Universidade Federal da Paraíba
Centro de Ciências Exatas e da Natureza
Departamento de Informática
Programa de Pós-Graduação em Informática

Daniel Charles Ferreira Porto

MLSD: Um Protocolo de Divulgação de Estados dos Enlaces para Redes em Malha Sem Fio Infraestruturadas

João Pessoa – Paraíba
Março de 2010

Daniel Charles Ferreira Porto

MLSD: Um Protocolo de Divulgação de Estados dos Enlaces para Redes em Malha Sem Fio Infraestruturadas

Dissertação de Mestrado apresentada à banca examinadora do Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba, como requisito parcial para a obtenção do título de Mestre em Informática.

Orientador: Glêdson Elias da Silveira

João Pessoa – Paraíba
Março de 2010

P853m Porto, Daniel Charles Ferreira.
MLSD: um protocolo de divulgação de estados dos enlaces para redes em malha sem fio infraestruturadas / Daniel Charles Ferreira Porto. - - João Pessoa : [s.n.], 2010.
89 f.
Orientador: Glêdson Elias da Silveira.
Dissertação (Mestrado) – UFPB/CCEN.

1.Informática. 2.Protocolo de roteamento. 3.Redes em malha sem fio - RMSF. 4.Escalabilidade.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado do DANIEL CHARLES FERREIRA PORTO, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 26 de março de 2010.

Aos vