPR para repassar mensagens de controle.

Mesmo apresentando tais vantagens, pesquisas têm constatado que o critério de seleção de rotas do OLSR, com base apenas no número de saltos pode não proporcionar as melhores rotas. Segundo Aslam et al. [ASL04] tais rotas podem não dar suporte a

qualidade de serviço (QoS) para satisfazer os requisitos de aplicações como transmissão de voz e vídeo, visto que nem sempre o menor caminho atende a restrições quanto a taxa de perdas de pacotes, atrasos, *jitter* (variação do atraso) e banda mínima por se apresentarem instáveis.

Em Leguay et al. [LEG06] foram analisadas várias extensões do protocolo OLSR que visam prover qualidade de serviço às rotas selecionadas. Dentre as propostas analisadas, está a apresentada em Lamont et al. [LAM03], que propõe que o conjunto MPR de um nodo seja formado pelos nodos que possuam *links* de maior largura de banda, isto porque no critério padrão do OLSR nodos de *links* de baixa largura de banda ou que apresente alto congestionamento podem ser selecionados. Segundo Leguay et al. [LEG06], essa proposta apresenta as vantagens de manter compatibilidade com a versão padrão do OLSR e não gerar nenhuma mensagem adicional na rede, todavia apresenta como desvantagem a utilização de apenas um critério. Em Badis et al. [BAD04] é apresentado o QOLSR (QoS OLSR) o qual busca encontrar MPRs que maximizem a banda disponível e minimizem o atraso fim-a-fim entre vizinhos a 2 saltos. Para tanto são acrescentados alguns campos nas mensagens de controle do OLSR padrão. As mensagens HELLO são adaptadas para suportar informações de QoS, visto que deverá carregar informações como largura de banda e atraso de cada vizinho a 1 salto. As mensagens de controle de topologia TC que são difundidas para todos os nodos da rede, também carregarão informações sobre a largura de banda e atraso dos *links* do nodo que a enviou. As principais vantagens dessa abordagem é que pode utilizar várias métricas de QoS e realmente garante rotas com maior comprimento de banda e menor atraso fim-a-fim. Todavia aumenta o tamanho do cabeçalho das mensagens TC e é incompatível com a versão padrão do OLSR.

Em Couto et al. [COU03] é apresentada uma nova métrica para seleção de rotas o ETX (*Expected Transmission Count*), a qual busca rotas de maior largura de banda por encontrar rotas com o menor *número esperado de transmissões* requeridas para que um pacote possa ser entregue e seu recebimento possa ser confirmado pelo destino final, mesmo que essa rota apresente um número maior de saltos. Caso haja mais de uma rota com o mesmo ETX, então é escolhida aquela que possuir menor número de saltos. Esta abordagem leva em consideração o fato de que em MANETs várias rotas podem ter a mesma quantidade de saltos, entretanto com diferentes qualidades dos *links,* podendo ocorrer que rotas com maior número de saltos apresentem maior qualidade. Uma extensão do OLSR tem sido proposta pelo projeto OLSR.ORG utilizando essa métrica

[COR07] [COU03]. Segundo Cordeiro et al. [COR07], de acordo com estudo comparativo feitos por Passos et al. [PAS06] o protocolo OLSR com a extensão ETX em redes móveis *ad hoc* sem fio tende a causar instabilidade de rotas e altas taxas de perda de pacotes. Isso ocorre porque em algumas situações a métrica ETX pode dar uma falsa visão a respeito do estado da rede.

Em Cordeiro et al. [COR07], é proposto OLSR-LD (OLSR *Link Delay*), o qual apresenta uma extensão para a escolha de rotas no protocolo OLSR baseado na métrica de retardo de transmissão entre dois nodos, em adição às métricas de menor perda e número mínimo de saltos. Esse protocolo, segundo o autor, une características do OLSR a uma técnica de medição de capacidade de enlace chamada *adHoc Probe*, a qual foi proposta por Chen et al. [CHE05]. O algoritmo do *AdHoc Probe* mede o atraso de um enlace apenas em um sentido, para tanto, pares de pacotes de tamanho fixo devem ser enviados em direção única, com o tempo de envio registrado em cada pacote transmitido. Ao receber esse pacote de controle, o destino pode então calcular o atraso de via única OWD (*One Way Delay*) e a capacidade do enlace em um sentido.

Segundo Cordeiro et al. [COR07], ao se calcular o OWD mínimo é possível determinar, também, o retardo da transmissão. Como o *AdHoc Probe* fornece informações do atraso apenas em um sentido, a mensagem HELLO foi alterada para que informações do retardo possam ser obtidas em ambos os sentidos do enlace, para tanto, duas mensagens de controle HELLO são enviadas ao mesmo tempo, com o mesmo tamanho e em tempos regulares. Caso o tamanho fixo da mensagem não seja alcançado, esse pacote é preenchido com *bits* extras para que todos as mensagens HELLO do OLSR-LD tenham o mesmo tamanho. A mensagem TC também foi alterada, a fim de carregar informações do retardo de transmissão entre o nodo emissor e todos os seus MPR *Selectors.* A ideia é que rotas com menor retardo na transmissão possam ser escolhidas reduzindo o atraso e a perda de pacotes. Como se observa, porém, esse protocolo introduz uma grande carga de roteamento na rede, pois duplica o número de mensagens HELLO, além de inserir *bits* adicionais para que o tamanho fixo possa ser alcançado.

4. PROTOCOLO DE ROTEAMENTO PROPOSTO – OLSR_PD

Neste capítulo é descrito o novo critério de seleção de rotas para o OLSR. Neste critério, em cada entrada na tabela de roteamento, o próximo nodo (próximo salto) em direção a um destino é selecionado se ele faz parte da menor rota de maior grau de conectividade, o que difere do OLSR original que usa apenas o tamanho do caminho (caminho mais curto em saltos) como critério. À versão do OLSR executando esse critério nomeamos como OLSR_PD (OLSR *based in Path Degree*) ou OLSR baseado no grau de conectividade do caminho.

Os criadores do OLSR na RFC 3626 (*Request for Comments*) [JAC03] afirmam que, em situações onde exista vários caminhos diferentes, de mesmo comprimento, até um dado destino, vários nodos vizinhos a 1 salto podem ser selecionados como próximo salto. Esses autores sugerem que, nessas situações, seja possível selecionar como próximo salto aqueles nodos MPR que possuírem maior grau de disposição em repassar (*willingness*) pacotes em nome de outros nodos. Considerando a situação acima, não propomos como critério de desempate a avaliação do grau de disposição em cooperar dos nodos MPR, mas avaliamos se o nodo MPR, vizinho a 1 salto, faz parte do menor caminho de maior grau de conectividade.

O restante deste capítulo está organizado da seguinte forma: na Seção 4.1 a proposta de implementação do protocolo é descrita; na Seção 4.2 são apresentadas as mudanças efetuadas no OLSR original para viabilização do novo critério de seleção de rotas; na Seção 4.3 é apresentado o algoritmo de seleção de rotas do protocolo proposto; e por último, na Seção 4.4, é apresentado um exemplo de construção de tabela de roteamento utilizando o OLSR_PD.

4.1. Descrição do critério proposto

O OLSR_PD não é um novo protocolo de roteamento, mas uma extensão do OLSR que segue os princípios básicos de funcionamento desse protocolo (Capítulo 3), utilizando as mesmas mensagens, estruturas de dados, e comportamentos nele implementados. Por herdar tais características do OLSR, o OLSR_PD é, também, um protocolo de roteamento dinâmico, por fazer roteamento salto-a-salto, e robusto, pois se adapta bem às falhas se

recuperando delas sem que seja necessário reinicializar todo serviço e que todos os pacotes tenham de ser reenviados toda vez que ocorrerem falhas.

Tanto no OLSR como no OLSR_PD são mantidas entradas na tabela de roteamento para cada destino da rede. Por fazerem roteamento salto-a-salto, uma entrada na tabela de roteamento não registra toda rota até ao destino, mas apenas o endereço do próximo salto para alcançá-lo. Em redes MANETs, em situações de alta densidade de nodos na rede, ocorre de vários nodos poderem ser utilizados como próximo salto na rota, pois fazem parte de rotas que possuem o mesmo número de saltos e levam aos mesmos destinos. Para esta situação, propomos, neste trabalho, um novo critério de seleção de rotas ao OLSR, o qual visa tomar como próximo salto na rota, aquele nodo que fizer parte da “menor rota de maior grau de conectividade”.

O principal propósito em adotar esse critério é que, caso ocorram falhas na rota em uso e *links* dessa rota se tornem indisponíveis, o nodo imediatamente anterior a falha, sendo um nodo da rota de maior grau, deve ter maior probabilidade, se comparado a nodos de outras rotas, de calcular, mais rapidamente, uma rota alternativa para o destino. Paralelamente a isso, tendo detectado a falha, o nodo deve enviar uma mensagem TC aos outros nodos, com informações das alterações na topologia da rede. Nodos anteriores a este, na mesma rota de dados, receberão a mensagem TC e, por fazerem parte da rota de maior grau, têm maior chance de calcular uma rota alternativa para o destino, se comparados a nodos de outras rotas. Assim, utilizando este critério, espera-se que os nodos da rota escolhida tenham maior probabilidade de encontrar rotas alternativas para alcançar um dado destinatário de dados, de tal forma que o descarte de pacotes, devido à mudança topológica durante o tráfego de dados, seja minimizado. Além da redução no descarte de pacotes, espera-se reduzir ainda o atraso fim-a-fim e a variação do atraso (*jitter*), com pouco ou insignificante aumente do *overhead* na rede, quando comparado à versão original do OLSR.

São encontrados na literatura alguns algoritmos roteamento que usam critérios semelhantes ao proposto nesse trabalho. Em Shoroeder [SCH07] é proposto um protocolo de roteamento reativo, dinâmico e tolerante a falhas (Subseção 2.3.2), em que cada nodo, sempre que precisa enviar ou encaminhar algum pacote de dados, executa o algoritmo do fluxo máximo para avaliar qual de seus vizinhos a 1 salto possui maior número de caminhos redundantes em direção ao destino. Aquele que possuir o maior fluxo máximo, também é o nodo que possui maior quantidade de caminhos disjuntos até o

destino e, por consequência maior é o número de atalhos que podem ser utilizados em caso de falha. Segundo Shoroeder [SCH07], uma característica interessante desse protocolo é que, nem sempre o caminho mais robusto, ou de maior grau de redundância, será o menor caminho, podendo ocorrer de destinos que estão a dois ou três saltos de distância precisarem do dobro ou mais saltos para que sejam alcançados.

É importante destacar as diferenças básicas entre o algoritmo proposto neste trabalho e o de Shoroeder [SCH07]. Primeiro que o protocolo apresentado aqui é pró-ativo e o outro reativo; segundo que, em Shoroeder [SCH07] dado um par de vértices (origem, destino), para cada nodo vizinho do nodo origem é avaliada todas as arestas a fim de identificar quantas delas levam ao destino, obtendo-se assim, a aresta que a partir dela, possui o maior número de caminhos de *links* disjuntos até o destino, mesmo que esse caminho possua o dobro ou mais saltos que o menor caminho para esse destino. No OLSR_PD é verificado, para cada destino, qual o menor caminho que apresenta maior grau de conectividade, o qual é dado pela soma do grau de conectividade de todos os nodos de uma rota.

Para implementar esse critério no OLSR_PD, é preciso que cada nodo conheça o grau de conectividade de todos os nodos MPR da rede, pois somente nodos MPR podem ser utilizados na formação de rotas em redes com nodos OLSR. Para calcular o seu grau de conectividade, cada nodo MPR *S*, antes de enviar uma mensagem TC, conta o número de vizinhos contidos na tabela de vizinhos a 1 salto, com os quais possua um *link* de comunicação simétrico. Então, cada nodo *S* insere a informação do seu grau de conectividade em todas as suas mensagens TC, que são periodicamente enviadas para todos os nodos da rede. Ao receber uma mensagem TC cada nodo cria ou atualiza uma tabela de topologia, armazenando informações dos *links* entre o nodo remetente *S* e seus MPR *Selectors* e também o grau do nodo *S,* informações essas, essenciais para o cálculo do grau de conectividade das possíveis rotas para cada destino da rede. Esses conceitos serão melhores detalhados nas próximas seções.

4.2. Alterações feitas no OLSR

Como visto na Seção 3.2, cada nodo ao receber as mensagens de HELLO toma conhecimento de todos os *links* de comunicação dele com todos os seus vizinhos a até 2 saltos de distância, dando origem a três bases de informações: base de informações do

conjunto de *links* (Subseção 3.3.2); base de informações sobre vizinhos a 1 salto (Subseção 3.3.3); e base de informações sobre vizinhos a 2 saltos (Subseção 3.3.4). Ao receber mensagens TC, enviadas por todos os nodos MPR da rede, cada nodo recebe informações dos *links* dos nodos MPRs com seus MPR *Selectors* (nodos que selecionaram o nodo que enviou a mensagem TC como seu MPR) (Seção 3.1), dando origem a uma tabela chamada “base de informações da topologia da rede” ou “tabela de topologia”.

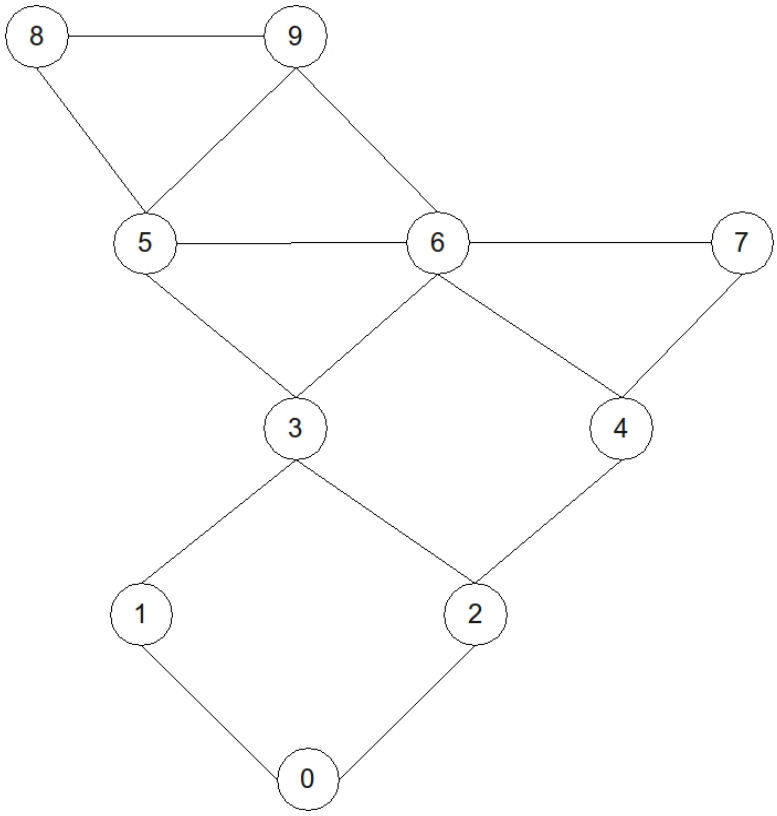

Figura 15 - Grafo de uma rede com 10 nodos - formação da tabela de topologia

Nas tabelas de topologias são registradas apenas informações dos nodos MPR e dos respectivos MPR *Selectors.* Isso significa que essa tabela de topologia não possui a representação completa de todos os *links* de comunicação da rede, mas apenas informações essenciais para que cada nodo crie e mantenha rotas para todos os destinos da rede. Tendo em vista que o critério de seleção de rotas proposto neste trabalho, o menor caminho de maior grau de conectividade, baseia-se nas informações sobre a quantidade de *links* (arcos) que incidem sobre um nodo (vértice), verifica-se que as informações contidas na tabela de topologia do OLSR são insuficientes para

implementação de tal critério. Esse fato pode ser melhor compreendido analisando-se os dados da tabela de topologia do nodo 9 (Tabela 3), da rede representada no grafo da Figura 15.

Tabela 3 - Tabela de topologia do nodo 9 da rede da Figura 15 aos 20 segundos simulação

T_dest_addr	*T_last_addr*	*T_seq*	*T_time*
5	6	4	33,663160
7	6	4	33,663160
9	6	4	33,663160
3	6	4	33,663160
4	6	4	33,663160
8	5	2	33,915676
6	5	2	33,915676
3	5	2	33,915676
5	3	3	33,447526
2	3	3	33,447526
6	3	3	33,447526
1	3	3	33,447526
3	1	1	33,915676
0	1	1	33,915676
4	2	1	30,081618
0	2	1	30,081618
2	0	0	30,081618
7	4	3	34,356037
2	4	3	34,356037

Essa tabela de topologia foi extraída do arquivo de registro da simulação (*traces*) com 10 nodos executando o OLSR (Figura 15), no tempo de 20 segundos de simulação. Ela é composta por 4 colunas (veja Subseção 3.3.7), sendo que a coluna *T_dest_addr* recebe o endereço da interface de um nodo MPR *Selector;* a coluna *T_last_addr* recebe o principal endereço do nodo MPR selecionado; a coluna *T_seq* recebe o número de sequência contido na mensagem TC; e o *T_time* registra o tempo de validade da tupla na tabela de topologia.

Conforme se observa na Tabela 3 a partir da primeira linha, primeira coluna, os nodos *5, 7, 9, 3* e *4* fazem parte do conjunto de MPR *Selectors* do nodo *6*, e que os nodos *8, 6, 3* fazem parte do conjunto de MPR *Selectors* do nodo *5* e assim sucessivamente. Com base nessas duas colunas podemos obter, parcialmente, o número de arestas incidentes sobre um vértice. Parcialmente porque, nem todos os vizinhos de um nodo o selecionam como seu MPR. Para o nodo *6* identificamos na tabela de topologia o conjunto de 5 enlaces *E= {(6,3), (6,4), (6,5), (6,7), (6,9)}* o que coincide exatamente com o grau de

conectividade desse nodo *d(6)=5,* conforme se observa na Figura 15. Já para o nodo 5 é encontrado apenas 3 arestas, a saber *E={(5,3), (5,6), (5,8)}* o que, todavia, não representa o grau desse nodo, pois *d(5)*=4. Como se observa, apenas com base nas informações da tabela de topologia não é possível obter o grau dos nodos, a fim de calcular rotas com base no grau de conectividade dos nodos do caminho.

Visto que, o OLSR não possui todas as informações necessárias para o cálculo do grau de conectividade dos nodos da rede, algumas alterações nas estruturas da mensagem TC e das tabelas de topologia e de roteamento se fizeram necessárias, como descrito na próximas subseções.

4.2.1. Alteração no cabeçalho da mensagem TC

Para que cada nodo tome conhecimento do grau de conectividade todos os nodos MPR da rede, visto que apenas as informações das estruturas de dados disponíveis no OLSR não são suficientes, foi acrescentado ao cabeçalho da mensagem TC um novo campo o qual chamamos de "*Grau_nodo_que_envia_TC"* (Figura 16 - seta em destaque), aumentando o cabeçalho dessa mensagem em dois 2 *bytes* (16 bits) que é suficiente para representar números entre 0 e 65535.

O grau de um nodo é calculado toda vez que um nodo MPR gerar uma mensagem TC contendo a sua tabela de MPR *Selectors* (3.2.2). Como visto na seção 2.5 o grau de um nodo (vértice) é dado pelo total de arestas (*links*) incidentes nesse nodo. No OLSR, essa informação pode ser obtida contando-se a quantidade de vizinhos simétricos que um dado nodo possui, com base em sua "tabela de vizinhos a 1 salto". Após contar o número de vizinhos, esse total é acrescentado ao campo "*Grau_nodo_que_envia_TC"* e então a mensagem é enviada para todos os nodos da rede, sendo repassados apenas pelos nodos MPR do nodo que cria ou que repassa essa mensagem.

Como dito por Jacquet et al. [JAC01] e Qayyum et al. [QAY02] "...em uma rede de nodos OLSR, rotas sempre serão uma sequência de saltos através de MPRs até um determinado destino". Por causa dessa característica do OLSR que faz com que, apenas os nodos MPR gerem ou repassem mensagens TC pela rede. A fim de manter a compatibilidade com o funcionamento básico desse protocolo, no OLSR_PD o cálculo do grau de conectividade é feito apenas por nodos MPR, visto que nodos não MPR, naquele

momento, nunca serão selecionados como próximo salto ou contados como nodos intermediários em uma rota.

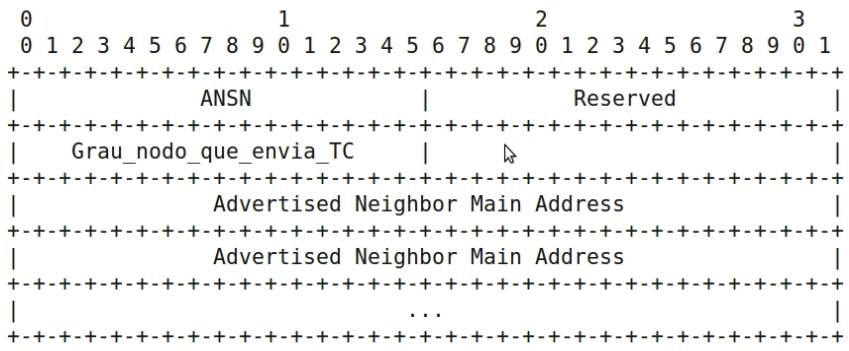

Figura 16 - Estrutura da mensagem TC do OLSR_PD com o campo que carrega o grau do emissor

4.2.2. Alterações na tabela de topologia

Para armazenar o grau de conectividade do nodo que envia a mensagem TC, foi acrescentado à tabela de topologia um campo chamado ***T_grau.*** **A seguir** é apresentado a estrutura da nova tabela de topologia:

*T_dest_addr – T_last_addr – T_seq – **T_grau** – T_time*

Na Tabela 4 é apresentada a tabela de topologia para o nodo 9 da rede da Figura 15 agora calculada pelo OLSR_PD. Na quarta coluna, "*T_grau",* é encontrado o grau de todos os nodos MPR listados na segunda coluna dessa tabela (*T_last_addr)*. Nessa rede apenas os nodos 7 e 8 não constam como MPR de nenhum nodo, por isso não têm os seus endereços listados na coluna 2 sendo, portanto, lhes atribuído grau de conectividade igual a 0, isso porque não são MPR. Isso não é um problema para a formação de rotas no OLSR_PD, visto que, no OLSR nodos não MPR nunca são utilizados como nodos intermediários ou como próximo salto na tabela de roteamento, mas apenas como nodo destino. A situação dos nodos 7 e 8 muda, se e somente se, ocorrer mudanças na topologia da rede e algum nodo elegê-los como seu MPR.

Tabela 4 - Nova tabela de topologia do nodo 9 da rede da Figura 15 calculada pelo OLSR_PD

OLSR				OLSR_PD				
T_dest_addr	*T_last_addr*	*T_seq*	*T_time*	*T_dest_addr*	*T_last_addr*	*T_seq*	*T_grau*	*T_time*
5	6	4	33,663160	5	6	4	**5**	33,684931
7	6	4	33,663160	7	6	4	**5**	33,684931
9	6	4	33,663160	9	6	4	**5**	33,684931
3	6	4	33,663160	3	6	4	**5**	33,684931
4	6	4	33,663160	4	6	4	**5**	33,684931
8	5	2	33,915676	8	5	2	**4**	33,853610
6	5	2	33,915676	6	5	2	**4**	33,853610
3	5	2	33,915676	3	5	2	**4**	33,853610
5	3	3	33,447526	5	3	3	**4**	33,488732
2	3	3	33,447526	2	3	3	**4**	33,488732
6	3	3	33,447526	6	3	3	**4**	33,488732
1	3	3	33,447526	1	3	3	**4**	33,488732
3	1	1	33,915676	3	1	1	**2**	34,094501
0	1	1	33,915676	0	1	1	**2**	34,094501
4	2	1	30,081618	4	2	1	**3**	34,842804
0	2	1	30,081618	0	2	1	**3**	34,842804
2	0	0	30,081618	2	0	0	**2**	34,842804
7	4	3	34,356037	7	4	3	**3**	34,094501
2	4	3	34,356037	2	4	3	**3**	34,094501

4.3. Cálculo da Tabela de roteamento no OLSR_PD

Tanto no OLSR quanto no OLSR_PD a tabela de roteamento é calculada toda vez que a rede é inicializada ou recalculada toda vez que ocorrem alterações na topologia da rede. A diferença básica é que, para suportar o cálculo do "grau do caminho" proposto neste trabalho, a tabela de roteamento do OLSR_PD recebe um campo chamado "*R_soma_grau*" responsável por guardar o grau de conectividade do caminho a ser percorrido a partir do próximo salto na rota "*R_last_addr*". A estrutura dessa tabela de roteamento é apresentada a seguir:

1. *R_dest_addr - R_next_addr - R_dist - R_iface_addr -* ***R_soma_grau***
2. *R_dest_addr - R_next_addr - R_dist - R_iface_addr -* ***R_soma_grau***

Cada entrada na tabela de roteamento de nodos OLSR_PD pode ser interpretada da seguinte forma: o grau do caminho "*R_soma_grau*" da menor rota de maior grau até o destino "*R_dest_addr"* está a uma distância *"R_dist"* do nodo local, cujo próximo salto é *"R_next_addr",* nodo este, que faz parte do menor caminho de maior grau de

conectividade e pode ser alcançado a partir da interface de rede *"R_iface_addr"* do nodo local.

Conforme proposto na RFC 3626 [JAC03] e implementado por Ros [ROS08], o cálculo da tabela de roteamento em nodos OLSR é feito executando-se um algoritmo do "caminho mais curto". Após encontrar o primeiro "caminho mais curto" o vizinho simétrico do nodo local, participante dessa rota, é eleito como o "próximo salto" (*R_next_addr*) em direção ao destino. No OLSR_PD, também é utilizado o algoritmo do "caminho mais curto", todavia, são avaliados todos os caminhos disponíveis de mesmo comprimento que levam ao destino, porém escolhido apenas aquele que possuir o "maior grau de conectividade". Após a escolha da rota, o vizinho simétrico do nodo local, participante dessa rota, é então selecionado como "próximo salto" para alcançar aquele destino.

O procedimento utilizado pelo OLSR para encontrar o "caminho mais curto" em uma rede, baseia-se em executar o algoritmo de "busca em largura" sobre o grafo da rede. Em Cormen et al. [COR02] encontramos a seguinte definição para o algoritmo de "busca em largura":

> Dado um grafo G=(V,E) e um vértice de origem distinta *s,* a busca em largura explora sistematicamente as arestas de *G* até "descobrir" cada vértice acessível a partir de *s.* O algoritmo calcula a distância (menor número de arestas) desde *s* até todos os vértices acessíveis desse tipo. Ele também produz uma "árvore primeiro na extensão" com raiz *s* que contém todos os vértices acessíveis. Para qualquer vértice *v* acessível a partir de *s,* o caminho na árvore primeiro na extensão de *s* até *v* corresponde a um "caminho mais curto" de *s* até *v em G,* ou seja, um caminho que contém o número mínimo de arestas. O algoritmo funciona sobre grafos orientados e também não orientados.
> A busca em largura recebe esse nome porque expande a fronteira entre vértices descobertos e não descobertos uniformemente ao longo. Isto é, o algoritmo descobre todos os vértices à distância *k* a partir de *s,* antes de descobrir quaisquer vértices à distância *k*+1. ([COR02] pg 422)

Para calcular a tabela de roteamento do OLSR_PD, o algoritmo de busca em largura é executado em cada nodo da rede, criando entradas de rotas para todos os destinos que estão a uma distância *h* do nodo *X,* ou seja, primeiro avalia-se todos os arcos *X->Y,* onde *Y* é qualquer vizinho simétrico de *X*. O grau do caminho para cada destinho *Y* é igual ao total de vizinhos de *Y* informado nas mensagens de HELLO e disponível na tabela de vizinhos a 2 saltos. Para cada destino a 2 saltos (*h=2*), representados pelo arco *Y->Z*, onde *Y* é um nodo vizinho a 1 salto, cujo campo "*willingness*" seja diferente de "nunca disponível" e desde que exista uma entrada na tabela de vizinhos a 2 saltos com *Y* como *N_neighbor_main_addr* e *Z* como *N_2hop_addr,* o grau do caminho até *Z* é dado pela soma do "*R_soma_grau*" do nodo *Y*

mais o grau do nodo destino *Z* contido na no campo *"T_grau"* da tabela de topologia. Em caso de haver mais de um caminho para o destino *Z,* sempre o que possuir o maior grau de conectividade será o escolhido. Para destinos a mais de 2 saltos (sempre incrementando a distância *h* em 1), representados pelo arco *U->V*, onde existe uma entrada na tabela de topologia da rede (Subseção 4.2.2) com *V* como *T_dest_add*r e *U* como *T_last_addr,* já o grau do caminho até *V* é dado pela soma do grau de *V,* encontrado no campo "*T_grau*" da tabela de topologia, mais o "*R_soma_grau*" do nodo *"R_dest_addr"* da tabela de roteamento que seja igual ao nodo *U.* Havendo mais de um caminho com a mesma distância até *U,* aquele que possuir maior grau de conectividade é o escolhido.

Todo o procedimento para calcular ou recalcular as tabelas de roteamento em nodos executando o OLSR_PD, com exceção do cálculo do grau de conectividade do caminho, segue basicamente os mesmos princípios de execução do OLSR. A seguir, é apresentado o procedimento utilizado para o cálculo da tabela de roteamento do OLSR_PD, no qual é dado destaque, em "negrito", para o cálculo do grau de conectividade da rotas da rede.

1 Todas as entradas da tabela de rotas são removidas.

2 As novas entradas na tabela de roteamento são registradas iniciando com vizinhos simétricos a 1 salto (h=1) como nodos destinos.

2.1 Para cada registro na tabela de vizinhos a 1 salto, cujo *link* seja simétrico e cujo tempo de validade do *link* seja maior que a hora atual (*L_time* >= hora atual) uma nova entrada de rota é registrada na tabela de roteamento, onde destino (*R_dest_addr*) e o endereço do próximo salto (*R_next_addr*)são ambos iniciados com o endereço do vizinho analisado, a distância (*R_dist*) é iniciada com 1 salto e o grau do caminho (*R_soma_grau*) é dado pelo total de vizinhos do nodo destino, como se segue:

R_dest_addr = *L_neighbor_iface_addr*, da tupla do *link* analisado;

R_next_addr = *L_neighbor_iface_addr*, da tupla do *link* analisado;

R_dist = 1;

R_iface_addr = *L_local_iface_addr* da tupla do *link* analisado;

R_soma_grau **= Total de vizinhos do nodo *L_neighbor_iface_addr* tal que*: L_neighbor_iface_addr == N_neighbor_main_addr* contidos na tabela de vizinhos a 2 salto;**

3 Para cada nodo a 2 saltos, que não seja também um nodo vizinho a 1 salto, ou o próprio nodo que executa o cálculo, e tal que exista pelo menos uma entrada na "tabela de vizinhos a 2 saltos" em que *N_neighbor_main_addr* corresponda a pelo menos um nodo vizinho a 1 salto cujo campo *willingness* seja diferente de "nunca disponível", então a tupla da tabela de roteamento recebe:

R_dest_addr = O endereço principal do vizinho a 2 saltos (*N_2hop_addr*);

R_next_addr = O *R_next_addr* da entrada na tabela de roteamento em que:

R_dest_addr == N_neighbor_main_addr da tupla de vizinho a 2 salto em análise;

R_dist = 2;

R_iface_addr = O R_iface_addr da entrada na tabela de roteamento em que:

R_dest_addr == N_neighbor_main_addr da tupla de vizinho a 2 saltos;

R_soma_grau **= O grau do nodo *N_2hop_addr* contido no campo *"T_grau"* da tabela de topologia, tal que *N_2hop_addr == T_last_addr*, mais o grau do caminho do nodo *R_next_addr* da entrada na tabela de roteamento em que:**

R_dest_addr == N_neighbor_main_addr **da tupla de vizinho a 2 salto em análise;**

3.1 Caso seja encontrado mais de um caminho em direção ao destino *R_dest_addr*, antes de guardar a nova entrada na tabela de roteamento, o *R_soma_grau* da nova entrada é comparado com o *R_soma_grau* da antiga entrada. Se a nova entrada possuir maior grau, a antiga é substituída, caso contrário, a antiga entrada é mantida na tabela e a nova descartada.

4 As novas entradas de rotas para os nodos destinos a *h*+1 saltos são registrados na tabela de roteamento. O seguinte procedimento deve ser executado para cada valor de *h*, inciando com *h*=2 e aumentando ele em 1 a cada vez. A execução parará se nenhuma entrada é encontrada em uma iteração.

4.1 Para cada entrada na tabela de topologia, se o seu *T_dest_addr* não corresponder a algum destino *R_dest_addr* da tabela de roteamento "e" o *T_last_addr* da tupla em análise, corresponder a *R_dest_addr* de alguma entrada de rota, cujo *R_dist* seja igual a "*h*", então uma nova entrada de rota deve ser criada, onde:

R_dest_addr = *T_dest_addr* da tupla da tabela de topologia em análise;

R_next_addr = *R_next_addr* da entrada de rota registrada onde: *R_dest_addr == T_last_addr*;

R_dist = h+1;

R_iface_addr = *R_iface_addr* da entrada de rota registrada

onde: *R_dest_addr == T_last_addr*.

R_soma_grau **= O grau de *T_dest_addr* contido no campo *"T_grau"* da tabela de topologia mais o grau do caminho para o nodo *R_dest_addr* da entrada na tabela de roteamento em que:**

R_dest_addr == T_last_addr **da tupla da tabela de topologia em análise;**

4.2 Caso o *T_dest_addr* corresponda a *R_dest_addr* de alguma entrada da tabela de roteamento(caso já exista rota para o destino) e a distância *R_dist* do nodo *R_dest_addr* seja diferente de *h+1*, então descarta-se essa nova entrada e passa-se a análise da próxima tupla da tabela de topologia.

4.3 Caso o *T_dest_addr* corresponda a *R_dest_addr* a alguma entrada da tabela de roteamento, cuja distância *R_dist* seja igual a *h+1*, então é verificado se o grau do caminho (*R_soma_grau)* da nova entrada é maior que o *R_soma_grau* da antiga. Se for maior a antiga é substituída, caso contrário, descarta-se a nova entrada de rota e a antiga é mantida.

5 Em caso de haver múltiplas interfaces de rede, para cada entrada na base de múltiplas interfaces de rede(Subseção 3.3.1), onde exista uma entrada de rota, tal que *R_dest_addr == I_main_addr* então uma nova entrada de rota é criada, onde:

R_dest_addr = *I_iface_addr* da entrada da base de múltiplas interfaces de rede;

R_next_addr = *R_next_addr* (da entrada de rota onde *R_dest_addr == I_main_addr*)

R_dist = *R_dist* (da entrada de rota onde *R_dest_addr == I_main_addr*)

R_iface_addr = *R_iface_addr* (da entrada de rota onde *R_dest_addr == I_main_addr*).

R_soma_grau **= O grau de *I_iface_addr* contido no campo *"T_grau"* da tabela de topologia, tal que *I_iface_addr == T_dest_addr,* mais o grau do caminho do nodo *R_dest_addr* da entrada na tabela de roteamento em que:**

R_dest_addr == T_last_addr **da tupla da tabela de topologia em análise;**

5.1 Caso o *I_iface_addr* corresponda a *R_dest_addr* de qualquer entrada de rotas na tabela de roteamento, ou seja, caso já exista rota para o destino esse destino da tabela de múltiplas interfaces, então o *R_soma_grau* dessa nova entrada é comparado com o *R_soma_grau* da antiga entrada. Se a nova entrada possuir maior grau, a antiga é substituída, caso contrário, a antiga entrada é mantida na tabela roteamento e a nova descartada.

No Apêndice B é apresentado o algoritmo (pseudocódigo) e as estruturas de dados básicas utilizadas no cálculo da tabela de roteamento no OLSR_PD.

4.4. Exemplo de execução do algoritmo proposto

Para demonstrar o funcionamento do algoritmo que implementa o critério de seleção de rotas proposto neste trabalho, foram executadas simulações no NS-2 (Network Simulator) [FAL09] com o OLSR e com o OLSR_PD. O cenário da rede *ad hoc* na simulação utilizada como exemplo possui 10 nodos, sem mobilidade e cuja topologia é representada na Figura 17. A finalidade dessa simulação é obter as bases de informações criadas em cada protocolo para ajudar na compreensão do critério proposto neste trabalho. Os resultados foram extraídos do arquivo de registro de simulação (*trace file*) no tempo de 20 segundos de simulação.

Com base na Figura 17, tomemos os procedimentos executados pelo nodo *9* para calcular sua tabela de roteamento, de forma que esta contenha entradas para todos os nodos destinos da rede, conforme Tabela 11. Primeiro são calculadas rotas para todos os vizinhos simétricos a 1 salto, que nessa figura são representados pelos vértices pintados de azul e alcançados pelas arestas *(9,5), (9,6), (9,8).* O cálculo de rotas para vizinhos a 1 salto é feito com base nas informações da tabela de vizinhos a 1 salto (Tabela 5) e da base de informações do conjunto de *links* (Tabela 6) do nodo que executa o cálculo, nesse caso o nodo 9. Essas duas tabelas são idênticas em ambos os protocolos, portanto são representadas juntas.

Para construir sua tabela de rotas para vizinhos a 1 salto, o nodo 9 executa o algoritmo de roteamento sobre sua tabela de vizinhos a 1 salto (Tabela 5). O primeiro vizinho a ser avaliado, nesse exemplo é o nodo 5. Então é verificado se o *status* do *link* de comunicação com esse nodo é simétrico, ou seja, se o campo *N_status==1.* Como o nodo 5 é vizinho simétrico do nodo 9, então é avaliado na tabela de *links* (Tabela 6) se o tempo de vida do *link (9,5)* é expirado. Como o valor, em segundos, do campo *L_time==30,488732 s* é maior que o tempo de simulação atual (20,00000 s), então uma nova entrada de rota é inserida na tabela de roteamento do nodo 9, onde: o campo destino dessa tupla *(R_dest_addr)* recebe o endereço do nodo 5 em análise, o campo de próximo salto em direção ao destino *(R_next_addr)* recebe o endereço do próprio nodo 5, visto que, o destino está a apenas 1 salto de distância *(R_dist),* portanto não necessitando de nodos intermediários, sendo o próprio destino o próximo salto na rota. Já o campo *R_iface_addr* recebe o endereço da interface de rede do nodo local que liga este nodo ao nodo próximo salto em direção ao destino *(R_next_addr)* que nesse caso é

também o endereço identificador do próprio nodo local 9. O campo grau do caminho *(R_soma_grau)* para do nodo destino 5 é obtido analisando-se quantas vezes o nodo 5 aparece no campo "*N_neighbor_main_addr"* da tabela de vizinhos a 2 saltos (Tabela 8) mais 1. Observe que na Tabela 8 constam os *links* partindo do nodo 5: *(5,6), (5,8), (5,3),* ou seja, apenas 3 *links.* Por não constar nessa tabela o *link* entre os nodos 5 com o nodo 9, que executa o cálculo, é sempre somado mais 1 ao total de *links* identificados, de forma que o grau do caminho do nodo 9 até o nodo 5 é igual a "4".

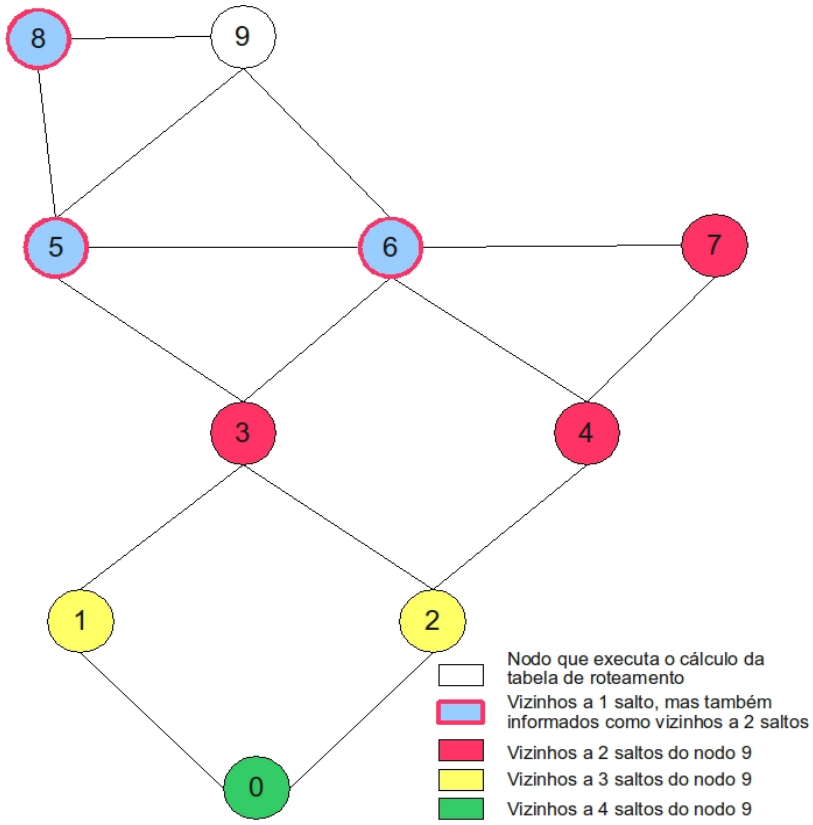

Figura 17 - Topologia da rede utilizada para demonstrar o cálculo da tabela de roteamento do nodo representado pelo vértice 9

Tabela 5 - Tabela de vizinhos a 1 salto do nodo 9 criadas tanto no OLSR quanto no OLSR_PD

OLSR e OLSR_PD		
N_neighbor_main_addr	*N_status*	*N_willingness*
5	1	3
6	1	3
8	1	3

Tabela 6 - Base de informações do conjunto de links do nodo 9

OLSR e OLSR_PD				
L_local_iface_addr	*L_neighbor_iface_addr*	*L_SYM_time*	*L_ASYM_time*	*L_time*
9	5	24,488732	24,488732	30,488732
9	6	25,994314	25,994314	31,994314
9	8	25,314634	25,314634	31,314634

Calculada entradas de rota para o nodo destino 5, então são avaliados todos os outros nodos vizinhos a 1 salto seguindo os mesmos critérios apresentados acima. Após encerrado o cálculo de rotas para os vizinhos a 1 salto a tabela de roteamento do nodo 9 estará como apresentado na Tabela 7.

Tabela 7 - Tabela de roteamento do nodo 9 após cálculo de rota para vizinhos a 1 salto

OLSR				OLSR_PD				
R_dest_addr	*R_next_addr*	*R_iface_addr*	*R_dist*	*R_dest_addr*	*R_next_addr*	*R_iface_addr*	*R_dist*	*R_soma_grau*
5	**5**	9	1	5	**5**	9	1	**4**
6	**6**	9	1	6	**6**	9	1	**5**
8	**8**	9	1	8	**8**	9	1	**2**

Para calcular as entradas de rotas para vizinhos a 2 saltos, o nodo 9 executa o algoritmo de roteamento sobre sua tabela de vizinhos a 2 saltos (Tabela 8). Na Figura 17, os nodos a 2 saltos do nodo 9, são representados pelos vértices pintados de vermelho. Analisando a Tabela 8, o primeiro vizinho a 2 saltos avaliado como destino, é o nodo 5. O primeiro passo é verificar se o nodo 5, informado nessa tabela como um vizinho a 2 saltos não é também um vizinho a 1 salto. Verifica-se que o nodo 5 é na verdade um vizinho a 1 salto, visto que, o nodo 5 aparece como vizinho a 2 saltos *(N_2hop_addr)* e também aparece como um vizinho simétrico a 1 salto *(N_neighbor_main_addr),* logo o cálculo de uma entrada de rota para o nodo 5 é cancelada e passa-se então para análise do próximo registro. É importante observar que na mesma situação do nodo 5 estão os nodos 6 e 8, pois são informados como vizinhos a 2 saltos, mas também são vizinhos a 1 salto, logo os mesmos procedimentos são executados para esses nodos. Na Figura 17, os vertices que representam os nodos 5, 6 e 8 são pintados de azul, por serem vizinhos a 1 salto, porém as bordas desses vértices são pintadas de vermelho, representando que esses nodos também são informados como sendo vizinhos a 2 saltos.

Após avaliado o nodo 5, na próxima iteração é analisado o nodo 3 como um destino a 2 saltos. A leitura dessa tupla, da Tabela 8, pode ser assim compreendida: que o vizinho

simétrico 6 informa que possui um *link* simétrico com o nodo 3, e que o tempo de vida dessa tupla expira no tempo de 24.493948 da simulação. Como exposto no parágrafo anterior, primeiro é verificado se o nodo 3 não é também um vizinho a 1 salto. Verifica-se que esse nodo não é vizinho a 1 salto, pois não existe entrada na tabela de vizinhos a 1 salto, cujo vizinho simétrico seja o nodo 3. Verificando ainda a entrada para o destino 3, é analisado se o nodo 6 *(N_neighbor_main_addr)* da tabela de vizinhos a 2 saltos (Tabela 8) é também um vizinho simétrico, *N_neighbor_main_addr,* contido na Tabela 5 e se este possui o grau de disposição em repassar mensagens em nome de outros *"willingness"* diferente de "nunca disponível", o que é positivo, visto que, o nodo 6 consta como um vizinho simétrico e possui *willingness==3,* o que significa que o nodo 6, além de alcançar o nodo 3, está "disponível" para repassar as mensagens deste nodo.

Encerrados os procedimentos de cálculo de rota para o destino 3, uma nova entrada de rota é inserida na tabela de roteamento do nodo 9, onde: o campo destino dessa tupla *"R_dest_addr"* recebe o endereço do nodo 3; o campo de próximo salto em direção ao destino "*R_next_addr"* recebe o endereço do nodo 6, pois é o nodo imediatamente anterior ao nodo 3 e por estar a 1 salto do nodo origem; o campo "*R_dist*" que recebe o número de saltos para alcançar o nodo 3 que é igual a "2"; o campo *R_iface_addr* recebe o endereço da interface de rede "9" que também é o identificador do nodo que executa o cálculo; e por último o valor do campo "*R_soma_grau",* que registra o grau do caminho até o destino 3, que é igual a *"9".* Esse valor é obtido analisando se existe alguma entrada na tabela de topologia do nodo 9 (Tabela 10), cujo o campo *T_last_addr* seja igual ao nodo destino 3 em análise. Na 9º tupla da Tabela 10 é encontrado o nodo 3 no campo *T_last_addr,* então é tomado o valor de *"T_grau"* dessa tupla, que é igual a *"4"*, mais o grau do caminho do nodo destino *"R_dest_addr"* da tabela de roteamento (Tabela 7) que seja igual ao nodo *N_neighbor_main_addr* da tupla de vizinho a 2 saltos em análise, nesse caso do nodo 6. O grau do caminho para o destino 6 na tabela de roteamento é igual a *"5",* logo o grau do caminho até o nodo destino em análise "3" é igual a *"4+5=9".*

Nas próximas iterações na tabela de vizinhos a 2 saltos, são analisados e igualmente calculadas rotas para os destinos 7, 6, 8 e 3. Ao encontrar novamente o nodo 3, subentende-se que a partir do nodo 9 existe mais de uma rota para alcançar o nodo 3, sendo o primeiro caminho *(9-6-5)* e o segundo recém encontrado, *(9-5-3).* Após calculada o grau do caminho da nova rota até o destino 3, que é igual a *"8"* é verificado se este é maior que o grau do caminho da entrada anteriormente registrada para o nodo 3, que é

igual a *"9".* Como o grau do caminho da nova entrada é menor que a anteriormente armazenada, essa nova entrada é descartada e a antiga é mantida na tabela de roteamento. Os procedimentos para calcular rotas para vizinhos a 2 saltos são executados até que todas as tuplas da tabela de vizinhos a 2 saltos sejam analisadas. Encerrada essa segunda fase da construção da tabela de roteamento, essa conterá entradas de rota para vizinhos a 1 e 2 saltos, como representada na Tabela 9.

Tabela 8 - Tabela de vizinhos a 2 salto do nodo 9.

OLSR			OLSR_PD		
N_neighbor_main_addr	*N_2hop_addr*	*N_time*	*N_neighbor_main_addr*	*N_2hop_addr*	*N_time*
6	5	24,493948	6	5	25,994314
6	3	24,493948	6	3	25,994314
6	7	24,493948	6	7	25,994314
5	6	25,856694	5	6	24,488732
5	8	25,856694	5	8	24,488732
5	3	25,856694	5	3	24,488732
8	5	25,266669	8	5	25,314634
6	4	24,493948	6	4	25,994314

Tabela 9 - Tabela de roteamento do nodo 9 após cálculo de rota para vizinhos a 2 saltos

OLSR				OLSR_PD				
R_dest_addr	*R_next_addr*	*R_iface_addr*	*R_dist*	*R_dest_addr*	*R_next_addr*	*R_iface_addr*	*R_dist*	*R_soma_grau*
3	**5**	9	2	3	**6**	9	2	**9**
4	**6**	9	2	4	**6**	9	2	**8**
5	**5**	9	1	5	**5**	9	1	**4**
6	**6**	9	1	6	**6**	9	1	**5**
7	**6**	9	2	7	**6**	9	2	**5**
8	**8**	9	1	8	**8**	9	1	**2**

Para calcular as entradas de rotas para vizinhos que estão a mais de 2 saltos do nodo 9 são utilizadas apenas informações das tabelas de topologia e de roteamento construídas até este momento. Na primeira iteração pela tabela de topologia é criada a variável "*h"* que registra a distância a partir do nodo 9. Essa variável é inicializada com *h=2* saltos e será incrementada em 1 (*h+1*) a cada iteração sobre a tabela de topologia, ou seja, após analisadas todas as entradas dessa tabela em busca de nodos *T_dest_addr* que estejam a *h+1* saltos, e assim sucessivamente até que não sejam encontrados mais nodos a "*h==n"* saltos do nodo 9.

Tabela 10 - Tabela de topologia do nodo 9 da rede representada na Figura 17

OLSR				OLSR_PD				
T_dest_addr	*T_last_addr*	*T_seq*	*T_time*	*T_dest_addr*	*T_last_addr*	*T_seq*	*T_grau*	*T_time*
5	6	4	33,663160	5	6	4	**5**	33,684931
7	6	4	33,663160	7	6	4	**5**	33,684931
9	6	4	33,663160	9	6	4	**5**	33,684931
3	6	4	33,663160	3	6	4	**5**	33,684931
4	6	4	33,663160	4	6	4	**5**	33,684931
8	5	2	33,915676	8	5	2	**4**	33,853610
6	5	2	33,915676	6	5	2	**4**	33,853610
3	5	2	33,915676	3	5	2	**4**	33,853610
5	3	3	33,447526	5	3	3	**4**	33,488732
2	3	3	33,447526	2	3	3	**4**	33,488732
6	3	3	33,447526	6	3	3	**4**	33,488732
1	3	3	33,447526	1	3	3	**4**	33,488732
3	1	1	33,915676	3	1	1	**2**	34,094501
0	1	1	33,915676	0	1	1	**2**	34,094501
4	2	1	30,081618	4	2	1	**3**	34,842804
0	2	1	30,081618	0	2	1	**3**	34,842804
2	0	0	30,081618	2	0	0	**2**	34,842804
7	4	3	34,356037	7	4	3	**3**	34,094501
2	4	3	34,356037	2	4	3	**3**	34,094501

Para exemplificar a execução da quarta etapa do algoritmo descrito na Seção 4.3, passamos então a analisar a tabela de topologia do nodo 9 (Tabela 10 - OLSR_PD). Na primeira tupla dessa tabela é encontrado *T_dest_addr == 5* e *T_last_addr == 6* dentre outros campos. Ao ser executado para essa tupla, primeiro é verificado na tabela de roteamento, a que fora construída até este momento (Tabela 9 - OLSR_PD), se já existe alguma entrada de rota cujo *R_dest_addr* seja igual a *T_dest_addr==5* e se *T_last_addr==6* corresponde a algum *R_dest_addr.* Observa-se que já existe entrada de rota para o nodo 5, bem como também existe entrada de rota correspondente ao nodo 6, que é o nodo imediatamente anterior a 5 capaz de alcançá-lo, todavia, verifica-se que o nodo 5 não está a *h+1* saltos do nodo 9 (3 saltos nessa iteração), e que a distância do nodo 6 não é igual a *h* saltos, logo ignora-se essa entrada da tabela de topologia e passa-se para as próximas iterações. Nas 9 primeiras tuplas da Tabela 10 (coluna OLSR_PD) todos os nodos *T_dest_addr* correspondem a algum *R_dest_addr* da tabela de topologia, todavia esses destinos são reprovados, porque ou o *R_dist* do destino *R_dest_addr==T_dest_addr* não é igual *h+1* ou porque o *R_dist* do nodo *R_dest_addr==T_last_addr* não é igual a *h.* Como se observa na Tabela 9, os únicos nodos para os quais ainda não foram construídas entradas de rotas, são os nodos 0, 1 e 2, cujas distâncias, em relação ao nodo 9, são respectivamente 4, 3, 3 saltos.

Na iteração sobre a 10ª tupla da tabela de topologia temos *T_dest_addr==2* e *T_last_addr==3,* com base nos procedimentos do parágrafo anterior, observa-se na Tabela 9 que ainda não existe entrada de rotas para o destino 2 e que existe *R_dest_addr==T_last_addr,* ou seja, que já existe entrada de rota para o nodo 3 e que esse nodo está a *h==2* saltos do nodo origem, logo sabe-se que o nodo 2 está a *h==3* saltos do nodo origem. Então é criada uma nova entrada na tabela de roteamento onde: o destino *R_dest_addr* recebe 2*;* o próximo salto *R_next_addr* recebe 6 que é também o próximo salto para que o nodo 9 alcance o nodo 3*;* o endereço da interface de rede do nodo 9, capaz de alcançar o destino 3, *R_iface_addr* recebe 9, que é o próprio endereço identificador do nodo 9; o campo que registra a distância do destino 3 em relação ao nodo 9, *R_dist* recebe *h+1,* ou seja, 3; e por último o campo que registra o grau do caminho "*R_soma_grau"* que é obtido somando-se o grau do nodo destino 2, *d(2)=3,* mais o grau do caminho até o nodo 3, *d(9+6+3)=9,* que é o *T_last_addr==R_dest_addr* dessa tupla, logo o grau do caminho até o destino 2 passando por *T_last_addr* é igual a *d(9+6+3+2)=12.* Após inserir a entrada de rota para o destino 2, nas próximas iterações sobre a tabela de topologia, serão avaliados vários *T_dest_addr*, todavia, apenas entradas para 0 e 1 serão criadas, executando-se os mesmos procedimentos descritos até aqui.

Para exemplificar casos em que existem mais que uma rota para alcançar um mesmo destino a mais de 2 saltos do nodo origem, tomemos como exemplo a entrada para o nodo 2. Continuando as iterações sobre a tabela de topologia, ao alcançar a 17ª tupla, encontramos *T_dest_addr==2* e *T_last_addr==0.* Ao executar o algoritmo de seleção de rotas para vizinhos a *h+1* saltos do nodo origem, como descrito anteriormento, verifica-se que já existe entrada de rota cujo *R_dest_addr==T_dest_addr*, todavia não existe entrada de rota cujo *R_dest_addr==T_last_addr*, ou seja, não existe entrada de rotas para o nodo 0, então essa tupla é reprovada e não mais analisada. Na 19ª tupla, encontramos novamente *T_dest_addr==2* e *T_last_addr==4.* Após as mesmas análises, verifica-se que existem entradas de rotas tanto para os nodos 2 quanto para o nodo 4. A seguir é analisado então se o nodo 2 está a *h+1* saltos do nodo 9 e se o nodo 4 está a *h* saltos, o que é positivo, indicando a existência de uma nova rota em direção ao destino 2, logo deve-se criar uma entrada de rota temporária para o destino 2, onde: *R_dest_addr=2; R_last_addr=6; R_iface_addr=9; R_dist=3;* e *R_soma_grau=11*. Logo após, é comparado o grau do caminho da nova entrada, que é igual 8, com o grau do caminho da antiga entrada de rota, que é igual 12. Como o grau do caminho da nova

entrada é menor, está é descartada e antiga mantida. Nessa quarta etapa de execução desse algoritmo serão avaliados todos os destinos que estão a mais de 2 saltos do nodo origem, até que não seja mais encontrado nenhum destino a ser avaliado.

Como neste trabalho os nodos são configurados para trabalhar apenas com uma interface de rede, quando for executada a 5ª parte do algoritmo descrito na Seção 4.3, então a base de informações de múltiplas interfaces de redes (Subseção 3.3.2) será encontrada vazia, não sendo, portanto, executada nenhuma ação nessa etapa de execução do algoritmo.

Após executados os procedimentos descritos nesse capítulo, a tabela de roteamento do nodo 9 estará completa, como demonstrada na Tabela 11. Nessa tabela estão disponíveis entradas de rotas para todos os nodos da rede, com exceção do 9 que a calculou, as quais permitem que o nodo 9 envie ou repasse mensagens para quaisquer nodos da rede, escolhendo apenas o próximo salto em direção ao destino. Cada nodo ao receber uma mensagem a ser repassada, por possuir também uma tabela de rotas como a apresentada nesta seção, reencaminha os pacotes de dados que, de salto em salto, percorrendo sempre o caminho de maior grau de conectividade, alcança ao destino.

Tabela 11 - Tabela de roteamento do nodo 9 da rede representada na Figura 17 no tempo de 20 segundos de simulação

OLSR				OLSR_PD				
R_dest_addr	*R_next_addr*	*R_iface_addr*	*R_dist*	*R_dest_addr*	*R_next_addr*	*R_iface_addr*	*R_dist*	*R_soma_grau*
0	**5**	9	4	0	**6**	9	4	**14**
1	**5**	9	3	1	**6**	9	3	**11**
2	**5**	9	3	2	**6**	9	3	**12**
3	**5**	9	2	3	**6**	9	2	**9**
4	**6**	9	2	4	**6**	9	2	**8**
5	**5**	9	1	5	**5**	9	1	**4**
6	**6**	9	1	6	**6**	9	1	**5**
7	**6**	9	2	7	**6**	9	2	**5**
8	**8**	9	1	8	**8**	9	1	**2**

5. AVALIAÇÃO DO OLSR_PD

Neste trabalho a avaliação do desempenho do OLSR_PD em relação ao OLSR é feita com base no método de simulação, que é um dos principais métodos para avaliar o desempenho de protocolos de roteamento. Segundo Law e Kelton [LAW91], a utilização de simulação para avaliar desempenho de sistemas traz vantagens significativas para o pesquisador, tais como: possibilidade de criação e controle dos cenários experimentais, de maneira que não seria possível no sistema real; a possibilidade de simular longos períodos de execução de um sistema real em tempo reduzido; e em geral, é mais econômico que testar o sistema real, pois evita gastos com aquisição e manutenção de equipamentos.

Neste capítulo são apresentados os resultados da avaliação de desempenho dos protocolos OLSR e OLSR_PD, os quais são avaliados sob vários critérios. O restante desse capítulo está organizado como se segue: na Seção 5.1 são apresentadas as ferramentas utilizadas para simulação e extração dos resultados; na Seção 5.2 são apresentadas as métricas utilizadas para avaliar o desempenho desses protocolos; na Seção 5.3 são apresentadas as configurações utilizadas nas simulações executadas neste trabalho; e, por fim, na Seção 5.4, os resultados das simulações são apresentados e analisados sob diversas métricas e critérios.

5.1. Ambiente de simulação

Para implementar a proposta de alteração no OLSR, fora utilizado o simulador de redes NS-2 (*Network Simulator*) [FAL09] versão 2.31 que é um simulador de redes por eventos discretos direcionado para pesquisas em redes de computadores, criado basicamente utilizando duas linguagens de programação: o C++, que é uma linguagem compilada, com o qual é implementado o núcleo do NS-2; e o OTCL (*Object Tool Command Language*), que é uma linguagem interpretada, a qual é usado para escrever os *scripts* da simulação em si, ou seja, para construir a topologia das redes, os cenários de mobilidade, as configurações dos nodos e os protocolos de comunicação e de roteamento a serem utilizados.

O NS-2 dá suporte a simulações em redes cabeadas e *wireless* (tanto local como

via satélite) e implementa agentes que representam os nodos de uma rede, sendo cada agente composto por camadas, individualmente configuráveis, que simulam nodos de redes.

A distribuição padrão do NS-2 traz implementado os principais protocolos de roteamento em MANETs, como o DSDV, DSR, AODV e o TORA. Como se observa a versão padrão desse simulador não traz uma implementação do OLSR, por isso foi instalado uma versão do OLSR implementada para o NS-2 por Ros [ROS08] chamada UM-OLSR (*University of Murcia* - OLSR) que segue todas as recomendações da RFC 3626 [JAC03]. A partir da versão original do OLSR foram feitas alterações na mensagem de controle de topologia TC, na tabela de topologia e alterações no critério de seleção de rotas desse protocolo, dando origem assim à versão nomeada de OLSR_PD.

Na versão original do NS-2 encontramos algumas ferramentas para geração de modelos de mobilidade e padrões de tráfego para as simulações: O *Setdest* é o gerador de cenário e movimentação dos nodos com base no modelo de mobilidade *Random Waypoint*; e o *cbrgen.tcl* que é um *script* feito em TCL para geração de padrões de tráfego. Ambos os programas foram adotados nos experimentos com esses protocolos, pois geram arquivos de cenário de movimentação e de tráfego de dados na rede reconhecidos pelo NS-2 e por serem bastantes utilizados pela comunidade científica.

Para extrair os resultados apresentados neste trabalho, foi criado um pequeno *script* feito usando *Shell Script*[1] e a linguagem *AWK* para geração de relatórios com informações como taxa de entrega, pacotes repassados com sucesso e o número de descartes e seus motivos (Apêndice C). Para extração do atraso médio fim-a-fim e da variação do atraso (*Jitter)* foi utilizado o software *TraceGraph* [MAL03] que é uma ferramenta gráfica criada para auxiliar na análise dos resultados das simulações feitas no NS-2, gerando relatório e mais de 200 gráficos. As métricas de avaliação de desempenho citadas anteriormente, são descritas na próxima seção.

5.2. Métricas de avaliação de desempenho

Para medir o desempenho do OLSR_PD em relação ao OLSR, parâmetros como mobilidade e quantidade de nodos sofreram variações a fim de testar o desempenho do

1 Arquivo de texto que contem uma lista de comandos interpretados por um *Shell*, criado com a finalidade de automatizar tarefas repetitivas como criação de relatório e análise destes.

novo critério sob as mais diversas situações. A avaliação do OLSR_PD foi feita no sentido verificar tanto vantagens e desvantagens da adoção desse critério na seleção de rotas. As métricas de avaliação de desempenho utilizadas neste trabalho são as seguintes:

1. **Taxa de entrega de pacotes:** é dada pela razão entre o número de pacotes entregues ao destino e o número de pacotes gerados pela aplicação na fonte;

2. **Taxa de pacotes de dados repassados (*forwarding*):** É dado pela razão entre o total de pacotes repassados com sucesso pelo total de pacotes enviados. Pacotes repassados com sucesso, são pacotes que foram roteados por nodos intermediários em uma determinada rota, ou seja, pacotes que foram repassados pela camada de rede do receptor, por existir naquele momento uma rota em direção ao destino, e que chegaram no próximo salto em direção ao destino. Dependendo da quantidade de nodos intermediários entre origem e destino, um pacote pode ser repassados mais de uma vez.

3. **Atraso fim-a-fim**: Segundo Júnior [JUN08], atraso é a medida do tempo decorrido entre o envio de uma mensagem por um nodo e a recepção desta mensagem pelo nodo destino. Isto ocorre devido aos possíveis atrasos causados pela latência da descoberta de rotas, velocidade de transmissão do meio, congestionamentos, por causa do processamento ou espera na fila de um dispositivo intermediário.

4. **Variação do atraso (*Jitter*):** em redes de pacotes, *jitter* pode ser entendido como a variação no tempo e na sequência de entrega de pacotes, devido à variação do atraso [MAR08]. Segundo Mello [MEL01], *jitter* é uma distorção que acontece, por exemplo, quando fluxos de voz ou vídeo são transmitidos em uma rede e os pacotes não chegam ao seu destino dentro da ordem sucessiva ou em uma determinada cadência, ou seja, eles variam em termos de tempo de atraso. Essa distorção é particularmente prejudicial a tráfegos de dados multimídia, fazendo com que o sinal de áudio ou vídeo tenham uma qualidade distorcida ou fragmentada na recepção. Uma técnica comumente utilizada para minimizar a variação de atraso é a utilização de *buffer*, o qual vai armazenando os dados a medida que eles chegam e os encaminham para a aplicação a uma mesma cadência.

5. **Pacotes descartados:** Pacotes descartados representa o número de pacotes que foram enviados na rede, mas não foram recebidos pelo destino em um determinado período de tempo. Tais perdas podem ocorrer por causa de congestionamento, ou por problemas na camada física, tais como rompimento de cabos, interferência na

transmissão *wireless* ou na camada de redes como a ausência de rotas no momento do encaminhamento do pacote [JUN08] [MAR08]. Segundo Júnior [JUN08], a comunicação VoIP não é tolerante a perda de pacotes: perdas mínimas de 1%, por exemplo, podem degradar uma chamada, provocando erros audíveis.

6. **Total de pacotes de roteamento:** Pacotes de roteamento ou de controle são pacotes gerados pelo protocolo de roteamento a fim de descobrir manter rotas ou tabelas de roteamento. No caso do OLSR que é um protocolo pró-ativo baseado no estado do enlace, pacotes de roteamento são pacotes que carregam mensagens HELLO ou mensagem TC. Como um mesmo pacote OLSR pode carregar ambas as mensagens, HELLO e TC, as quais possuem tamanhos diferentes, neste trabalho essa métrica será dada em “total de *bytes*” de controle gerados na rede por simulação.

5.3. Cenários de avaliação

A fim de avaliar o desempenho do novo critério de seleção de rotas proposto neste trabalho, em relação ao OLSR, foram construídos cenários que permitam avaliar a taxa de entrega e repasse de pacotes com a variação da mobilidade e quantidade de nodos na rede, bem como os efeitos desse critério para outras métricas. Na Tabela 12 são apresentados os principais parâmetros utilizados para configurar as simulações.

Tabela 12 - Parâmetros de configuração usados nas simulações

Parâmetro	Valor	Descrição
Padrão utilizado	IEEE 802.11b	Padrão de comunicação *wireless* que permite uma taxa nominal de transmissão de 11 Mb/s.
Frequência	2.4 Ghz	Frequência utilizada por equipamentos de rede sem fio em redes *ad hoc* (frequência de 2,4 Ghz e padrão 802.11b o alcance do sinal pode chegar a 250 metros).
Número de nodos	25, 50, 100 e 200	Número de nodos em cada simulação.
Número de conexões	10, 15, 15 e 25	Número de conexões para cada quantidade de nodos.
Velocidade dos nodos	0-1,5, e 0-20m/s	Velocidades que podem variar entre o mínimo e o máximo em uma simulação.
Número de	10	Para cada quantidade de nodos/velocidade, foram

execuções por simulação		realizadas 10 execuções. Foram combinados 2 padrões de movimentação com cinco padrões de tráfego diferentes.
Protocolos de Roteamento	OLSR e OLSR_PD	Protocolos de roteamento para redes *wireless*, pró-ativos, que fazem roteamento salto a salto e se baseiam no estado do enlace, para reconstruir suas tabelas de roteamento.
Protocolo de tráfego de dados	Tráfego CBR	*Constant Bit Rate* ou taxa constante de *bits*
Área da simulação	750 X 750, 1000 x 1000, 1500 x 1500 e 2000 x 2000	Áreas dadas em metros quadrados definidas respectivamente para as quantidades de nodos 25, 50, 100 e 200 nodos.
Tempo de simulação	100 segundos	Tempo em segundos de simulação padrão para todos os cenários/mobilidade.
Modelo de Mobilidade	*Randon Waypoint*	Padrão de movimentação gerado pelo *Setdest*, um aplicativo que acompanha o NS-2 para geração de cenários e padrões de movimentação de pontos aleatórios. Um dos padrões de mobilidade mais utilizado em simulações com redes *ad hoc.*
Antena	*OmniAntenna*	Antena *Omni* direcional que permite uma cobertura de 360º.
Modelo de Propagação	*Shadowing*	Sombreamento – modelo de propagação que segundo o manual do *Network Simulator [FAL09]* é o mais indicado para ambientes *outdoor* .
Meio físico	(PHY) *WirelessPhy*	Meio físico de transmissão sem fio.
Taxa de Geração de Pacotes	4 pacotes/s	Como o trafego é CBR, são enviados a cada segundo 4 pacotes.
Protocolo de transporte	UDP	(*User Datagram Protocol*) Protocolo de transporte não orientado a conexão.
Tamanho dos Pacotes	1000 bytes	Tamanho dos pacotes definidos na configuração do padrão de tráfego.
Tempo de Pausa dos nodos	2 segundos	Tempo máximo de pausa de um nodo antes que ele inicie um nova movimentação como proposto pelo padrão de movimentação *Randon Waypoint.*
Capacidade da fila *(Buffer)*	50 pacotes	*Buffer* onde pacotes gerados aguardam pelo acesso ao meio para que possam ser enviados. Caso essa fila esteja cheia, pacotes mais antigos são descartados. Segundo [PER01] todos os pacotes, de dados e de roteamento, passam por esse *buffer* (fila)

		a qual implementa regras de prioridade para o envio dos pacotes. Pacotes de controle tem prioridade sobre os pacotes de dados.

Para geração dos cenários de movimentação, foi utilizado o *software Setdest*, que cria cenários baseado no modelo de mobilidade *Random Waypoint*, modelo proposto e utilizado pela primeira vez por Johnson e Maltz [JOH96]. Neste modelo cada nodo se move em direções aleatórias dentro da área de simulação. A adoção do *Random Waypoint* como padrão de mobilidade para as simulações, visa criar um ambiente em que a mobilidade dos nodos provoque rompimento de enlaces e consequentemente a reconfiguração da tabela de roteamento de cada nodo da rede. Isso possibilita avaliar a taxa de entrega de pacotes de cada protocolo em cenários de mobilidade imprevisível e falhas de rota.

Para geração do tráfego de dados nas simulações, foi utilizado o gerador *cbrgen.tcl*, o qual cria aleatoriamente conexões TCP (*Transport Control Protocol*) ou UDP (*User Datagram Protocol*) entre dois nodos. Nas simulações realizadas neste trabalho optou-se por utilizar o padrão de tráfego CBR (*Constant Bit Rate*) para operar em conjunto ao protocolo de transporte UDP, visto que o TCP emprega mecanismos de controle de congestionamento, o que não funciona com o serviço de tráfego CBR, pois este continua enviando pacotes sempre a uma mesma taxa, independente se há ou não congestionamento na rede.

O motivo da utilização do tráfego CBR é que, dado dois protocolos de roteamento, ambos executando o mesmo cenário de movimentação e tráfego CBR, a quantidade de pacotes de dados gerados pela aplicação é igual nos dois protocolos, o que permite avaliá-los em condições de igualdade, pois ambos são executados sob uma mesma carga de trabalho.

Nas simulações aqui apresentadas os pacotes de dados tiveram seu tamanho fixado em 1000 *bytes,* e foram gerados a uma taxa constante de 4 pacotes por segundos. A transmissão de dados de cada conexão é iniciada em tempos aleatórios dentro do intervalo de 0 a 30 segundos da simulação.

Como observado na Tabela 12, os principais parâmetros que sofreram variações nas simulações foram, quantidades de nodos, a mobilidade dos nodos e o número de conexões por simulação. Foram criados cenários com 25, 50, 100 e 200 nodos, sendo o

número de fontes de tráfego 10, 15, 15 e 25 e o tempo de simulação igual a 100 segundos para todos os cenários. Quanto à variação da mobilidade dos nodos, para cada quantidade de nodos, foram criados dois cenários de baixa mobilidade, onde a velocidade dos nodos variam entre 0 a 1,5 m/s, e dois cenários de alta mobilidade, em que a velocidade dos nodos variam entre 0 a 20 m/s. Para criar 10 diferentes cenários em cada simulação, considerando a relação quantidade de nodos/mobilidade, foram criados cinco padrões de tráfegos com origens e destinos diferentes, os quais combinados com os dois cenários de mobilidade por simulação, originam os dez diferentes cenários.

Ao variar a quantidade nodos nos ambientes de simulação, torna-se possível avaliar qual o nível de degradação no desempenho do OLSR e do OLSR_PD diante do aumento de nodos na rede. A variação no número de nodos neste trabalho não tem a finalidade de aumentar a densidade da rede, visto que à medida que o número de nodos cresce, cresce também a área na qual estão dispostos. A densidade da rede, neste trabalho, é definida como o número médio de vizinhos a 1 salto por nodo ([NGU06] e [HAE06]). Em Haerri [HAE06] a densidade de uma rede, que pode ser formalizada como: $Densidade = N.\pi.R^2/A^2$, onde N é o número de nodos da rede; a constante π é igual a $\pi=3,1416$; R equivale ao alcance do sensor do rádio transmissor dos nodos da rede; e A equivale à área da rede em metros quadrados. Para as simulações com 25, 50, 100 e 200 nodos as densidades são, respectivamente, 8,72; 9,81; 8,72; e 9,81 vizinhos por nodo. É importante observar que esse cálculo considera uma distribuição perfeita dos nodos, todavia não é o que ocorre nas simulações, pois a disposição dos nodos e o padrão de mobilidade pode fazer com que, nodos de uma mesma rede, tenham densidades reais bem diferentes, o que, todavia, não será levado em conta neste trabalho.

Como se observa, a densidade média dos nodos das redes com 25 e 100 nodos é de 8,72, já as redes com 50 e 200 nodos possuem densidade igual a 9,81. Verifica-se que há uma diferença de 1,09 na densidade entre essas redes. Essa variação não interfere nas análises propostas neste trabalho, haja vista que, a intenção é avaliar o OLSR e OLSR_PD quanto à variação da mobilidade e quantidade de nodos na rede e possibilitar mais atividade de roteamento, para os cenários simulados.

Quanto a variação do parâmetro mobilidade, a intenção é avaliar a taxa de entrega de pacotes do OLSR_PD em relação ao OLSR em situações de baixa e alta mobilidade. Isto porque, como ambos os protocolos são pró-ativos, em situações de mobilidade basta

uma mudança na vizinhança de um nodo, que será ativada a atualização das tabelas de rotas de todos os nodos da rede, provocando queda no desempenho desses protocolos.

Para compensar essas variações e dar maior credibilidade aos resultados, para cada quantidade de nodos foram realizadas 20 rodadas de execuções para cada quantidade de nodos, sendo 10 para os cenários de baixa mobilidade e 10 para os cenários de alta mobilidade. Também para todas as métricas avaliadas são calculados o intervalo de confiança (IC^2) de 95% relativos à média das amostras em cada simulação, que é um índice bastante utilizado para análise de resultados de simulações, os quais estão representados nos gráficos como uma barra de erros.

Conforme se observa na tabela Tabela 12 a área das simulações também variaram: para 25 nodos a área é de 750x750 metros; para 50 nodos 1000x1000; para 100 nodos 1500x1500; e para 2000 nodos 2000x2000. Essa variação visa criar uma distância suficiente para que haja vários nodos intermediários nas conexões e, consequentemente, aumento de atividades de roteamento durante as simulações, o que não aconteceria se a área tivesse tamanho fixo, visto que o alcance do sinal de rádio de nodos 802.11b operando a uma frequência de 2,4 GHz pode chegar a 250 metros.

5.4. Análise dos resultados

Nesta seção são apresentados os resultados das simulações realizadas neste trabalho. Essas simulações foram conduzidas com a finalidade de avaliar o comportamento e a eficiência do OLSR e do OLSR_PD sob um variado conjunto de situações. De uma forma geral, os parâmetros que variaram nos vários modelos de simulações são:

1. A mobilidade dos nodos, a fim de avaliar o comportamento desses protocolos em cenários de baixa mobilidade, que proporciona um ambiente mais estável com pouca quebra de *links* e menor atividade de cálculo e recálculo das tabelas de roteamento, e em ambientes de alta mobilidade, que cria um ambiente com muitas quebras de *links,* e de constante atividade de calcular e recalcular tabelas de

2 Cálculo do IC é feito utilizando a função INT.CONFIANÇA(alfa;DesvPad; tamanho) do *software Open Calc* do pacote *Open* Office. Os parâmetros passados para a função são os seguintes:

- **alfa -** é o coeficiente de confiança que é uma probabilidade variando entre 0 – 1;
- ***DesvPad*** - é o desvio padrão do intervalo das amostras;
- **tamanho -** equivale ao número de amostras analisadas

roteamento;

2. A quantidade de nodos, que ao ser aumentada, faz crescer o número de pacotes de controle na rede, o que faz aumentar a disputa pelo canal de comunicação entre pacotes de dados e pacotes de roteamento e o tempo para que cada nodo calcule ou recalcule suas tabelas de roteamento;

3. A área das simulações, que em conjunto à quantidade de nodos cria um ambiente com maior ou menor atividade de roteamento.

As métricas que serão observadas nos resultados das simulações são: a taxa de entrega de pacotes, a taxa e quantidade de pacotes reencaminhados com sucesso, o atraso fim-a-fim, a variação do atraso (*jitter*), carga de roteamento normalizada da quantidade de *bytes* e de pacotes de controle utilizadas na criação e manutenção das tabelas de roteamento. Também são identificadas as principais causas que levam às perdas de pacotes

5.4.1. Taxa de entrega de pacotes

Na Tabela 13 e Figura 18, são apresentadas as taxas de entrega de pacotes para os cenários de baixa mobilidade (0-1,5 m/s) e alta mobilidade (0-20 m/s). Observa-se, por esses dados que, para ambos os cenários de mobilidade e quantidade de nodos, o OLSR_PD apresenta ganho na taxa de entrega de pacotes, em relação ao OLSR original.

Tabela 13 - Taxa de entrega de pacotes vs número de nodos - Mobilidade entre 0-1,50 e 0-20 m/s

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	**OLSR_PD**	**OLSR**	**OLSR_PD**
25	47,28%	50,37%	69,87%	74,70%
50	31,50%	37,41%	43,47%	48,67%
100	15,48%	17,10%	18,86%	20,10%
200	5,57%	6,20%	5,40%	5,62%

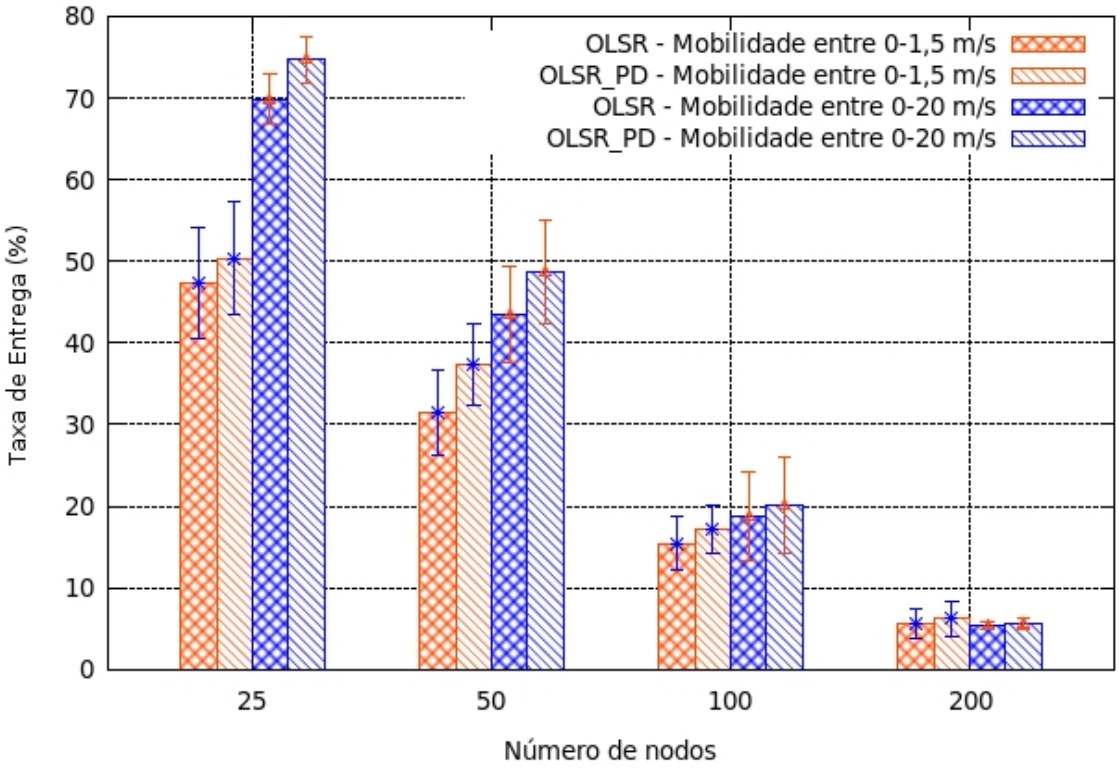

Figura 18 - Taxa de entrega de pacotes vs número de nodos - Mobilidade entre 0-1,50 e 0-20 m/s

Analisando os dados contidos na Tabela 13 e no gráfico da Figura 18, verifica-se que, à medida que cresce o número de nodos, a taxa de entrega de pacotes diminui rapidamente. Para ambientes de baixa mobilidade, nas redes com 25 nodos, a taxa de entrega de pacotes do OLSR e OLSR_PD atinge, respectivamente, 47,28% e 50,37% já para redes com 200 nodos, essa taxa fica em 5,57% para o OLSR e 6,20% para o OLSR_PD. Para ambiente de alta mobilidade, em redes com 25 nodos, o OLSR e OLSR_PD apresenta, respectivamente, a taxa de entrega de 69,87% e 74,70%, já para redes com 200 nodos, essa taxa fica em 5,40% para o OLSR e 5,62% para o OLSR_PD. Segundo Abolhasan et al. [ABO04], não ser bons em escalabilidade, é uma característica de protocolos pró-ativos, e mesmo o OLSR que, segundo esse mesmo autor, apresenta a melhor escalabilidade dentre os protocolos pró-ativos avaliados em seu trabalho, sofre tal degradação, visto que, cada nodo precisa manter, em todo tempo, rotas para todos os outros nodos da rede.

O aumento da rede, em protocolos pró-ativos, pode representar um aumento significativo no número de pacotes de roteamento circulando pela rede, os quais disputarão os mesmos canais de comunicação e a largura de banda com pacotes de

dados, provocando congestionamento e colisões que levam a descartes e perdas desses pacotes. Um dos principais motivos desse descarte é o tamanho da fila da camada MAC, onde os pacotes passam antes de terem acesso ao meio, que neste trabalho foi configurada com espaço para 50 pacotes apenas. Nessa fila os pacotes de roteamento têm prioridade em relação aos pacotes de dados, e como o número de pacotes de controle aumenta à medida que aumenta o número de nodos da rede, essa fila fica cada vez mais ocupada por pacotes de roteamento. Exemplo disso se observa na Tabela 22, conforme aumenta o número de nodos na rede ocorre um significativo aumento no número de descartes por todos os motivos, mas o descarte por falta de espaço na interface da fila (IFQ) da camada MAC tem o maior aumento, pois enquanto nos cenários com 25 nodos, para ambas as mobilidades, a média de descarte por IFQ, conforme Tabela 24, representa pouco mais de 5% dos descartes, para cenários com 200 nodos o número de pacotes descartados por IFQ, representa em torno de 25% a 29% dos motivos de descarte em ambos os protocolos. Essa situação é agravada porque o protocolo de transporte utilizado neste trabalho, o UDP, em conjunto ao protocolo gerador de tráfego CBR, não implementam controles de congestionamento, de forma que, mesmo que as filas nos roteadores estejam cheias, os nodos origens de tráfego continuam enviando pacotes de maneira constante. Segundo Tanembaun [TAN03], quando pacotes chegam a um roteador e a memória deste está cheia, esses pacotes podem ser descartados.

Outro aspecto a se analisar nesses resultados é que, manter a densidade da rede igual ou próxima, para os protocolos em estudo, não garante a mesma e nem compatíveis taxas de entrega, pois nesse caso, a taxa entrega é diretamente influenciada pelo aumento no número de nodos na rede. Esse é o caso por exemplo dos cenários com 25 e 100 nodos e 50 e 200 nodos que possuem densidade iguais, mas que por causa do aumento na quantidade de nodos na rede é registrado um significativo aumento na quantidade de pacotes de controle na rede, o que influência, diretamente, para a queda na taxa de entrega de pacotes.

Outra situação que se observa no gráfico da Figura 18, é que as taxas de entrega foram bastante baixas, mesmo para a quantidade mínima de nodos, 25 nesse caso, a taxa de entrega, para ambos os protocolos, ficou entre 47% a 74% e para a quantidade máxima de nodos, 200 nodos, a taxa registrada ficou em pouco mais de 5%. Dentre os principais fatores que determinaram o baixo desempenho, destacamos o tamanho das áreas determinadas para cada simulação, que combinado à mobilidade, faz com que os nodos se dispersem, ou mesmo se isolem, em determinados momentos, provocando

perdas de pacotes. As áreas para cenários com 25, 50, 100 e 200 nodos, propostos neste trabalho, possuem respectivamente as seguintes medidas: 750x750, 1000x1000, 1500x1500 e 2000x2000 metros. Essas áreas permitem manter a semelhança na densidade das redes simuladas, e foram assim criadas com o propósito de aumentar o comprimento das rotas, em número de saltos, à medida que se aumenta o número de nodos, a fim de possibilitar cenários com maior atividade de roteamento de pacotes. Todavia isso pode influenciar no desempenho do OLSR, pois segundo Jacquet et al. [JAC02] e Huhtonen [HUH04] esse protocolo, por suas características, apresenta melhor desempenho em redes densas, ou seja, redes onde os nodos estejam mais próximos uns dos outros.

Quanto a variação da mobilidade, se observa que a taxa de entrega, nos cenários de mobilidade entre 0 e 20 m/s, em ambos os protocolos, se mostrou superior às taxas alcançadas nos cenários de baixa mobilidade, para as simulações com 25, 50 e 100 nodos. A maior diferença é encontrada nas simulações com 25 nodos, pois enquanto no cenário de baixa mobilidade o OLSR apresenta taxa de 47,28% e o OLSR_PD 50,37%, no cenário de alta mobilidade o OLSR registra a taxa de 69,87% e o OLSR_PD 74,70%. Resultados obtidos em Gowrishankar et al. [GOW07] e Choi e Ko [CHO04] mostram que a tendência dos protocolos de roteamento é que, ao se aumentar a mobilidade dos nodos da rede, diminua a taxa de entrega, visto que mais pacotes de controle são gerados, aumentando a disputa pelos recursos da rede. Todavia, não foi o que ocorreu nas simulações para os cenários com 25, 50 e 100 nodos.

Essa diferença na taxa de entrega, descrita acima, pode ser compreendida analisando-se os dados da taxa de repasse de pacotes com sucesso (Tabela 15 e Figura 20) e comparando-os com as taxas de entrega (Tabela 13 e Figura 18), de forma que se torna possível inferir que, nas simulações de cenários de alta mobilidade, é maior o número de pares origem-destino alcançáveis com apenas 1 salto, ou seja, onde ocorrem comunicação direta. Considerando que tanto a taxa de entrega, quanto a taxa de repasses com sucesso são dadas pela relação entre taxa de entrega/taxa de repasse pelo número de pacotes enviados e, considerando ainda, que pacotes repassados com sucesso são eventos praticados apenas por nodos intermediários em rotas com múltiplos saltos, para estes resultados em específico, ao extrairmos a diferença da Taxa de Entrega de Pacotes (TEP) pela Taxa de Repasse com Sucesso (TRS), obtemos, no mínimo, a taxa de entrega de pacotes que foram feitas por rotas com apenas 1 salto. No mínimo porque um pacote pode ser repassado várias vezes em uma rota, o que, todavia, não

será considerado nessa análise.

A situação anterior pode ser ilustrada tomando-se os resultados obtidos, para as redes com 25 nodos em cenários de alta mobilidade (0-20 m/s) (Tabela 13). Primeiro subtrai a TEP pela TRS, que neste caso, para o OLSR TEP=69,87% e TRS= 29,73%, logo a diferença entre TEP-TRS=40,14% e para o OLSR_PD a TEP=74,70% e a TRS=38,62%, logo TEP-TRS=36,08%. Considerando que, para os cenários analisados, a mobilidade e a disposição topológica de cada nodo são as mesmas, mudando-se apenas o protocolo de roteamento, pode-se concluir que, no mínimo, 40,14% da TEP alcançada por esses protocolos foram registradas entre vizinhos a 1 salto. Para reforçar essa hipótese, vale destacar que a TRS alcançada nas redes com 25, 50, e 100 (Figura 20), em cenários de baixa mobilidade, foram maiores que as alcançadas nos cenários de alta mobilidade e que, mesmo assim, a taxa de entrega registrada foi menor nas três redes.

Analisando, ainda, os resultados do atraso médio fim-a-fim (Tabela 17) verifica-se, para as redes com 25, 50 e 100 nodos, que o atraso médio registrado nos cenários de alta mobilidade é bem inferior ao registrado em cenários de baixa mobilidade. Segundo Tanembaun [TAN03], dentre os fatores que influenciam o atraso médio na entrega de pacotes estão o comprimento do caminho e o nível de congestionamento. Visto que, nas redes de alta mobilidade, é registrada uma significativa redução no atraso, em ambos os protocolos, infere-se que a distância média, em número de saltos, entre os pares origem-destino são menores que a distância em cenários de alta mobilidade.

Na Tabela 14 e Figura 19, são apresentados os valores do número médio de nodos intermediário percorridos pelos pacotes de dados em cada simulação. Esses valores são obtidos pela razão entre o total de pacotes recebidos pela soma do número de nodos intermediários por onde esses pacotes passaram. Como se observa, nos cenários de alta mobilidade, tanto o OLSR quanto o OLSR_PD, apresentam redução no número médio de nodos intermediários na rota dos pacotes de dados, pois enquanto que no cenário de baixa mobilidade o OLSR e OLSR_PD apresentam, respectivamente, o número médio de nodos intermediários de 0,88380119 e 0,93204027 nodos, no cenário de alta mobilidade esses valores reduzem para, respectivamente, 0,56391286 e 0,65006015 nodos. Esses valores comprovam o que se infere nos parágrafos anteriores, pois mostram que o aumento na taxa de entrega de pacotes, em ambos os protocolos, em cenários de alta mobilidade, deve-se ao fato de que a maioria das trocas de pacotes foram feitas através de comunicação direta com os vizinhos a 1 salto. A troca de pacotes por comunicação

direta com os vizinhos a 1 salto evita descartes de pacotes por evitar congestionamentos em roteadores intermediários, além de proporcionar significativa redução na média do atraso fim-a-fim, como se observa na Tabela 17 e Figura 21.

Tabela 14 - Número médio de nodos intermediários percorridos pelos pacotes de dados recebidos, dado pela razão entre o total de pacotes de dados recebidos pela soma do número de nodos intermediários por onde os pacotes passaram

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	OLSR_PD	OLSR	OLSR_PD
25	0,88380119	0,93204027	0,56391286	0,65006015
50	0,88380119	0,93204027	0,56391286	0,65006015
100	1,21611263	1,34886166	0,86213789	0,96898740
200	0,99480194	1,06887167	0,65497256	0,62118351

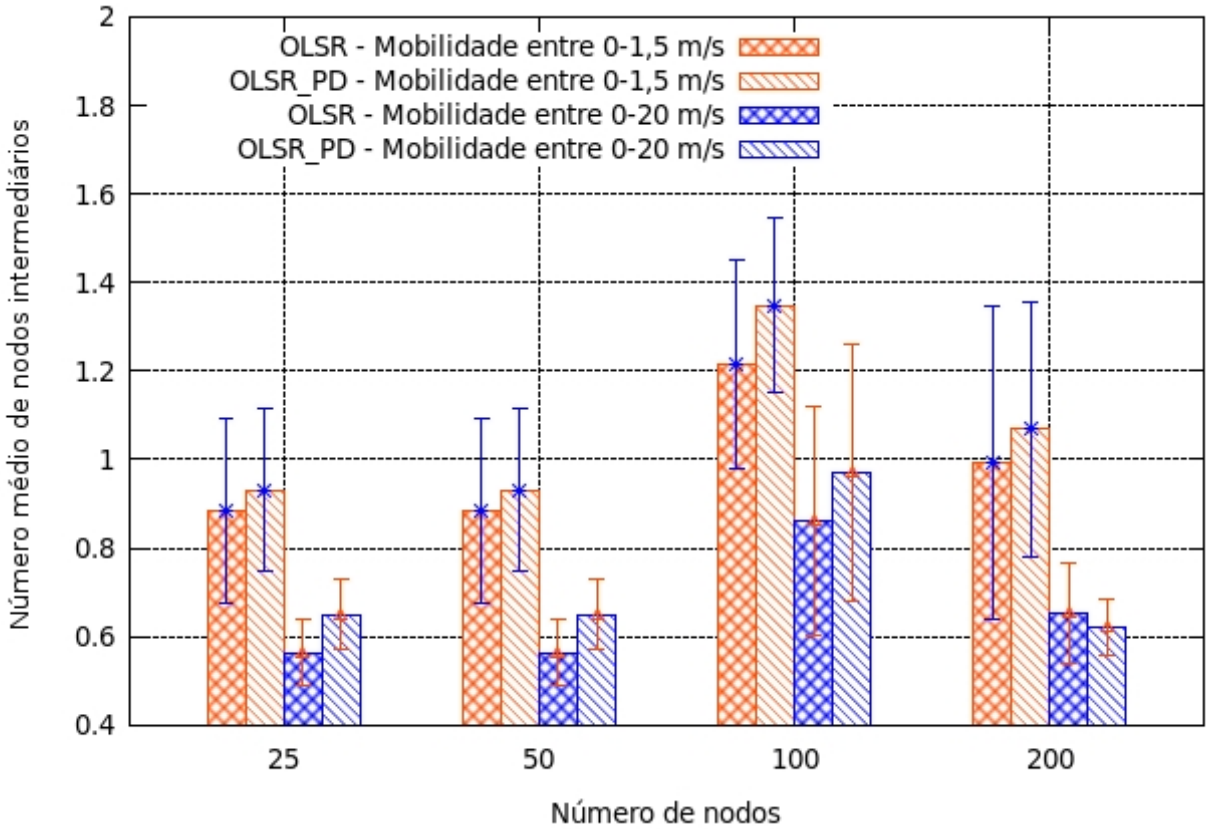

Figura 19: Número médio de nodos intermediários percorridos pelos pacotes de dados recebidos, dado pela razão entre o total de pacotes de dados recebidos pela soma do número de nodos intermediários por onde os pacotes passaram

Observando-se os valores da Tabela 14 e Figura 19, verifica-se que o OLSR_PD apresenta, em sete de oito cenários, um número médio de nodos intermediários maior que o OLSR. Isso significa que o OLSR_PD conseguiu entregar um maior número de pacotes, em relação ao OLSR, por caminhos com mais número de saltos, ou seja, significa que o OLSR_PD apresentou melhor desempenho que o OLSR na comunicação com múltiplos saltos, para os cenários propostos. Isso justifica o melhor desempenho do OLSR_PD, em relação ao OLSR, quanto à taxa de entrega de pacotes (Tabela 13 e Figura 18) e reforça a ideia, proposta neste trabalho, de que ao fazer roteamento por caminhos com maior grau de conectividade, aumenta-se a probabilidade de que, em caminhos com múltiplos saltos, os pacotes de dados sejam entregues ao destino.

5.4.2. Taxa de pacotes repassados com sucesso

Na Tabela 15e Figura 20 são apresentadas as taxas médias de pacotes repassados com sucesso, ou seja, a quantidade de vezes que a camada de redes repassou pacotes de dados e estes chegaram ao próximo nodo da rota, conforme determinado pelo nodo que repassou os pacotes. O número de repasses de pacotes com sucesso está diretamente relacionado à taxa de entrega de pacotes, visto que, demonstra a capacidade do protocolo de roteamento de, em ambientes de múltiplos saltos, encontrar as melhores rotas de forma que mais pacotes alcancem seu destino.

Os resultados apresentados na Tabela 15 e Figura 20 mostram que o OLSR_PD, quanto à taxa de repasse de pacotes, apresenta um desempenho superior à versão original do OLSR em todos os cenários de mobilidade e para todas as quantidades de nodos. Como se observa, por esses dados, ambos os protocolos apresentam suas melhores taxas de repasse no cenário de baixa mobilidade (0-1,5 m/s) e com o menor número de nodos na rede (25 nodos). Nesse cenário, o OLSR alcançou a taxa de 50,57%, enquanto o OLSR_PD alcançou 56,77%, ou seja, esse último apresenta um ganho médio de 6,2% no número de repasses com sucesso. Já a menor taxa média de repasse foi registrada no cenário de alta mobilidade (0-20 m/s) com 200 nodos na rede, onde a menor taxa do OLSR foi de 5,85% e a do OLSR_PD foi de 6,27%, ou seja, o OLSR_PD apresentou um ganho médio de 0,42% em repasses com sucesso.

Como se observa, aumentando-se o número de nodos na rede e a mobilidade dos nodos, a taxa média de repasse diminui. E isso porque, com o aumento do número de

nodos na rede, haverá mais nodos enviando pacotes de controle HELLO e TC na rede, além de que, cada pacote tem o seu tamanho aumentado, pois deve carregar informações de mais nodos. Já com o aumento da mobilidade ocorrem mais quebras de *links,* o que pode motivar perdas de pacotes já em trânsito e disparar a emissão de novos pacotes de controle, que devem ser enviados antes do tempo padrão. Com mais pacotes de roteamento na rede, recursos como largura de banda e espaços nas filas de saída da camada MAC, ficam saturados com pacotes de controle, visto que estes têm prioridade sobre os pacotes de dados, provocando mais descartes de pacotes.

Outro fator que pode ter determinado essa significativa redução na da taxa de repasse, nos cenários de alta mobilidade, é a presença de mais pares fontes-destinos de tráfegos alcançáveis com apenas um salto entre si, de forma que não se faz necessário o repasse dos pacotes, ou seja, apresentam comunicação direta. Isso é facilmente verificável, se considerarmos que tanto a taxa de entrega quanto a taxa de repasse com sucesso são obtidas pela divisão do total de pacotes entregues ou repassados pelo total de pacotes enviados, e que pacotes repassados com sucesso são eventos registrados apenas em nodos intermediários de rotas com múltiplos saltos. Logo ao extrair a diferença entre a taxa de recebidos e taxa de repassados com sucesso, é possível inferir que a diferença representa, no mínimo, a taxa de pacotes entregues por canais de comunicação direta, conforme apresentado na Subseção 5.4.1. Na Tabela 14 e Figura 19 são apresentados os números médios de nodos intermediários percorridas pelos pacotes de dados recebidos. O que se observa, nesses resultados, é que nos cenários de alta mobilidade a média de nodos intermediários fica bem abaixo que a média em cenários de baixa mobilidade, o que prova que grande parte dos pacotes de dados trocadas em ambientes de alta mobilidade foram realizadas entre vizinhos diretos, sem a necessidade de nodos roteadores intermediários.

Tabela 15 - Taxa de pacotes repassados com sucesso vs número de nodos - Mobilidade entre 0-1,50 e 0-20 m/s

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	**OLSR_PD**	**OLSR**	**OLSR_PD**
25	54,49%	59,23%	43,42%	50,79%
50	50,72%	61,68%	39,98%	52,55%
100	41,39%	45,34%	28,05%	32,27%
200	15,99%	17,86%	10,64%	11,20%

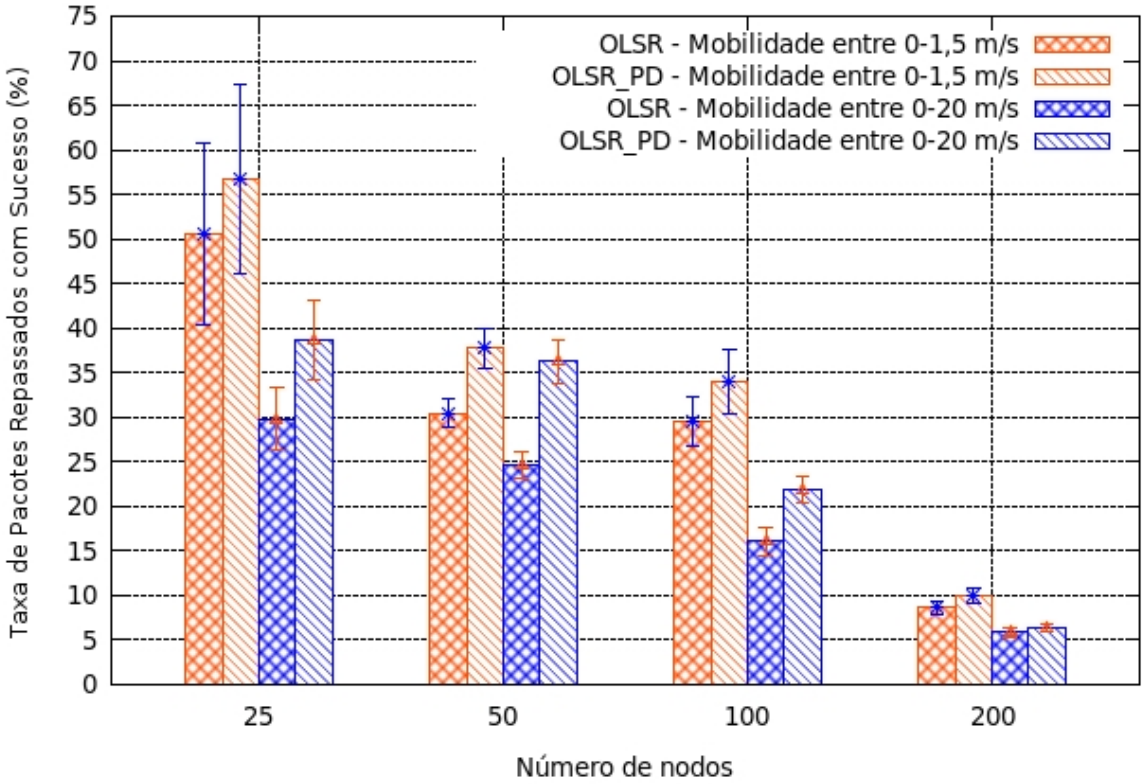

Figura 20 - Taxa de pacotes repassados com sucesso vs número de nodos – Mobilidade entre 0-1,5 e 0-20 m/s

Como se observa no gráfico da Figura 20, o OLSR_PD apresenta seus melhores resultados, em relação ao OLSR, nos cenários de alta mobilidade, principalmente com 25, 50 e 100 nodos. Ao extrair o percentual da diferença[3] entre o OLSR_PD e o OLSR essa diferença fica mais evidente, conforme Tabela 16. Por esses resultados, é possível verificar que nessas três redes, no cenário de alta mobilidade, o desempenho do OLSR_PD praticamente dobra, em relação ao desempenho em redes de baixa mobilidade. Apenas para rede com 200 nodos é que o desempenho do OLSR_PD cai, em relação ao cenário de baixa mobilidade.

Por apresentar suas melhores taxas de repasse nos cenários de maior mobilidade, é possível afirmar que, como proposto, as rotas escolhidas pelo OLSR_PD, aumenta a probabilidade de que em caso de falhas em *links*, o que é comum em ambientes de alta mobilidade, o nodo imediatamente anterior à falha, por fazer parte da rota de maior grau de conectividade, encontre mais rapidamente uma rota alternativa para o destino dos pacotes.

3 Percentual da diferença, neste trabalho, é dada pela razão entre a diferença entre o OLSR_PD e o OLSR pelo total do OLSR.

Tabela 16 - Percentual da diferença entre a taxa de repasse com sucesso entre o OLSR_PD e o OLSR dada pela razão entre a diferença entre o OLSR_PD e o OLSR pelo total do OLSR

Mobilidade entre 0-1,5 m/s		Mobilidade entre 0-20 m/s	
Nodos	Percentual da diferença	Nodos	Percentual da diferença
25	12,27%	25	29,91%
50	24,17%	50	47,54%
100	15,40%	100	36,36%
200	16,09%	200	7,14%

5.4.3. Atraso fim a fim

Na Tabela 17 e na Figura 21 são apresentados os resultados do desempenho dos protocolos OLSR e OLSR_PD em relação à média do atraso fim-a-fim. Como se observa nos resultados da Tabela 17 o OLSR_PD apresenta um aumento no atraso médio fim-a-fim em relação ao OLSR, em três dos quatro cenários com 25 e 50 nodos. Já para cenários com 100 e 200 nodos, o OLSR_PD apresenta redução no atraso em três dos quatro cenários simulados. De uma maneira geral, como era de se esperar, à medida que se aumenta o número de nodos, aumenta-se também o atraso médio e isso porque se aumenta o número de pacotes de controle na rede e a disputa pelos canais de comunicação e, neste trabalho em específico, aumenta-se também o tamanho da área na qual os nodos estão dispostos, o que faz aumentar o número de nodos intermediários nas rotas, por aumentar a distância entre os nodos mais extremos. Como exemplo, para cenários com 25 nodos o atraso médio fica na casa dos 4 e 7 centésimos de segundos, enquanto que para cenários com 200 nodos esse atraso médio aumenta, ficando entre 3,2 a 4,6 segundos.

Tabela 17 - Média do Atraso fim-a-fim por número de nodos vs mobilidade

Número de nodos	**Mobilidade de 0-1,5 m/s**		**Mobilidade de 0-20 m/s**	
	OLSR	**OLSR_PD**	**OLSR**	**OLSR_PD**
25	0,065632447	0,075923783	0,060995848	0,040636697
50	0,874465874	0,967308964	0,350206623	0,357892156
100	1,220178068	1,254863911	0,744913159	0,727538202
200	4,606038522	4,550317799	3,346305934	3,265112694

Conforme resultados da Tabela 17, o OLSR_PD registra maiores atrasos, com

relação ao OLSR, nos cenários de baixa mobilidade e em redes com 25 e 50 nodos. O aumento nesses dois casos é de aproximadamente 0,01 segundos. Já para no cenário de alta mobilidade, na rede com 25 nodos o OLSR_PD mostra uma redução no atraso médio, de aproximadamente 0,02 segundos. Para redes com 50 nodos e alta mobilidade, o OLSR_PD apresenta maior atraso médio que o OLSR, com 0,007 segundos de diferença. Para redes com 100 e 200 nodos, em cenário de alta mobilidade o OLSR_PD registra redução no atraso médio de, respectivamente, 0,017 segundos e de 0,08 segundos. Já para cenários de baixa mobilidade e rede com 100 nodos é registrado um aumento no atraso de 0,03 segundos e uma redução no atraso de 0,055 segundos para redes com 200 nodos. Desses resultados, infere-se que o OLSR_PD registra os menores atrasos nos cenários de alta mobilidade e para redes com maiores números de nodos, e isso por conta do novo critério de seleção de rotas, que aumenta a probabilidade de que nodos participantes da rota de maior grau de conectividade, consigam encontrar mais rapidamente caminhos alternativos em direção a um destino, em caso de falhas nos *links* da rota principal.

Ao se analisar os resultados da Tabela 17, especificamente para cenários com baixa mobilidade, verifica-se que o OLSR_PD apresenta um aumento na média do atraso fim-a-fim em relação ao OLSR. Alguns fatores, como o processamento adicional para se calcular o grau de conectividade, podem ter influenciado o aumento do atraso apresentado pelo OLSR_PD, todavia, é importante destacar que o atraso fim-a-fim é extraído da média do atraso de todos os pacotes recebidos nos respectivos destinos. Visto que, o OLSR_PD apresentou maior taxa de entrega de pacotes (Figura 18), e que o OLSR_PD entregou mais pacotes por rotas, cuja média de nodos intermediários eram maiores que o OLSR (Tabela 14 e Figura 19), infere-se que o tempo para envio dos pacotes por essas rotas influenciou para o aumento na média do atraso apresentada pelo OLSR_PD.

É importante destacar, que ambos os protocolos alcançam melhor desempenho, em relação à média do atraso, em todos os cenários de alta mobilidade. Essa não é a tendência, pois Choi e Ko [CHO04] afirmam que o aumento na velocidade do nodos induz a frequentes mudanças de topologia e aumenta a probabilidade de que *links* se quebrem, consumindo, então, tempo adicional com processo de descoberta de novas rotas, o que consequentemente aumenta o atraso na entrega de pacotes. A explicação para esse fato, também, deve-se à presença de mais pares fontes-destinos de tráfegos alcançáveis com apenas 1 salto nessas redes, de forma que, pacotes trocados por esses pares, não

sofrem com o processamento adicional registrado em rotas de múltiplos saltos, reduzindo, assim, a média do atraso fim-a-fim (Tabela 14 e Figura 19). Reflexos do tráfego entre vizinhos a 1 saltos é registrado na Figura 18, onde as taxas de entregas, para cenários de alta mobilidade, para redes com 25, 50 e 100 nodos, apresentam ganhos superiores às taxas apresentadas em cenários de baixa mobilidade. Na Subseção 5.4.1, os efeitos dessa disposição topológica são discutidos com mais detalhes.

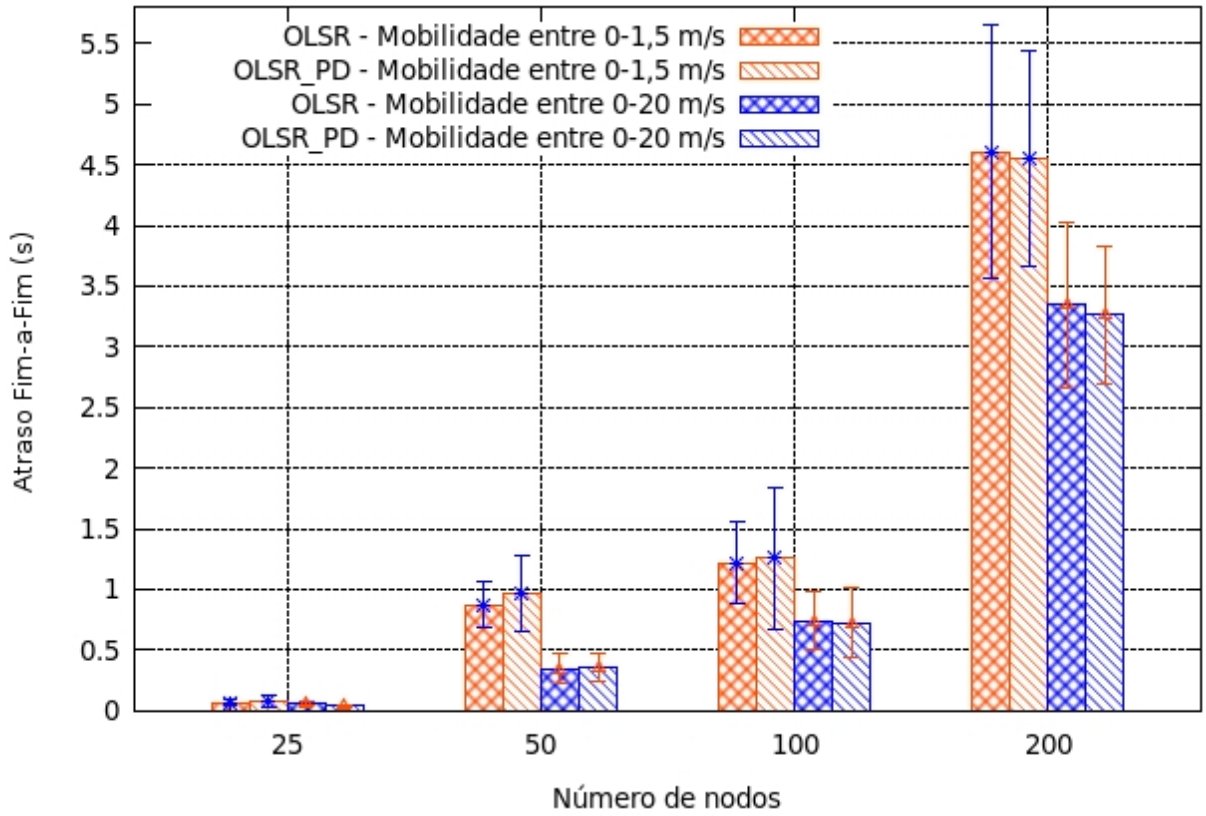

Figura 21 - Média do atraso fim-a-fim vs o número de nodos – Mobilidade entre 0-1,5 e 0-20 m/s

5.4.4. Variação do atraso (*Jitter*)

Na Figura 22 e na Tabela 18 são apresentados os resultados da variação do atraso de ambos os protocolos analisados neste trabalho. É importante destacar que a avaliação da variação do atraso e do atraso médio fornecem informações fundamentais para validação da funcionalidade do OLSR_PD, pois se os pacotes de dados são entregues com menos atraso, ou que a variação desse atraso é menor que o apresentado pelo

OLSR original, significa que o critério de seleção de rotas proposto para o OLSR_PD consegue, em caso de falhas nas rotas previamente escolhidas, realmente reduzir o tempo para que rotas alternativas sejam escolhidas.

Nos resultados apresentados na Tabela 18 e na Figura 22, se observa que o OLSR_PD tem um melhor desempenho que o OLSR original em seis das oito redes simuladas. Semelhante aos resultados apresentados na análise do atraso médio fim-a-fim, também a variação do atraso aumenta à medida que o número de nodos aumentam. As causas para esse aumento são as mesmas apresentadas na Subseção 5.4.3: aumento no tamanho da área de simulação; e aumento no número de nodos intermediários nas comunicações.

Tabela 18 - Variação do Atraso por número de nodos vs mobilidade

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	**OLSR_PD**	**OLSR**	**OLSR_PD**
25	0,086559202	0,102236287	0,104512899	0,062613169
50	1,121775782	1,127531727	0,503922730	0,455238524
100	1,288437016	1,184606155	0,831919409	0,796967492
200	3,357559377	3,068307629	2,830582691	2,544217361

Como a variação do atraso está diretamente relacionada ao atraso fim-a-fim, os dados dos gráficos mostram que, igualmente aos resultados da média do atraso, tanto o OLSR quanto o OLSR_PD apresentaram menor variação do atraso em cenários de alta mobilidade. As causas também são as mesmas e deve-se à presença de mais pares fontes-destinos de tráfegos alcançáveis com apenas 1 salto nessas redes (Tabela 14 e Figura 19), de forma que, pacotes trocados por esses pares não sofrem com o processamento adicional registrado em rotas de múltiplos saltos, reduzindo, também, a variação média do atraso. Na Subseção 5.4.1, os efeitos dessa disposição topológica são discutidos com mais detalhes.

O ganho no desempenho, apresentado pelo OLSR_PD, quanto à variação média do atraso, se mostrou bastante satisfatório, em relação ao OLSR original. A redução média da variação do atraso do OLSR_PD, chega à casa dos décimos de segundos, enquanto que, a redução no atraso médio fim-a-fim fica na casa dos centésimos de segundo. Para os cenários de baixa mobilidade, enquanto que no OLSR o menor e o maior *Jitter* médio foram, respectivamente, de 0,086559202 segundos e de 3,357559377

segundos, no OLSR_PD esses valores foram de 0,102236287 segundos e de 3,068307629 segundos. De maneira análoga, para cenários de alta mobilidade, enquanto que no OLSR o menor e o maior *Jitter* médio foram, respectivamente, de 0,104512899 segundos e de 2,830582691 segundos, e no OLSR_PD esses valores foram de 0,062613169 segundos e de 2,544217361 segundos. Com base nesses resultados verifica-se que o OLSR_PD se mostra mais adequado para operar em ambientes com maior mobilidade e número de nodos na rede.

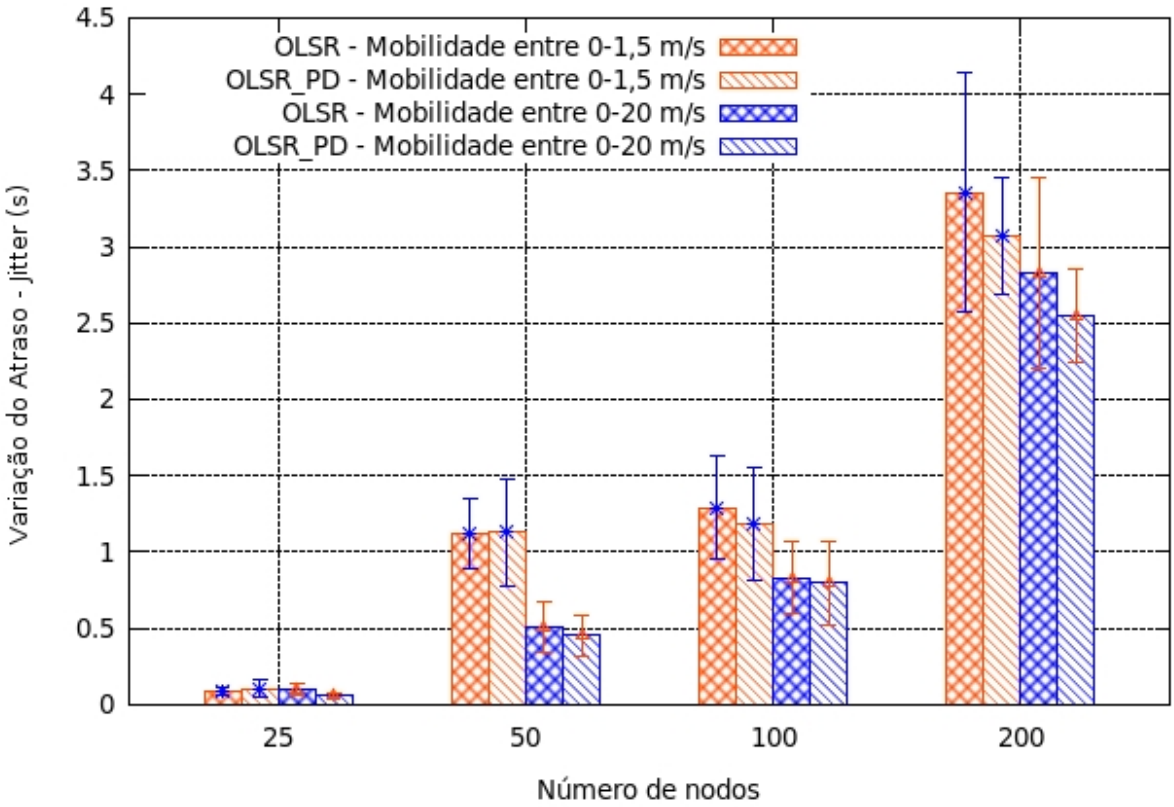

Figura 22 - Variação do atraso (*Jitter*) por número de nodos vs mobilidade entre 0-1,5 e 0-20 m/s

5.4.5. Carga de roteamento normalizada (*overhead*)

Nesta seção são apresentados os resultados da carga de roteamento normalizada que é dada pela quantidade pacotes de roteamento enviados por pacotes de dados entregues. Essa métrica visa identificar em cada protocolo quantos pacotes de roteamento são necessários para que um pacote de dados seja entregue. A carga de roteamento normalizada também será apresentada em termos de quantos *bytes* de

roteamento são necessários para que um *byte* de dados seja entregue.

Segundo Choi e Ko [CHO04], o OLSR por ser um protocolo de roteamento pró-ativo, orientado a tabelas, apresenta alta carga de roteamento por necessitar manter, em todo tempo, rotas para todos os nodos da rede. As mensagens de controle geradas pelo OLSR e OLSR_PD, consideradas para esse cálculo, são as mensagens de HELLO e as mensagens TC.

Como descrito na Seção 4.2, para implementar o critério de seleção de rotas no OLSR_PD, não foi necessário criar ou disparar nenhuma mensagem de controle adicional na rede, mas foi acrescentado ao cabeçalho da mensagem TC, um campo de dois *bytes,* para carregar as informações sobre o grau de conectividade dos nodos emissores das mensagens TC. Devido a esse fato é de se esperar um pequeno aumento no número de *bytes* de controle na rede, visto que para cada mensagem TC enviada na rede, mais dois *bytes* de controle trafegam pela rede. O total médio de *bytes* gerados em cada simulação são apresentados na Tabela 19.

Tabela 19 - Média do total de *bytes* de controle gerados em cada simulação

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	OLSR_PD	OLSR	OLSR_PD
25	344.505,20	351.024,60	288.204,80	296.571,80
50	1.182.851,60	1.193.575,80	1.059.120,80	1.069.381,60
100	3.422.302,40	3.444.403,40	4.058.555,20	4.092.836,50
200	9.782.228,40	9.788.893,40	11.877.333,60	12.121.746,80

Como observado na Tabela 19 o OLSR_PD, como previsto, acrescenta alguns *bytes* de controle na rede, todavia, como apresentado na Tabela 20 e Figura 23, em termos de quantidade de pacotes de roteamento necessários para que um pacote de dados seja entregue, o OLSR_PD apresenta um ganho em relação ao OLSR original. Esse ganho se mostra constante em todos os cenários de mobilidade e para todas as quantidades de nodos. Esse ganho deve-se primeiramente ao ganho nas taxas de entregas de pacotes apresentadas pelo OLSR_PD.

Quanto à carga de roteamento, em termos de *bytes* de controles necessários para que um *byte* de dados seja entregue, seguindo as tendências dos resultados da Tabela 20 e Figura 23, também nesse critério, o OLSR_PD apresentou menor carga de

roteamento em todos os cenários simulados, como apresentados na Tabela 21 e Figura 24.

Tabela 20 - Carga de roteamento normalizada dada pela razão entre o número de pacotes de controle enviados pelo número de pacotes de dados recebidos

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	OLSR_PD	OLSR	OLSR_PD
25	1,965545	1,862944	0,999159	0,954211
50	5,925618	4,852237	3,576968	3,165804
100	34,676169	29,983696	34,669440	32,758412
200	181,962316	171,622343	158,747746	155,585823

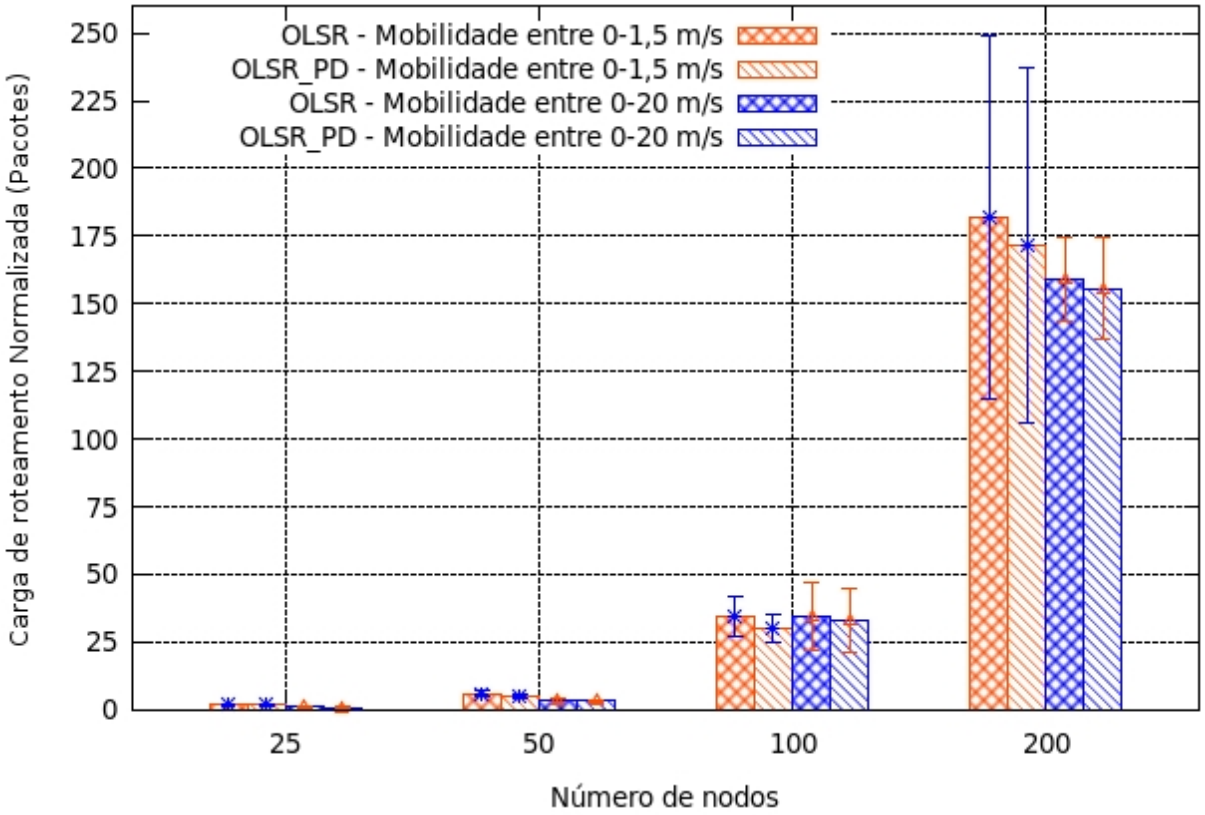

Figura 23 - Carga de roteamento normalizada em número de pacotes vs o número de nodos – mobilidade entre 0-1.5 m/s

Outro detalhe registrado nos gráficos, apresentados nesta seção, é que a carga de roteamento na rede aumenta à medida que aumenta o número nodos. Isso porque quando se aumenta o número de nodos, aumenta-se o número de nodos transmitindo ou

repassando mensagens TC por toda a rede, bem como o número de nodos enviando mensagens de HELLO. Quanto à mobilidade dos nodos, observando-se a carga de roteamento normalizada nos 2 gráficos apresentados, verifica-se que esta reduz em situações de alta mobilidade, para as redes com 25 e 50 nodos. Isso pode ser explicado pelo fato de que, a taxa de entrega de pacotes em cenários de alta mobilidade, foi maior do que a apresentada em cenários de baixa mobilidade (Figura 18), e como a carga de roteamento normalizada é dada pela razão entre pacotes de roteamento gerados por pacotes de dados entregues, o que faz com que a carga de roteamento normalizada seja menor nos cenários de alta mobilidade, o que significa que foram necessários menos pacotes de controle para que um pacote de dados fosse entregue.

Observando, porém, o total de *bytes* registrados em cada uma das simulações com baixa e alta mobilidade, apresentado na Tabela 19, verifica-se que, em ambos os protocolos, a medida que aumenta a velocidade cresce também o número de *bytes* de roteamento na rede, e isso porque a mobilidade faz romper *links* de comunicação, fazendo com que cada nodo reconstrua todas as suas tabelas novamente, gerando mais mensagens HELLO e TC na rede.

Como apresentado nesta subseção, o OLSR_PD adiciona *bytes* de roteamento extras na rede, a fim de que o critério de seleção de rotas aqui proposto possa ser implementado, todavia a análise da carga de roteamento normalizada (Tabela 20, Figura 23 e Tabela 21, Figura 24) mostra que o custo para enviar um pacote de dados no OLSR_PD é inferior, em todos os cenários, que o custo apresentado pelo OLSR original.

Tabela 21 - Carga de roteamento normalizada dada pela razão entre o número de *bytes* de controle gerados pelo número de *bytes* de dados recebidos

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	OLSR_PD	OLSR	OLSR_PD
25	0,213801	0,204667	0,115834	0,111198
50	0,744479	0,622048	0,478389	0,427856
100	4,556671	3,997817	5,086028	4,855219
200	25,815386	24,683983	25,239435	25,004784

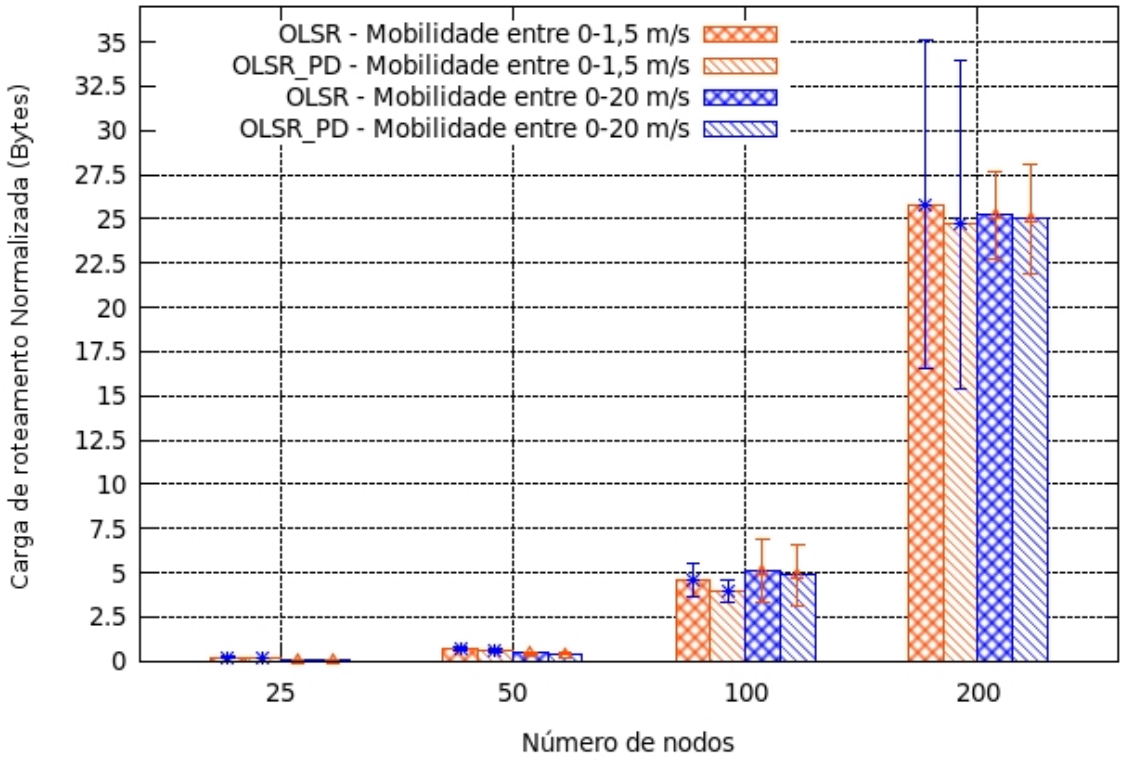

Figura 24 - Carga de roteamento normalizada em *bytes* vs o número de nodos – mobilidade entre 0-1,5 e 0-20 m/s

5.4.6. Pacotes descartados

Nesta seção são apresentados os números médio de pacotes descartados em todas as simulações e o motivo do descarte. Pacotes de dados são descartados por vários motivos, como congestionamento nas filas dos roteadores, colisões de pacotes e quebra de *links.* Todavia segundo Tanembaun [TAN03], em redes sem fio a maioria das perdas de pacotes deve-se às interferências (ruídos) no enlace aéreo.

Na Tabela 22 é apresentado um resumo da média de pacotes descartados em todos as simulações realizadas neste trabalho. De uma maneira geral, observa-se que o OLSR_PD apresenta um menor descarte de pacotes em ambos os cenários de mobilidade e para todas as quantidades de nodos.

Os dados dessa tabela revelam que a principal causa de descarte de pacotes, nessas simulações é o excesso de retransmissões sem resposta feitas pela camada MAC (*Media Access Control*). Esse tipo de evento de descarte é registrado no *trace file* como o evento CBK (*Callback*). Quando a camada de redes recebe um pacote de dados e insere

em seu cabeçalho a rota (ou o próximo salto) para chegar ao destino, então este é repassado para a camada MAC a fim de que esta obtenha acesso ao meio e transmita o pacote. Esse pacote aguarda na fila *(buffer* de saída) da camada MAC até que o acesso ao meio tenha sido alcançado ou que este pacote seja descartado.

Tabela 22 - Quantidade de pacotes descartados, motivo de descarte por protocolo e quantidade de nodos

	25 Nodos		50 Nodos		100 Nodos		200 Nodos	
Motivo do descarte – Mobilidade entre 0-1.5 m/s	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**
Descartados por NRTE na origem	192,10	205,10	94,40	105,20	322,40	306,20	1.185,00	1.198,60
Descartados por NRTE geral	101,60	82,50	19,00	17,20	58,30	76,10	143,80	150,70
Descartados por CBK	1.444,50	1.325,30	3.359,80	3.062,60	3.507,70	3.429,70	4.496,30	4.486,80
Descartados por IFQ	91,10	98,10	391,90	379,70	482,00	485,40	2.253,40	2.227,40
Outros (END, ARP, LOOP, TTL etc..)	381,00	363,70	461,40	415,90	729,90	701,90	914,30	897,20
Total de descarte	2.210,30	2.074,70	4.326,50	3.980,60	5.100,30	4.999,30	8.992,80	8.960,70
Motivo do descarte – Mobilidade entre 0-20 m/s	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**
Descartados por NRTE na origem	42,10	44,80	91,70	72,50	450,90	480,50	1.425,70	1.415,30
Descartados por NRTE geral	7,30	6,10	7,10	7,30	59,10	64,50	99,80	92,60
Descartados por CBK	1.065,50	890,20	2.825,60	2.560,10	3.122,10	3.019,70	3.808,70	3.820,80
Descartados por IFQ	88,50	67,50	282,90	254,20	492,00	464,10	2.556,60	2.594,80
Outros (END, ARP, LOOP, TTL etc..)	161,10	158,40	382,40	383,40	670,60	672,90	833,80	790,20
Total de descarte	1364,50	1167,00	3589,70	3277,50	4794,70	4701,70	8724,60	8713,70

Segundo Tanembaun [TAN03], existem duas técnicas para que a camada MAC acesse ao meio em redes sem fio *ad hoc*. O primeiro consiste basicamente em o nodo roteador escutar o meio para certificar-se de nenhum outro nodo esteja transmitindo naquele momento. Caso o meio esteja livre o pacote de dados é enviado e o nodo remetente fica aguardando uma mensagem ACK (*Acknowledgment*) acusando o recebimento do pacote, se passado um período de tempo e essa mensagem de ACK não chegar, então esse pacote é reenviado um determinado número de vezes, até que o remetente receba uma confirmação. Caso esta confirmação não chegue, após *n* tentativas de envio, o pacote é devolvido à camada de redes e, neste trabalho em específico, esse pacote é descartado recebendo o *flag* CBK. Na segunda técnica, antes que os pacotes de dados sejam enviados, as camadas MAC, do remetente e do destinatário, trocam pequenos quadros chamados RTS (*Request to Send*) e CTS (*Clear to Send*). Um quadro RTS solicita a liberação para transmitir dados, e então a camada

MAC do origem fica aguardando que o destinatário retorne um pacote CTS, que garante livre acesso ao meio até o fim da transmissão. Se o nodo origem envia um quadro RTS mas não recebe um CTS, após um número determinado de tentativas o pacote de dados é devolvido à camada de redes porque a camada MAC não conseguiu transmitir o pacote. Este é descartado, sendo registrado no arquivo de *trace* com o *flag* CBK.

As principais causas para que pacotes sejam descartados por CBK pela camada MAC são, principalmente, a quebra de *links* por causa da mobilidade, os ruídos e colisões de pacotes com os de outros transmissores [TAN03].

O segundo motivo para descarte de pacotes registrado neste trabalho, é a ausência de rota no momento em que este chega na camada de rede. No NS-2 este tipo de descarte é identificado como NRTE (Nenhuma rota disponível). Como se observa na Tabela 22, o descarte por ausência de rota está dividido em “descartados por NRTE na origem”, que equivale aos pacotes que foram descartados porque no momento em que foi iniciada a transmissão de pacotes, a tabela de roteamento do nodo origem não dispunha das rotas para o destino dos pacotes. Já pacotes “Descartados por NRTE geral”, registra o número de pacotes que foram descartados por falta de rotas em nodos intermediários da rota.

Na Figura 25, é apresentado a média normalizada de pacotes descartados por falta de rotas, a qual é dada pela razão entre média de pacotes descartados por falta de rotas (NRTE geral) pelo total de pacotes repassados com sucesso. Por esse gráfico é possível identificar qual a média de pacotes descartados por falta de rotas, para cada pacote de dados repassados com sucesso. É importante destacar que apenas pacotes descartados nos nodos intermediários entram nesse cálculo, pois a atividade de repasse nunca é realizada no nodo que originou o pacote, mas apenas nos nodos intermediários de uma rota.

Quanto a média normalizada, apresentada na Tabela 23 e Figura 25, o OLSR_PD apresenta um melhor desempenho em praticamente todas os tamanhos de rede e cenários de mobilidade, com exceção do cenário de baixa mobilidade com 100 nodos. É importante observar que no cenário de baixa mobilidade com 100 nodos o OLSR repassou 1.548,65 pacotes com sucesso, enquanto que o OLSR_PD repassou 1.787,10, o que significa que o OLSR_PD conseguiu repassar, a mais que o OLSR, a quantia de 238,45 pacotes de dados. Ocorre, todavia, que o OLSR_PD descartou uma média de 76,1 pacotes por falta de rotas nos nodos intermediários, enquanto que o OLSR descartou a

média de 58,3 pacotes, diferença esta, que fez o desempenho do OLSR_PD cair para este cenário, o que, todavia, não interferiu para que o OLSR_PD apresentasse maior taxa de entrega de pacotes do OLSR_PD, haja vista, no computo geral de descartes, o OLSR apresentou maior descarte de pacotes que o OLSR_PD.

Tabela 23 - Média normalizada de descarte por falta de rotas em nodos intermediários, dada pela razão entre pacotes "descartados por NRTE geral" por pacotes repassados com sucesso

Número de nodos	Mobilidade de 0-1,5 m/s		Mobilidade de 0-20 m/s	
	OLSR	OLSR_PD	OLSR	OLSR_PD
25	0,056994	0,041223	0,040176	0,004481
50	0,011878	0,008659	0,005502	0,003835
100	0,037646	0,042583	0,070136	0,056138
200	0,191555	0,172920	0,195475	0,169287

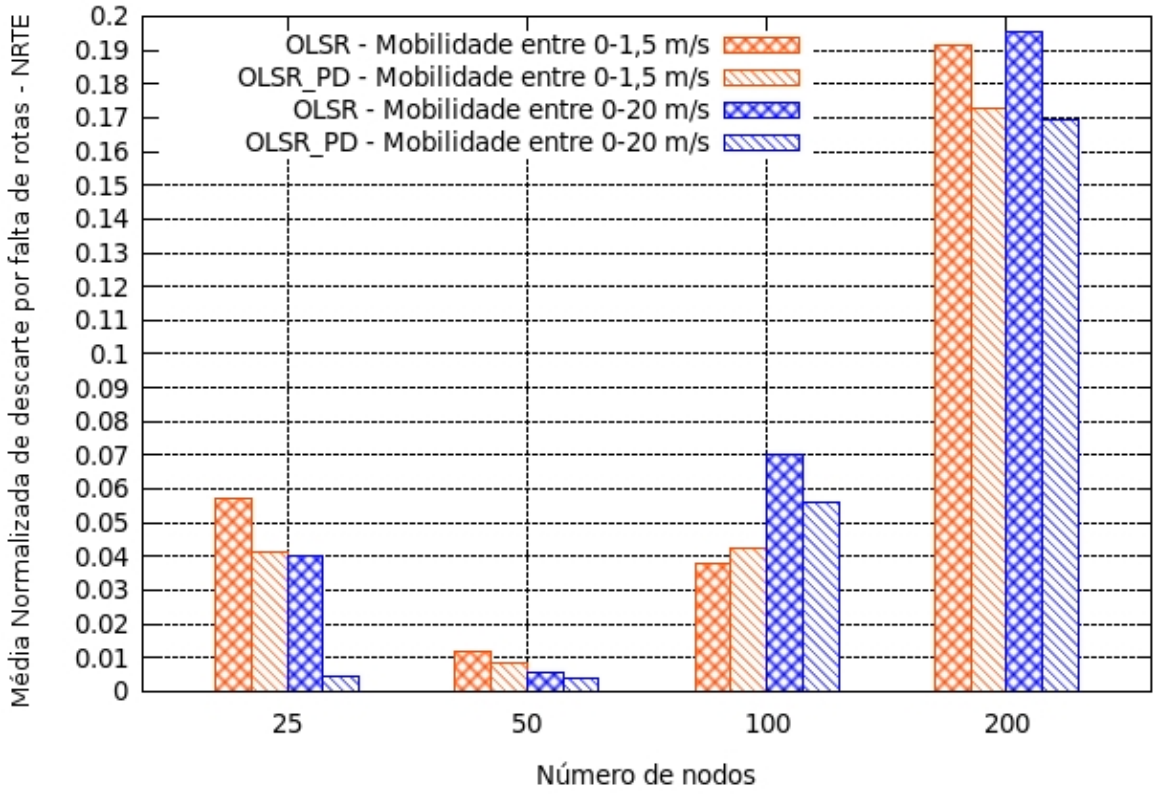

Figura 25 - Média normalizada de descarte por falta de rotas em nodos intermediários, dada pela razão entre pacotes "descartados por NRTE geral" por pacotes repassados com sucesso

Analisando, ainda, a média normalizada de descartes, foi registrada na rede com

25 nodos, no cenário de alta mobilidade, a maior diferença entre os protocolos, pois no OLSR_PD para cada pacote repassado com sucesso, uma média de 0,004481 pacotes são descartados por falta de rotas, enquanto que no OLSR, essa relação sobe para 0,040176 pacotes descartados.

A terceira maior causa de descarte de pacotes é o descarte por falta de espaço interface da fila da camada MAC que pode estar cheia, e quando os pacotes de dados chegam esses são rejeitados sendo descartados. Neste trabalho a fila da camada MAC foi configurada para no máximo 50 pacotes. Observando a Tabela 24, verifica-se que à medida que aumenta o número de nodos na rede, o descarte por falta de espaço na fila (IFQ) tem o maior aumento, em ambos os protocolos, pois enquanto nos cenários com 25 nodos, para cenário com baixa e alta mobilidade, a média de descarte por IFQ corresponde por 4,12% a 6,49% dos pacotes descartados na rede. Já para a rede com 200 nodos, para cenários de baixa e alta mobilidade, a média de descarte por IFQ representa em torno de 25% a 29% dos motivos de descarte em ambos os protocolos.

Tabela 24 - Percentuais de descartes de pacotes por motivo de descarte, quantidade de nodos e protocolo de roteamento

	25 Nodos		50 Nodos		100 Nodos		200 Nodos	
Motivo do descarte – Mobilidade entre 0-1.5 m/s	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**
Descartados por NRTE na origem	8,69%	9,89%	2,18%	2,64%	6,32%	6,12%	13,18%	13,38%
Descartados por NRTE geral	4,60%	3,98%	0,44%	0,43%	1,14%	1,52%	1,60%	1,68%
Descartados por CBK	65,35%	63,88%	77,66%	76,94%	68,77%	68,60%	50,00%	50,07%
Descartados por IFQ	4,12%	4,73%	9,06%	9,54%	9,45%	9,71%	25,06%	24,86%
Outros (END, ARP, LOOP, TTL etc..)	17,24%	17,53%	10,66%	10,45%	14,31%	14,04%	10,17%	10,01%
Total de descarte	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
Motivo do descarte – Mobilidade entre 0-20 m/s	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**	**OLSR**	**OLSR_PD**
Descartados por NRTE na origem	3,09%	3,84%	2,55%	2,21%	9,40%	10,22%	16,34%	16,24%
Descartados por NRTE geral	0,53%	0,52%	0,20%	0,22%	1,23%	1,37%	1,14%	1,06%
Descartados por CBK	78,09%	76,28%	78,71%	78,11%	65,12%	64,23%	43,65%	43,85%
Descartados por IFQ	6,49%	5,78%	7,88%	7,76%	10,26%	9,87%	29,30%	29,78%
Outros (END, ARP, LOOP, TTL etc..)	11,81%	13,57%	10,65%	11,70%	13,99%	14,31%	9,56%	9,07%
Total de descarte	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%

As outras causas de descarte de pacotes, separadamente, possuem valores pouco

expressivos, todavia, neste trabalho, também, foram identificadas mais quatro causas de descarte de pacotes, sendo seus *flags* “END” que indica pacotes que foram descartados porque a simulação chegou ao fim, e eles ainda estavam na fila da camada MAC de algum nodo. Outra causa é quando um pacote tem seu tempo de vida TTL (*Time to Live*) expirado e tem de ser descartado para não ficar percorrendo a rede indefinidamente. Um pacote também pode ser descartado se ele for recebido mais que uma vez por um receptor, pois fica caracterizado que esse pacote pode estar em *loop* infinito, logo é descartado e no arquivo *trace* recebe o *flag* “LOOP“. Por último, destacamos o descarte por ARP (*Adress Resolution Protocol*). Esse tipo de descarte ocorre quando um pacote de dados sai da camada de rede. Primeiro ele passa pela camada de *link* de dados LL (*Link Layer*) que converte o endereço de IP em endereços de MAC, ou seja, em endereço físico do destinatário. Se o ARP possui o endereço físico do destino, ele o coloca no cabeçalho do pacote, senão o pacote é armazenado temporariamente no *buffer* ARP, que envia uma mensagem, por *broadcast,* a fim de identificar o endereço MAC do nodo portador do endereço de IP fornecido no pacote de dados. Esse *buffer* possui espaço para apenas um pacote, para cada destino desconhecido. Se no intervalo em que está sendo feita a consulta chegar um novo pacote, cujo endereço de IP seja o mesmo de um pacote já armazenado no *buffer* ARP, o pacote mais antigo é descartado e o novo é alocado em seu lugar. Pacotes descartados por esse motivo recebem o *flag* ARP.

Nesta subseção se verificou que em ambos os protocolos o descarte de pacotes é alto, reduzindo significativamente a taxa de entrega de pacotes (Figura 18), todavia se observa que o OLSR_PD consegue apresentar uma redução no número de descartes, de uma maneira geral, e um ganho na média normalizada de pacotes/*bytes* descartados.

6. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho é apresentada a proposta de um novo critério de seleção de rotas para o OLSR, que além do critério do comprimento da rota, considera ainda, se esta possui o maior grau de conectividade, dentre as rotas disponíveis. A essa versão do OSLR chamamos de OLSR_PD ou OLSR *baseado no grau de conectividade do caminho*. Para implementar o OLSR_PD foram mantidas as principais características do OLSR, e alterado apenas o critério de seleção de rotas.

Para avaliar o desempenho desses dois protocolos foram executadas várias baterias de simulações, utilizando o simulador de redes NS-2, cujos resultados foram analisados e apresentados neste trabalho. De uma maneira geral, observou-se que o OLSR_PD apresenta um ganho de desempenho nos quesitos taxa de entrega de pacotes e taxa de repasse de pacotes com sucesso, e isso foi registrado em todos os cenários de mobilidade e quantidades de nodos, comprovando assim que o critério de seleção de rotas proposto aumenta a probabilidade de entrega de pacotes, visto que reduz o número de de pacotes descartados em todos os cenários simulados.

Na avaliação da métrica atraso médio fim-a-fim o OLSR_PD apresenta melhor desempenho médio em quatro das oito simulações executadas, obtendo melhor desempenho em cenários de maior mobilidade e maior quantidade de nodos. Quanto à variação desse atraso (*Jitter*) o OLSR_PD também apresenta um ganho de desempenho, em relação ao OLSR original, pois reduz a média da variação do atraso em seisdas oito simulações apresentadas, também apresentando melhor desempenho no ambiente de maior mobilidade. Esses resultados reforçam a tese de que a seleção de rotas que possuem o maior grau de conectividade, pode aumentar a probabilidade, de que, em casos de falhas nessas rotas os nodos imediatamente anteriores às falhas, por pertencer à rota de maior grau, têm maior probabilidade, se comparado a nodos de outras rotas, de calcular, mais rapidamente, uma rota alternativa para o destino.

Os resultados experimentais apontam, como já era previsto, que o OLSR_PD acrescenta alguns *bytes* de controle na rede, se comparado ao OLSR. Todavia quando é feita a análise da carga de roteamento normalizada entre esses protocolos, verifica-se que o custo para transmitir 1 *byte* no OLSR_PD é menor que no OLSR, o que torna insignificante o número de *bytes* de controle a mais na rede, se comparado aos benefícios

do aumento da taxa de entrega de pacotes.

De uma maneira geral, o OLSR_PD apresentou um desempenho superior ao OLSR original. Consegue apresentar maiores taxas de entrega em cenários de baixa mobilidade (0-1,5 m/s) e mantém esse desempenho em cenários de alta mobilidade (0-20 m/s). Considerando o número de nodos na rede, ambos os protocolos apresentam perda de desempenho, mas mesmo nesses cenários o OLSR_PD mantem um ganho em relação ao OLSR.

Como proposta para trabalhos futuros, fica a ideia de acrescentar ao critério de seleção de rotas do OLSR_PD uma nova característica, que considere, também, a média de conectividade dos nodos desse caminho, de forma que seja escolhida sempre a menor rota cujos nodos do caminho possuam o maior grau médio de conectividade. Também deixamos a proposta de implementar critérios de QoS no OLSR_PD para seleção dos nodos MPR de forma que a menor rota de maior grau de conectividade também seja uma rota que atenda à requisitos de QoS. Sugerimos ainda a avaliação do OLSR e do OLSR_PD utilizando-se outras métricas de avaliação de desempenho, como variação na densidade da rede. Por fim sugerimos que a avaliação do desempenho do OLSR_PD em relação às outras extensões do OLSR como o QOLSR, OLSR-ML e o OLSR-LD e com outros protocolos pró-ativos e reativos utilizados em MANETs.

REFERÊNCIAS BIBLIOGRÁFICAS

[ABO04] ABOLHASAN, M.; WYSOCK, T.; DUTKIEWICZ, E. "A review of routing protocols for mobile ad hoc networks". *Ad Hoc Networks,* vol. 2-1, Janeiro 2004, pp. 1-22.

[ASL04] ASLAM, N.; PHILLIPS, W.; ROBERTSON, W. “Composite Metrics for Quality of Service Routing in OLSR”. In: 17th Canadian Conference On Electrical and Computer Engineering, 2004, pp. 759-762.

[BAD04] BADIS, H.; MUNARETTO, A.; AGHA, K.; PUJOLLE, G. “Optimal path selection in a link state QoS routing protocol”. In: 60th IEEE Vehicular Technology Conference, Vol. 5, 2004, pp. 2570-2574.

[BEL57] BELMAN, R. E. “Dynamic Programming”. Princeton University Press, Princeton, 1957, 360p.

[CAS02] CASTANEDA, R.; DAS, S. R.; MARINA, M. K. "Query Localization Techniques for On-Demand Routing Protocols in Ad Hoc Networks". *Wireless Networks,* Vol. 8, Março 2002, pp. 137-151.

[CHA08] CHAKERES, I. D.; PERKINS, C. E. “Dynamic MANET On-demand (DYMO) Routing Protocol”. Capturado em: http://tools.ietf.org/html/draft-ietf-manet-dymo-12, Agosto 2008.

[CHE05] CHEN, L. J.; SUN, T.; YANG, G.; SANADINI, M. Y.; GERLA, M. “AdHoc Probe: Path Capacity Probing in Wireless Ad Hoc Networks”. In: IEEE International Conference on Wireless Internet, 2005, pp. 156-163.

[CHO04] CHOI, J. M.; KO, Y. B. "A performance evaluation for ad hoc routing protocols in realistic military scenarios". In: 9th International Conference on Cellular and Intelligent Communications, Outubro 2004, 5p.

[COR02] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN C. “Algoritmos: Teoria e Prática. Rio de Janeiro: Elsevier, 2002, 2 Ed, 916p.

[COR07] CORDEIRO, W. L. da C. “Provisionamento de Qualidade de Serviço em Redes Ad Hoc sem Fio Utilizando Medição de Retardo de Enlace” Trabalho de conclusão de curso, UFPA, 2007, 102p.

[COR08] CORSON, S.; MACKER J. "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations". Capturado em: http://tools.ietf.org/html/draft-ietf-manet-issues-02, Setembro 2008.

[COU03] COUTO, D. S. J.; AGUAYO, D.; BICKET, J.; MORRIS, R. “A High-Throughput Path Metric for Multi-Hop Wireless Routing”. In: 9th International Conference on Mobile Computing and Networking”, 2003, pp. 134-146.

[CRA98] CRAWLEY, E.; NAIR, R.; RAJOGOPALAN B.; SANDICK H. “A Framework for QoS-based Routing in the Internet”, RFC 2386, Agosto 1998, 37p.

[DIJ59] DIJKSTRA E. W. "A note on two problems in connexion with graphs".

Numerische Mathematik, vol. 1, 1959, pp. 269-271.

[DRO08] DROZDEK, A. “Estrutura de dados e algoritmos em C++”. São Paulo: Cengage Learning. 2008, 579p.

[FAL09] FALL, K. E.; VARADHAN K. “NS Notes and Documentation”. Capturado em: http://www.isi.edu/nsnam/ns/ns-documentation.html, abril de 2009.

[FOR62] FORD, L. R.; FULKERSON, D. R. “Flows in Networks”, Princeton University Press, Princeton, NJ, 1962, 208p.

[GOW07] GOWRISHANKAR, S.; BASAVARAJU, T. G.; SINGH, M.; SARKAR S. K. "Scenario based Performance Analysis of AODV and OLSR in Mobile Ad hoc Networks", In: 24th South East Asia Regional Computer Conference, 2007, pp. 18-19.

[HAE06] HAERRI J.; FILALI F.; BONNET C. "Performance Comparison of AODV and OLSR in VANETs Urban Environments under Realistic Mobility Patterns", In: 5th IFIP Mediterranean Ad-Hoc Networking Workshop, 2006, pp. 14-17.

[HAS08] HASS, Z. J.; PEARLMAN, R. “Zone routing protocol for ad-hoc networks”. Capturado em: http://net.pku.edu.cn/yidong/ROOT/draft-ietf-manet-zone-zrp-04.txt, Setembro 2008.

[HSU03] HSU, J.; BHATIA, S.; TAKAI, M.; BAGRO, R.; ACRICHE, M. J. “Performance of Mobile Ad Hoc Networking Routing Protocols in Realistic Scenarios.” In: Military Communications Conference, MILCOM 2003. IEEE, 2003, pp. 1268-1273.

[HUH04] HUHTONEN, A. "Comparing AODV and OLSR Routing Protocols", HUT T-110.551 Seminar on Internetworking, Sjökulla, 2004, 9p.

[JAC01] JACQUET, P.; et al., “Optimized link state routing protocol for ad hoc networks (OLSR)”. In: 21st Multi Topic Conference, IEEE INMIC, 2001, pp. 62-68.

[JAC02] JACQUET, P.; LAOUITI, A.; MINET P.; VIENNOT; Laurent. "Performance of multipoint relaying in ad hoc mobile routing protocols". In: 2th International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols, 2002, pp. 387-398.

[JAC03] JACQUET P.; MUHLETHALER, P.; QAYYUM, A. “Optimized link state routing protocol”. IETF draft, RFC 3626, 2003, 75p.

[JAC97] JACQUET, P.; MINET P.; MUHLETHALER P.; RIVIERRE N. “Increasing realibility in cable-free Radio LANs: Low level forwarding in HIPERLAN”. *Wireless Personal Communications*, vol. 4, Janeiro 1997, pp. 65-80.

[JAI91] JAIN, R. “The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling” John Wiley & Sons. 1991, 17p.

[JOH96] JOHNSON, D. B.; MALTZ, D. A. "Dynamic Source Routing in Ad Hoc Wireless Networks". *Mobile Computing*, vol. 353, 1996, pp. 153-181.

[JUN08] JÚNIOR, I. D. de J. “QoS para VoIP através do HTB”. Capturado em:

http://repositorio.zaeel.org/artigos/qos_voip_htb.pdf, Novembro 2008.

[KIM06] KIM, D.; LEE, W.; PARK, B.-N.; KIM, J.; SHIN, C.; SHIN, C. “A Power Balanced Multipath Routing Protocol inWireless Ad-Hoc Sensor Networks”. In: 6th IEEE International Conference on Computer and Information Technology (CIT'06), 2006, 222p.

[KUR06] KUROSE, J. F.; ROSS, K. W. “Redes de computadores e a internet: uma abordagem top-down”. São Paulo: Pearson Addison Wesley, 2006, 3ª ed, 656p.

[LAB00] LABOVITZ, C.; AHUJA, A.; BOSE A.; JAHANIAN, F. “Delayed internet routing convergence”. *IEEE/ACM Transactions on Networking* (TON), vol. 9-3, Junho 2001, pp. 293-306.

[LAM03] LAMONT, L.; GE, Y.; KUNZ, T. “Quality of Service Routing in Ad hoc Networks Using OLSR”. In: 36th HAWAII International Conference on System Sciences, IEEE, 2003, 9p.

[LAW91] LAW, A. M.; KELTON D. W. “Simulation Modeling and Analysis”. Ed. Hardcover McGraw-Hill Inc., 2 ed, 1991, 759p.

[LEE01] LEE, S. J.; GERLA, M. ”Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks”. In: ICC 2001 - IEEE International Conference on Communications, 2001, vol. 10, pp. 3201-3205.

[LEG06] LEGUAY, J.; CONAN, V.; FRIEDMAN, T. “QoS Routing in OLSR with Several Classes of Service”. In: 4th IEEE International Conference on Pervasive Computing and Communications Workshops, 2006, 420p.

[MAC97] MACKER, J. and CORSON, S. “Mobile Ad Hoc Networks (MANET) - IETF WG Charter”. Capturado em: http://www.ietf.org/html.charters/manet-charter.html, Novembro 2008.

[MAL03] MALEK, J.; NOWAK, K. “Trace Graph”. Capturado em: http://www.tracegraph.com./download.html, Novembro 2008.

[MAL08] MALKIN, G. “RIP version 2 Carrying additional information - RFC 1723”. Capturado em: http://www.faqs.org/rfcs/rfc1723.html, Outubro 2008.

[MAR02] MARINA, M. K.; DAS S. R., “Ad hoc On-demand Multipath Distance Vector Routing”. ACM SIGMOBILE Mobile Computing and Communications Review, vol. 6-3, Julho 2002, pp. 92-93.

[MAR08] MARTINS, J., “Qualidade de Serviço (QoS) em Redes IP, Principios Básicos, Parâmetros e Mecanismos”. Capturado em: http://professores.unisanta.br/santana/downloads%5CTelematica%5CCom_Dados_2%5CTexto%20QoS_IP_Itelcon.pdf, Novembro 2008.

[MEL01] MELLO, E. T. L. “Qualidade de Serviço em Redes IP com DiffServ: Avaliação através de medições”. Dissertação de Mestrado, Programa de Pós-Graduação em Ciências da Computação, UFSC, 2001, 110p.

[MOY89] MOY, J. “The OSPF specification RFC 1131”. Capturado em: http://www.ietf.org/rfc/rfc1131.txt, Outubro 2008.

[NGU06] NGUYEN, D. Q.; MINET, P. “QoS support and OLSR routing in a mobile ad hoc network”. In: IEEE Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL’06), 2006, 74p.

[OOM06] OOMM, B. J.; MISRA, S. “A Fault-Tolerant Routing Algorithm for Ad Hoc Networks Using a Stochastic Learning-Based Weak Estimation procedure”. In: 2nd IEEE International Conference On Wireless and Mobile Computing, Networking and Communications (WiMob 2006), 2006, pp. 31-37.

[PAS06] PASSOS, D.; TEIXEIRA, D. V.; MAGALHÃES, L. C. S.; ALBUQUERQUE, C. V. N. “Mesh Network Performance Measurements”. In: 5th International Information and Telecommunication Technologies Symposium, 2006, 8p.

[PAW03] PAWLIKOWSKI, K. “Do Not Trust All Simulation Studies Of Telecommunication Networks”, In: ICOIN’2003, International Conference on Information Networking, 2003, pp. 899-908.

[PER01] PERKINS, C. E.; ROYER E. M.; DAS. S. R.; MARINA, M. K. “Performance comparison of two on-demand routing protocols for ad hoc networks”, In: Proceedings on Personal Communications, 2001, pp. 16-28.

[PER03] PERKINS, C.; ROYER, E. M.; DAS, S. “Ad hoc On-Demand Distance Vector (AODV) Routing”. Capturado em: http://www.faqs.org/rfcs/rfc3561.html, Outubro 2008.

[PER94] PERKINS, C. E.; BHAGWAT, P. "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers". In: ACM Sigcomm, 1994, pp. 234-244.

[QAY02] QAYYUM, A.; VIENNOT, L.; LAOUITI, A. “Multipoint relaying technique for flooding broadcast message in mobile wireless networks”. In: 35th Hawaii International Conference on System Sciences, 2002, Island of Hawaii. Proceedings of...Washington DC: IEEE Press, 2002, p. 3866-3875.

[ROS08] ROS, F. J. “UM-OLSR – Implementação do protocolo de roteamento OLSR para o Network Simulator (NS-2)”. Capturado em: http://masimum.dif.um.es/um-olsr/html/, Novembro 2008.

[ROY99] ROYER E. M. TOH, C. K., "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks". *IEEE Personal Communications*, vol. 6-2, Abril 1999, pp. 46-55.

[SCH07] SCHROEDER, J.; JUNIOR, E. P. D. “Fault-Tolerant Dynamic Routing Based on Maximum Flow Evaluation”. In: 3rd IEEE Latin American Dependable Computing Symposium (IEEE LADC'2007), 2007, pp. 7-24.

[TAN03] TANEMBAUN, Andrew S. “Redes de Computadores”, Rio de Janeiro: Editora Campus, 2003, 4 ed, 945p.

[VIL03] VILLELA, B. A. M.; DUARTE, O. C. M. B. “Uma análise de protocolos de roteamento sob demanda de redes ad hoc”. In: XX Simpósio Brasileiro de Telecomunicações - SBT'03, 2003, 6p.

[XUE03] XUE Y.; NAHRSTEDT, K. “Fault Tolerant Routing in Mobile Ad Hoc Networks”. In: Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2003, pp. 1174-1179.

APÊNDICE A - Algoritmo para o cálculo da tabela de roteamento no OLSR

Estruturas de dados básicas para cálculo da tabela de roteamento do OLSR

```
//Tabela de roteamento de um nodo OLSR
struct OLSR_rt_entry {
    nsAddr          R_dest_addr;        //Endereço do destino do tipo "Endereço do NS2"
    nsAddr          R_next_addr;        //Endereço do próximo salto em direção ao destino
    nsAddr          R_iface_addr;       //Endereço da interface de rede local
    Int_32bits      R_dist;             //Distância em saltos do tipo inteiro de 32 bits

    OLSR_rt_entry*  rtable;     //Estrutura para buscas na tabela de roteamento
}

// tabela de interfaces associadas a um nodo
struct OLSR_iface_assoc_tuple {
    nsAddr          I_iface_addr;       //Endereço da interface de um nodo
    nsAddr          I_main_addr;        //Principal endereço de um nodo
    Real            I_time;             //Hora que tupla expira e deve ser removida

    OLSR_iface_assoc_tuple*         ifaceassocset;  //vetor ponteiro do tipo tabela interfaces
                                                    //de um nodo
}

//Registro de links de comunicação de um nodo com seus vizinhos
struct OLSR_link_tuple {
    nsAddr          L_local_iface_addr; //Interface de rede do nodo local
    nsAddr          L_nb_iface_addr;    //Interface de rede do nodo vizinho
    real            L_sym_time;         //O link é bidirecional até este tempo
    real            L_asym_time;        //O link é unidirecional até este tempo
    real            L_time;             //Hora que a tupla expira e deve ser removida

    OLSR_link_tuple*    linkset;    //vetor ponteiro do tipo conjunto de links
}

//Registro de vizinhos a 1 salto
struct OLSR_nb_tuple {
    nsAddr          N_nb_main_addr;         //Principal endereço do nodo vizinho
    int_8_bits      N_status;               //Tipo de vizinho e tipo de link
    int_8_bits      N_willingness;          //disposição em repassar pacotes para outros nodos

    OLSR_nb_tuple*      nbset;  //vetor ponteiro do tipo tabela de vizinhos a 1 salto
}

//Registro de vizinhos a 2 salto
struct OLSR_nb2hop_tuple {
    nsAddr          nb_main_addr;       //Endereço da principal interface do vizinho
    nsAddr          nb2hop_addr;        //Endereço principal do vizinho a 2 saltos
    Real            time;               //Tempo que a tupla expira e dever ser removida

    OLSR_nb2hop_tuple*          nb2hopset;      //vetor ponteiro do tipo tabela de vizinhos a 2
}
```

```
//Tabela de topologia
struct OLSR_topology_tuple {
    nsAddr      T_dest_addr;    // Endereço da principal interface do destino
    nsAddr      T_last_addr;    // Endereço principal do nodo que é vizinho do destino
    int_16b     T_seq;          // Número de sequência da mensagem TC
    real        T_time;         // Tempo no qual essa tupla expira e deve ser removida

    OLSR_topology_tuple*    topologyset;    //vetor ponteiro do tipo tabela de topologia
}
```

Variáveis adicionais utilizadas

added: variável lógica utilizada para controlar a execução sobre a tabela de topologia
entry: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry1: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry2: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry3: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry4: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
h: variável de controle utilizada para registrar a distância dos nodos pesquisados
it: armazena uma tupla da tabela de roteamento
it1: armazena uma tupla da tabela de vizinhos a 1 salto
it2: armazena uma tupla da tabela de links de comunicação
it3: armazena uma tupla da tabela de vizinhos a 2 saltos
it4: armazena uma tupla da tabela de topologia
it5: armazena uma tupla da tabela de tuplas associadas
link_tuple: ponteiro que faz referência para o endereço da variável it2
nb_tuple: ponteiro que aponta para tuplas da tabela de vizinhos a 1 salto
nb2hop_tuple: ponteiro que faz referência para o endereço da variável it3
ok: variável de controle
topology_tuple: ponteiro que faz referência para o endereço da variável it4
tuple: ponteiro que faz referência para o endereço da variável it5

Funções Utilizadas no Cálculo da Tabela de Roteamento

Function Void *delete();*
// apaga cada entrada da tabela de roteamento e libera a memória
Function ***OLSR_rt_entry**** *lookup*(nsAddr *dest*)
// Procura uma entrada na tabela de rotas para um destino // específico. Retorna a// entrada de rota para o destino dest, se este for encontrado.
Function int_8b& *status()*
//Retorna o status do link do nodo atual com o nodo vizinho
Function ***nsAddr*** *get_main_addr(**nsAddr** iface_addr)*
//Retorna o endereço da principal interface de um nodo
Function ***OLSR_nb_tuple**** *find_sym_nb_tuple*(***nsAddr*** *main_addr)* //Verifica se o nodo //de endereço *main_addr,* passado como parâmetro é um vizinho simétrico a 1 salto
Function ***OLSR_nb_tuple**** *find_nb_tuple(**nsAddr** main_addr,* **int_8b** *willingness)* //Verifica se o nodo de endereço *main_addr,* é um vizinho a 1 salto e se possui //disposição em cooperar diferente de "Nunca disponível"

Algoritmo utilizado para o cálculo da tabela de roteamento do OLSR

```
Procedure rtable_computation(){
    //1º passo: Função que remove todas as entradas de rotas da tabela de roteamento
    for (OLSR_rt_entry* it ← rtable.inicio; it <> rtable.fim; it++)
    {
        delete(*it);
```

```
}

//2º passo: Rotas para vizinhos simétricos a 1 salto (h=1) são calculadas.
    //Lê todas as entradas de vizinhos a 1 salto
for (OLSR_nb_tuple it1 ← nbset.inicio; it1 <> nbset.fim; it1++) {
    OLSR_nb_tuple* nb_tuple ← *it1;
    // Verifica se o status do link com o destino é simétrico
    if (nb_tuple.N_status = SIMETRICO) {
        for (OLSR_link_tuple it2 ← linkset.inicio; it2 <> linkset.fim; it2++) {
            OLSR_link_tuple* link_tuple ← *it2;
            //verifica se o tempo de validade do link é maior que a hora atual
            if (get_main_addr(link_tuple.L_nb_iface_addr) = nb_tuple.N_nb_main_addr
                        and link_tuple.L_time >= HORA_ATUAL) {
                //Cria uma entrada, na tabela de roteamento, para cada vizinho a 1 salto
                rtable.R_dest_addr      ← link_tuple.L_nb_iface_addr;
                rtable.R_next_addr      ← link_tuple.L_nb_iface_addr;
                rtable.R_iface_addr     ← link_tuple.L_local_iface_addr;
                rtable.R_dist           ← 1;
            }
        }
    }
}

//3º passo: Cálculo de rotas para vizinhos a 2 saltos
for (OLSR_nb2hop_tuple it3 ← nb2hopset.begin; it3 <> nb2hopset.end; it3++) {
    OLSR_nb2hop_tuple* nb2hop_tuple ← *it3;
    booleano ok ← true;
    //verifica se nb_main_addr, da tabela de vizinhos a 2 saltos, é um vizinho a 1 salto
    OLSR_nb_tuple* nb_tuple ← find_sym_nb_tuple(nb2hop_tuple.nb_main_addr);
    if (nb_tuple = NULL)
        ok ← false;
    else {
        //verifica se nb_main_addr é um vizinho a 1 salto e se possui disposição em
    //repassar pacotes, em nome de outros nodos, diferente de "Nunca disponivel"
        nb_tuple ← find_nb_tuple(nb2hop_tuple.nb_main_addr, Nunca_disponivel);
        if (nb_tuple <> NULL)
            ok ← false;
        else {
            // Verifica se o nodo a 2 saltos não é também um vizinho a 1 salto
            nb_tuple ← find_sym_nb_tuple(nb2hop_tuple.nb2hop_addr);
            if (nb_tuple <> NULL)
                ok ← false;
        }
    }

    if (ok) {
        OLSR_rt_entry* entry ← lookup(nb2hop_tuple.nb_main_addr);
        //Cria uma nova entrada de rota para cada nodo a 2 saltos
        rtable.R_dest_addr      ← nb2hop_tuple.nb2hop_addr;
        rtable.R_next_addr      ← entry.R_next_addr;
        rtable.R_iface_addr     ← entry.R_iface_addr;
        rtable.R_dist           ← 2;
    }
}

// 4º passo: busca os destinos a mais de 2 saltos, até que não seja encontrado mais nodos
for (int_32b h ← 2; ; h++) {
    booleano added ← false;
    // cada entrada da tabela de topologia é analisada
```

```
        for (OLSR_topology_tuple it4 ← topologyset.begin; it4 <> topologyset.end; it4++) {
            OLSR_topology_tuple* topology_tuple ← *it4;
            //Procura na tabela de rotas se existe R_dest_addr igual a T_dest_addr
            OLSR_rt_entry* entry1 ← lookup(topology_tuple.T_dest_addr);
            //Procura na tabela de rotas se existe R_dest_addr igual a T_last_addr
            OLSR_rt_entry* entry2 ← lookup(topology_tuple.T_last_addr);
            //Se não existir R_dest_addr = T_dest_addr e existir  R_dest_addr = T_last_addr
        //e a distância R.dist = h, então cria-se uma nova entrada de rota
            if (entry1 = NULL and entry2 <> NULL and entry2.R_dist = h) {
                //Cria uma nova entrada de rota para cada nodo a 2 saltos
                rtable.R_dest_addr      ← topology_tuple.T_dest_addr;
                rtable.R_next_addr      ← entry2.R_next_addr;
                rtable.R_iface_addr     ← entry2.R_iface_addr;
                rtable.R_dist           ← h+1;
                added ← true;
            }
        }

    // 5º passo: verifica se existe placas de redes adicionais no nodos
        for (OLSR_iface_assoc_tuple it5 ← ifaceassocset.begin; it5<>ifaceassocset.end; it5++)
    {
            OLSR_iface_assoc_tuple* tuple ← *it5;
            OLSR_rt_entry* entry3 ← lookup(tuple.I_main_addr);
            OLSR_rt_entry* entry4 ← lookup(tuple.I_iface_addr);
            if (entry3 <> NULL and entry4= NULL) {
                //Cria uma nova entrada de rota para cada nodo a 2 saltos
                rtable.R_dest_addr      ← tuple.I_iface_addr;
                rtable.R_next_addr      ← entry3.R_next_addr;
                rtable.R_iface_addr     ← entry3.R_iface_addr;
                rtable.R_dist           ← entry3.R_dist;
                added ← true;
            }
        }

        if not(added)
            // sai do laço de repetição "for" porque não foi encontrado nodos destinos
            break;
    }
}
```

APÊNDICE B - Algoritmo para o cálculo da tabela de roteamento no OLSR_PD

Estruturas de dados utilizadas para o cálculo da tabela de roteamento no OLSR_PD

```
//Tabela de roteamento de um nodo OLSR_PD
struct OLSR_PD_rt_entry {
    nsAddr          R_dest_addr;        //Endereço do destino do tipo "Endereço do NS2"
    nsAddr          R_next_addr;        //Endereço do próximo salto em direção ao destino
    nsAddr          R_iface_addr;       //Endereço da interface de rede local
    Int_32bits      R_dist;             //Distância em saltos do tipo inteiro de 32 bits
    int_16bits      R_soma_grau;        //Grau de conectividade da rota até o nodo R_dest_addr

    OLSR_PD_rt_entry*       rtable;     //Estrutura para buscas na tabela de roteamento
}

// tabela de interfaces associadas a um nodo
struct OLSR_PD_iface_assoc_tuple {
    nsAddr          I_iface_addr;       //Endereço da interface de um nodo
    nsAddr          I_main_addr;        //Principal endereço de um nodo
    Real            I_time;             //Hora que tupla expira e deve ser removida

    OLSR_PD_iface_assoc_tuple*      ifaceassocset;  //vetor ponteiro do tipo tabela interfaces
                                                    //de um nodo
}

//Registro de links de comunicação de um nodo com seus vizinhos
struct OLSR_PD_link_tuple {
    nsAddr          L_local_iface_addr; //Interface de rede do nodo local
    nsAddr          L_nb_iface_addr;    //Interface de rede do nodo vizinho
    real            L_sym_time;         //O link é bidirecional até este tempo
    real            L_asym_time;        //O link é unidirecional até este tempo
    real            L_time;             //Hora que a tupla expira e deve ser removida

    OLSR_PD_link_tuple*     linkset;    //vetor ponteiro do tipo conjunto de links
}

//Registro de vizinhos a 1 salto
struct OLSR_PD_nb_tuple {
    nsAddr          N_nb_main_addr;     //Principal endereço do nodo vizinho
    int_8_bits      N_status;           //Tipo de vizinho e tipo de link
    int_8_bits      N_willingness;      //disposição em repassar pacotes para outros nodos

    OLSR_PD_nb_tuple*       nbset;      //vetor ponteiro do tipo tabela de vizinhos a 1 salto
}

//Registro de vizinhos a 2 salto
struct OLSR_PD_nb2hop_tuple {
    nsAddr          nb_main_addr;       //Endereço da principal interface do vizinho
    nsAddr          nb2hop_addr;        //Endereço principal do vizinho a 2 saltos
    Real            time;               //Tempo que a tupla expira e dever ser removida

    OLSR_PD_nb2hop_tuple*   nb2hopset;  //vetor ponteiro do tipo tabela de vizinhos a 2
}
```

```
//Tabela de topologia
struct OLSR_PD_topology_tuple {
    nsAddr      T_dest_addr;      // Endereço da principal interface do destino
    nsAddr      T_last_addr;      // Endereço principal do nodo que é vizinho do destino
    int_16b     T_seq;            // Número de sequência da mensagem TC
    real        T_time;           // Tempo no qual essa tupla expira e deve ser removida
    int_16b     T_grau;       // Grau do nodo T_dest_addr informado na mensagem TC saltos

    OLSR_PD_topology_tuple*   topologyset;    //vetor ponteiro do tipo tabela de topologia
}
```

Variáveis adicionais utilizadas

added: variável lógica utilizada para controlar a execução sobre a tabela de topologia
entry: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry1: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry2: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry3: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
entry4: ponteiro que faz referência a um endereço de tupla da tabela de roteamento
grau_1hop: armazena o grau de conectividade de um vizinho a 1 salto
grau_2hop: armazena o grau de conectividade de um vizinho a 2 saltos
grau_nodo: armazena temporariamente a soma do grau de um nodo a mais de 2 saltos
h: variável de controle utilizada para registrar a distância dos nodos pesquisados
it: armazena uma tupla da tabela de roteamento
it1: armazena uma tupla da tabela de vizinhos a 1 salto
it2: armazena uma tupla da tabela de *links* de comunicação
it3: armazena uma tupla da tabela de vizinhos a 2 saltos
it4: armazena uma tupla da tabela de topologia
it5: armazena uma tupla da tabela de tuplas associadas
link_tuple: ponteiro que faz referência para o endereço da variável it2
nb_tuple: ponteiro que aponta para tuplas da tabela de vizinhos a 1 salto
nb2hop_tuple: ponteiro que faz referência para o endereço da variável it3
ok: variável de controle
topology_tuple: ponteiro que faz referência para o endereço da variável it4
tuple: ponteiro que faz referência para o endereço da variável it5

Funções Utilizadas no Cálculo da Tabela de Roteamento

Function Void *delete();*
// apaga uma entrada de rota da tabela de roteamento e libera a memória
Function ***OLSR_PD_rt_entry**** *lookup*(**nsAddr** *dest*)
// Procura uma entrada na tabela de rotas para um destino específico. Retorna a
// entrada de rota para o destino dest, se este for encontrado.
Function int_8b *status()*
//Retorna o status do link do nodo atual com o nodo vizinho
Function ***nsAddr*** *get_main_addr(**nsAddr** iface_addr)*
//Retorna o endereço da principal interface de um nodo
Function ***OLSR_PD_PD_nb_tuple**** *find_sym_nb_tuple(**nsAddr** main_addr)*
// Verifica se o nodo de endereço *main_addr,* passado como parâmetro é um vizinho
//simétrico a 1 salto
Function ***OLSR_PD_PD_nb_tuple**** *find_nb_tuple(**nsAddr** main_addr,* **int_8b** *willingness)*
//Verifica se o nodo de endereço *main_addr,* é um vizinho a 1 salto e se possui //disposição em cooperar diferente de "Nunca disponível"
Function ***int_16b*** *find_degree_nb2hop_tuple(**nsAddr** nb_main_addr*)
//Retorna o grau de conectividade do nodo vizinho com base nas informações da //tabela de vizinhos a 2 saltos

Function ***int_16b*** *find_degree_topology_tuple(**nsAddr** last_addr)*
//Retorna o grau de conectividade do nodo *T_last_addr* da tabela de topologia
Function ***OLSR_PD_rt_entry**** find(**nsAdrr** dest);

Algoritmo 1 - Cálculo da tabela de roteamento do OLSR_PD

```
Procedure rtable_computation() {
    //1º passo: Função que remove todas as entradas de rotas da tabela de roteamento
    for (OLSR_PD_rt_entry* it ← rtable.inicio; it <> rtable.fim; it++)
    {
        delete(*it);

    }

    //2º passo: Rotas para vizinhos simétricos a 1 salto (h=1) são calculadas.
        //Lê todas as entradas de vizinhos a 1 salto
    for (OLSR_PD_nb_tuple it1 ← nbset.inicio; it1 <> nbset.fim; it1++) {
        OLSR_PD_nb_tuple* nb_tuple ← *it1;
        // Verifica se o status do link com o destino é simétrico
        if (nb_tuple.N_status = SIMETRICO) {
            int_16b grau_1hop;
            for (OLSR_PD_link_tuple it2 ← linkset.inicio; it2 <> linkset.fim; it2++) {
                OLSR_PD_link_tuple* link_tuple ← *it2;
                //verifica se o tempo de validade do link é maior que a hora atual
                if (get_main_addr(link_tuple.L_nb_iface_addr) = nb_tuple.N_nb_main_addr
                            and link_tuple.L_time >= HORA_ATUAL)
                {
                    //Retorna o grau do nodo principal a 1 salto mais 1
                    grau_1hop ← find_degree_nb2hop_tuple(nb_tuple.nb_main_addr)+1;
                    //Cria uma entrada, na tabela de roteamento, para cada vizinho a 1 salto
                    rtable.R_dest_addr      ← link_tuple.L_nb_iface_addr;
                    rtable.R_next_addr      ← link_tuple.L_nb_iface_addr;
                    rtable.R_iface_addr     ← link_tuple.L_local_iface_addr;
                    rtable.R_dist           ← 1;
                    rtable.R_soma_grau      ← grau_1hop;
                }
            }
        }
    }

    //3º passo: Cálculo de rotas para vizinhos a 2 saltos
    for (OLSR_PD_nb2hop_tuple it3 ← nb2hopset.begin; it3 <> nb2hopset.end; it3++) {
        OLSR_PD_nb2hop_tuple* nb2hop_tuple ← *it3;
        booleano ok ← true;
         OLSR_PD_nb_tuple* nb_tuple;

        //verifica se nb_main_addr, da tabela de vizinhos a 2 saltos, é um vizinho a 1 salto
    nb_tuple ← find_sym_nb_tuple(nb2hop_tuple.nb_main_addr);
        if (nb_tuple = NULL)
            ok ← false;
        else {
            //verifica se nb_main_addr é um vizinho a 1 salto e se possui disposição em
        //repassar pacotes, em nome de outros nodos, diferente de "Nunca disponível"
            nb_tuple ← find_nb_tuple(nb2hop_tuple.nb_main_addr, Nunca_disponível);
            if (nb_tuple <> NULL)
                ok ← false;
            else {
```

```
                // Verifica se o nodo a 2 saltos não é também um vizinho a 1 salto
                nb_tuple ← find_sym_nb_tuple(nb2hop_tuple.nb2hop_addr);
                if (nb_tuple <> NULL)
                    ok ← false;
            }
        }

        if (ok) {
            //obtém o endereço do vizinho anterior, na rota, ao nodo a 2 saltos
            OLSR_PD_rt_entry* entry ← lookup(nb2hop_tuple.nb_main_addr);
            int_16b grau_2hop, old_degree, new_degree;
            //obtém o grau de conectividade até o destino dado pela soma entre o grau até o
            //nodo nb2hop_tuple.nb_main_addr mais o grau do nodo a destino a 2 saltos;
            grau_2hop ← entry.R_soma_grau + find_degree_topology_tuple
                    ( nb2hop_tuple.nb2hop_addr);
            //procura na tabela de roteamento se já existe rota para o destino a 2 saltos, se
    //existir é verificado se o grau de conectividade da nova entra de rota é maior ou
    //igual que o grau da rota mais antiga, se a condição for verdadeira a antiga
    //entrada de rota é substituída, senão antiga é mantida.
            OLSR_PD_rt_entry it = lookup(nb2hop_tuple.nb2hop_addr);

            if (it <> NULL)  {
                new_degree ← grau_2hop;
                old_degree ← it.R_soma_grau;
                if (new_degree >= old_degree)  {
                    delete (*it);
                    //Cria uma nova tupla na tabela de roteamento
                    rtable.R_dest_addr      ←  nb2hop_tuple.nb2hop_addr;
                    rtable.R_next_addr      ←  entry.R_next_addr;
                    rtable.R_iface_addr     ←  entry.R_iface_addr;
                    rtable.R_dist           ←  2;
                    rtable.R_soma_grau      ←  new_degree;
                }
            }
            //se não existe entrada de rota para o destino analisado, apenas insere a rota
            else {
                rtable.R_dest_addr      ←  nb2hop_tuple.nb2hop_addr;
                rtable.R_next_addr      ←  entry.R_next_addr;
                rtable.R_iface_addr     ←  entry.R_iface_addr;
                rtable.R_dist           ←  2;
                rtable.R_soma_grau      ←  grau_2hop;
            }
        }
    }

    // 4º passo: busca os destinos a mais de 2 saltos, até que não seja encontrado mais nodos
    for (int_32b h ← 2;     ; h++) {
        booleano added ← false;

        // cada entrada da tabela de topologia é analisada
        for (OLSR_PD_topology_tuple it4←topologyset.begin; it4<>topologyset.end; it4++)
    {
            OLSR_PD_topology_tuple* topology_tuple ← *it4;
            //Procura na tabela de rotas se existe R_dest_addr igual a T_dest_addr
            OLSR_PD_rt_entry* entry1 ← lookup(topology_tuple.T_dest_addr);
            //Procura na tabela de rotas se existe R_dest_addr igual a T_last_addr
            OLSR_PD_rt_entry* entry2 ← lookup(topology_tuple.T_last_addr);
            int_16b grau_nodo, old_degree, new_degree;
            //Se existir na tabela de roteamento R_dest_addr igual a T_dest_addr da tabela de
```

```
        //topologia, mas se a distância R.dist for diferente de h+1 então pula para a
    //próxima iteração na tabela de topologia
            if (entry1 <> NULL and entry1.R_dist <> h+1)  {
                    continue;
            }
            //Se existir na tabela de roteamento R_dest_addr igual a T_dest_addr, da tabela de
        //topologia, cuja distância R.dist seja igual a h+1, e se existir na tabela de
    //roteamento  R_dest_addr igual a T_last_addr, da tabela de topologia, cuja
    //distância R.dist seja igual a h, então cria uma nova entrada de rota para o
    //destino T_dest_addr e compara o grau de conectividade da nova entrada com o
    //grau da antiga entrada;
            else if (entry1 <> NULL and entry1.R_dist = h+1 and entry2 <> NULL and
                    entry2.R_dist = h)
            {
                    grau_nodo ← entry2.R_soma_grau + find_degree_topology_tuple
                                                    (topology_tuple.T_dest_addr);
                    //procura na tabela de roteamento se já existe rota para o destino
                ////T_dest_addr saltos, se existir é verificado se o grau de conectividade da
            //nova entra de rota é maior ou igual que o grau da rota mais antiga, se a
            //condição for verdadeira a antiga entrada de rota é substituída, senão
        //antiga é mantida.
                    new_degree ← grau_nodo;
                    old_degree ← entry1.R_soma_grau;

                    if (new_degree >= old_degree){
                        delete (*entry1);
                        //Cria uma nova tupla na tabela de roteamento
                        rtable.R_dest_addr          ←  topology_tuple.T_dest_addr;
                        rtable.R_next_addr          ←  entry2.R_next_addr;
                        rtable.R_iface_addr←  entry2.R_iface_addr;
                        rtable.R_dist               ←  h+1;
                        rtable.R_soma_grau          ←  new_degree;
                    }
                ////added ← true;
            }
            //Se não existir na tabela de roteamento R_dest_addr igual a T_dest_addr, da
        //tabela de topologia, e se existir na tabela     de roteamento  R_dest_addr igual a
    ////T_last_addr, cuja distância R.dist seja igual a h, então cria uma nova entrada de
    //rota para o    destino T_dest_addr
            else if (entry1 = NULL and  entry2 <> NULL and entry2.R_dist = h) {

                    grau_nodo ← entry2.R_soma_grau +
                            find_degree_topology_tuple(topology_tuple.T_dest_addr);
                    rtable.R_dest_addr      ←  topology_tuple.T_dest_addr;
                    rtable.R_next_addr      ←  entry2.R_next_addr;
                    rtable.R_iface_addr     ←  entry2.R_iface_addr;
                    rtable.R_dist           ←  h+1;
                    rtable.R_soma_grau      ←  grau_nodo;
                    added ← true;
            }
        }

    // 5º passo: verifica se existe placas de redes adicionais no nodos
    for (OLSR_PD_iface_assoc_tuple it5 ← ifaceassocset.begin; it5 <> ifaceassocset.end;        it5+
+)  {
        OLSR_PD_iface_assoc_tuple* tuple ← *it5;
        OLSR_PD_rt_entry* entry3 ← lookup(tuple.I_main_addr);
        OLSR_PD_rt_entry* entry4 ← lookup(tuple.I_iface_addr);
        int_16b grau_nodo;
```

```
        if (entry3 <> NULL and entry4= NULL) {
            grau_nodo ← entry3.R_soma_grau + find_degree_topology_tuple
                                                (entry3.R_next_addr);
            rtable.R_dest_addr      ←  tuple.I_iface_addr;
            rtable.R_next_addr      ←  entry3.R_next_addr;
            rtable.R_iface_addr     ←  entry3.R_iface_addr;
        rtable.R_dist           ←  entry3.R_dist;
            rtable.R_soma_grau      ←  grau_nodo;
            added ← true;
          }
        }
        if not(added)
            // sai do laço de repetição "for" porque não foi encontrado mais nodos destinos
            break;
    }
}
```

APÊNDICE C - *Script* para extração de informações de arquivo trace

```
#!/bin/sh
mkdir -pv cbr

echo ".....Aguarde!! Isto pode demorar alguns minutos....."
cat *.trt | sed 's/\[//g' | sed 's/\]//g' | sed 's/\_//g' | sed 's/\:/ /g'> cbr/elimina.tr

########################## 1 - PACOTES GERADOS##########################
egrep "^s.*AGT.*" cbr/elimina.tr > cbr/1.0_pacotes_gerados_AGT_Camada_aplicacao.tr
cat cbr/1.0_pacotes_gerados_AGT_Camada_aplicacao.tr  | awk -F" " '{print $6}' > cbr/gerados_AGT.tt
      sort cbr/gerados_AGT.tt | uniq -d > cbr/pacotes_gerados_AGT_DUPLICADOS.tr
      sort cbr/gerados_AGT.tt | uniq > cbr/pacotes_gerados_AGT_UNICOS.tr
      sort cbr/gerados_AGT.tt | uniq > cbr/gerados_AGT_1.tt

#################### 2 - PACOTES GERADOS NA CAMADA DE ##################
egrep "^s.*RTR.*cbr.*" cbr/elimina.tr > cbr/2.0_pacotes_gerados_RTR_Menos_drop_NRTE.tr
cat cbr/2.0_pacotes_gerados_RTR_Menos_drop_NRTE.tr | awk -F" " '{print $6}' > cbr/gerados_RTR.tt
      sort cbr/gerados_RTR.tt | uniq -d > cbr/pacotes_gerados_RTR_NRTE_DUPLICADOS.tr
      sort cbr/gerados_RTR.tt | uniq > cbr/pacotes_gerados_RTR_UNICOS.tr
      sort cbr/gerados_RTR.tt | uniq > cbr/gerados_RTR_1.tt

########################## 3 - PACOTES RECEBIDOS#######################
egrep "^r.*AGT.*" cbr/elimina.tr > cbr/recebidos.tr
cat cbr/recebidos.tr | awk -F" " '{if ($3==$16) {print }}' > cbr/4.0_pacotes_recebidos.tr
cat cbr/4.0_pacotes_recebidos.tr  | awk -F" " ' {print $6}' > cbr/recebidos.tt
      sort cbr/recebidos.tt | uniq -d > cbr/pacotes_recebidos_DUPLICADOS.tr
      sort cbr/recebidos.tt | uniq > cbr/pacotes_recebidos_UNICOS.tr
      sort cbr/recebidos.tt | uniq > cbr/recebidos_1.tt
      sort cbr/recebidos.tt | uniq > cbr/recebidos_2.tt

######################### 4 - PACOTES DROPADOS#########################
egrep "^D.*cbr.*" cbr/elimina.tr > cbr/drop_TOTAL_CBR.tr
cat cbr/drop_TOTAL_CBR.tr | awk -F" " '{if ($3==$14) {if($1=="D"){print }}}' > |
      cbr/drop_ORIGEM_GERAL.tr
      cat cbr/drop_ORIGEM_GERAL.tr | awk ' {print $6}' > cbr/drop_origem_1.tt
      sort cbr/drop_origem_1.tt | uniq > cbr/drop_NoORIGEM_UNICOS.tr
      sort cbr/drop_origem_1.tt | uniq > cbr/drop_origem.tt
      sort cbr/drop_origem_1.tt | uniq > cbr/drop_origem_ENVIADOS.tt

      cat cbr/drop_TOTAL_CBR.tr | awk -F" " '{if ($3==$14) {if($5=="NRTE") {print}}}' > cbr/drop_NRTE.tr
      cat cbr/drop_TOTAL_CBR.tr | awk -F" " '{if ($3==$14) {if($4=="IFQ") {print}}}' > cbr/drop_IFQ.tr
      cat cbr/drop_TOTAL_CBR.tr | awk -F" " '{if ($3==$14) {if($5=="CBK") {print}}}' > cbr/drop_CBK.tr
      cat cbr/drop_TOTAL_CBR.tr | awk -F" " '{if ($3==$14) {if($5=="LOOP") {print}}}' > cbr/drop_LOOP.tr

      cat cbr/drop_TOTAL_CBR.tr | awk -F" " '{if ($3!=$14) {print}}' > cbr/drop_OUTROS.tr
            cat cbr/drop_OUTROS.tr | awk ' {print $6}' > cbr/drop_outros_1.tt
            sort cbr/drop_outros_1.tt | uniq > cbr/drop_outros.tt
            sort cbr/drop_outros_1.tt | uniq > cbr/drop_OUTROS_unicos.tr

      cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="NRTE") {print}}' > cbr/drop_OUTROS_NRTE.tv
            cat cbr/drop_OUTROS_NRTE.tv | awk ' {print $6}' > cbr/drop_outros_NRTE.tt
            sort cbr/drop_outros_NRTE.tt | uniq > cbr/drop_OUTROS_NRTE_unicos.tr
      cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="LOOP") {print}}' > cbr/drop_OUTROS_por_LOOP.tv
            cat cbr/drop_OUTROS_por_LOOP.tv | awk ' {print $6}' > cbr/drop_outros_LOOP.tt
            sort cbr/drop_outros_LOOP.tt | uniq > cbr/drop_OUTROS_por_LOOP_unicos.tr
      cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="TTL") {print}}' >  cbr/drop_OUTROS_por_TTL.tv
            cat cbr/drop_OUTROS_por_TTL.tv | awk ' {print $6}' > cbr/drop_outros_TTL.tt
            sort cbr/drop_outros_TTL.tt | uniq > cbr/drop_OUTROS_por_TTL_unicos.tr
      cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="TOUT") {print}}' > cbr/drop_OUTROS_por_TOUT.tv
            cat cbr/drop_OUTROS_por_TOUT.tv | awk ' {print $6}' >  | cbr/drop_outros_TOUT.tt
            sort cbr/drop_outros_TOUT.tt | uniq > cbr/drop_OUTROS_por_TOUT_unicos.tr
      cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="CBK") {print}}' > cbr/drop_OUTROS_por_CBK.tr
```

```
        cat cbr/drop_OUTROS_por_CBK.tr | awk ' {print $6}' > cbr/drop_outros_CBK.tt
        sort cbr/drop_outros_CBK.tt | uniq > cbr/drop_OUTROS_por_CBK_unicos.tr
    cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="ARP") {print}}' > cbr/drop_OUTROS_por_ARP.tv
        cat cbr/drop_OUTROS_por_ARP.tv | awk ' {print $6}' > cbr/drop_outros_ARP.tt
        sort cbr/drop_outros_ARP.tt | uniq > cbr/drop_6.7.1_OUTROS_por_ARP_unicos.tr
    cat cbr/drop_OUTROS.tr | awk -F" " '{if ($4=="IFQ") {if ($5=="---") {print}}}' > |
    cbr/drop_OUTROS_IFQ.tv
        cat cbr/drop_OUTROS_IFQ.tv | awk ' {print $6}' > cbr/drop_outros_IFQ.tt
        sort cbr/drop_outros_IFQ.tt | uniq > cbr/drop_OUTROS_por_IFQ_unicos.tr
    cat cbr/drop_OUTROS.tr | awk -F" " '{if ($5=="END") {print}}' > cbr/drop_OUTROS_por_END.tv
        cat cbr/drop_OUTROS_por_END.tv | awk ' {print $6}' > cbr/drop_outros_END.tt
        sort cbr/drop_outros_END.tt | uniq > cbr/drop_OUTROS_por_END_unicos.tr

    cat cbr/drop_TOTAL_CBR.tr  | awk -F" " '/^D/ {print $6}' > cbr/drop_CBR.tt
        sort cbr/drop_CBR.tt | uniq > cbr/drop_01_TOTAL_CBR_UNICOS.tr
        sort cbr/drop_CBR.tt | uniq > cbr/drop_unicos.tt
        sort cbr/drop_CBR.tt | uniq > cbr/drop_unicos_1.tt

    cat cbr/drop_TOTAL_CBR.tr  | awk -F" " '/^D/ {print $3 " "$6}' > cbr/drop_repetidos_duplo.tt
        sort cbr/drop_repetidos_duplo.tt  | uniq > cbr/drop_repetidos_duplo_unico.tt

########################## 5 - PACOTES FORWARD##########################
egrep "^f" cbr/elimina.tr > cbr/pct_forward_total_eventos.tr
    cat cbr/pct_forward_total_eventos.tr | awk -F" " '/^f/ {print $3 " " $6}' > cbr/forward_duplo.tt
        sort cbr/forward_duplo.tt | uniq > cbr/forward_duplo_1.tt
        sort cbr/forward_duplo.tt | uniq > cbr/forward_duplo_2.tt

###################### 6 - PACOTES DE CONTROLE GERADOS###################
egrep "^s.*OLSR.*" cbr/elimina.tr > cbr/pacotes_CONTROLE_Gerados.tr
    cat cbr/pacotes_CONTROLE_Gerados.tr | awk -F" " 'BEGIN{soma=0}; {soma += $8} END{print soma          " |
    TOTAL_bytes_pacotes_CONTROLE"}' > cbr/total.txt

########################### 7 - RESUMO ##############################
cat cbr/pacotes_gerados_RTR_UNICOS.tr >> cbr/recebidos_1.tt
cat cbr/pacotes_gerados_RTR_UNICOS.tr >> cbr/drop_unicos.tt
cat cbr/drop_01_TOTAL_CBR_UNICOS.tr >> cbr/recebidos_2.tt
cat cbr/drop_repetidos_duplo_unico.tt  >> cbr/forward_duplo_1.tt

sort cbr/recebidos_1.tt | uniq -d > cbr/repetidos_enviados_E_recebidos.tr
sort cbr/recebidos_1.tt | uniq -u > cbr/repetidos_enviados_NAO_recebidos.tr
sort cbr/drop_unicos.tt| uniq -d > cbr/repetidos_enviados_E_drop.tr
sort cbr/drop_unicos.tt | uniq -u > cbr/repetidos_enviados_NAO_drop.tr
sort cbr/recebidos_2.tt | uniq -d > cbr/repetidos_dropado_MAS_recebidos.tr
    cat cbr/repetidos_dropado_MAS_recebidos.tr >> cbr/drop_origem.tt
        sort cbr/drop_origem.tt | uniq -d > cbr/drop_NoOrigem_MAS_recebidos.tt
        sort cbr/drop_origem.tt | uniq -d > cbr/drop_1_NoOrigem_MAS_recebidos_1.tt
    cat cbr/repetidos_dropado_MAS_recebidos.tr >> cbr/drop_outros.tt
        sort cbr/drop_outros.tt | uniq -d > cbr/drop_Outros_MAS_recebidos.tr
    cat    cbr/drop_Outros_MAS_recebidos.tr >> cbr/drop_NoOrigem_MAS_recebidos.tt
        sort cbr/drop_NoOrigem_MAS_recebidos.tt | uniq -d >
    cbr/drop_MAS_receb_repetidos_OUTROS_ORIGEM.tt
    cat cbr/drop_MAS_receb_repetidos_OUTROS_ORIGEM.tt >>
    cbr/drop_1_NoOrigem_MAS_recebidos_1.tt
        sort cbr/drop_1_NoOrigem_MAS_recebidos_1.tt | uniq -u >
    cbr/drop_1_NoOrigem_MAS_recebidos.tr
        sort cbr/drop_1_NoOrigem_MAS_recebidos_1.tt | uniq -u >
    cbr/drop_1_NoOrigem_PARA_ENVIADOS.tt
    cat cbr/repetidos_dropado_MAS_recebidos.tr >> cbr/drop_unicos_1.tt
        sort cbr/drop_unicos_1.tt | uniq -u > cbr/drop_TOTAL_CBR_UNICOS_MENOS_drop_MAS_recebidos.tr

###################### 8 - PACOTES REALMENTE ENVIADOS ###################
cat cbr/pacotes_gerados_AGT_UNICOS.tr >> cbr/drop_origem_ENVIADOS.tt
    sort cbr/drop_origem_ENVIADOS.tt | uniq -u > cbr/ENVIADOS_gerados_RTR_MENOS_drop_NoOrigem.tr

########################### 9 - FORWARD ############################
sort cbr/forward_duplo_1.tt  | uniq -d > cbr/forward_EXCLUIDOS.tt
```

```
        cat cbr/forward_EXCLUIDOS.tt >> cbr/forward_duplo_2.tt
sort cbr/forward_duplo_2.tt | uniq -u > cbr/pacotes_forward_BEM_SUCEDIDOS.tr
        cat cbr/pacotes_forward_BEM_SUCEDIDOS.tr | awk '{print $2}' > cbr/forward_unico_filtrado.tt
sort cbr/forward_unico_filtrado.tt | uniq > cbr/pacotes_forward_UNICOS.tr

######################## 10 - PACOTES PERDIDOS - LOSS ######################
cat cbr/elimina.tr  | awk -F" " '/^[rfD].*cbr.*/{print $6}' > cbr/todos_unicos.tt
cat cbr/gerados_AGT.tt >> cbr/todos_unicos.tt
sort cbr/todos_unicos.tt | uniq -u > cbr/pacotes_PERDIDOS.tr

################ 11 - REMOÇÃO DE ARQUIVOS NÃO MAIS NECESSÁRIO ###############
rm cbr/elimina.tr cbr/recebidos.tr #cbr/*.tt
wc -l cbr/*.tr > RESUMO_CBR1.xls
        cat cbr/total.txt >> RESUMO_CBR1.xls
        cat RESUMO_CBR1.xls | sed "s/^ *//g" > RESUMO_CBR.xls
rm RESUMO_CBR1.xls     # elimina o arquivo de colunas não alinhadas
rm -R cbr

echo "script desenvolvido por Gunnar Ramos gunnar.ramos@gamil.com."
```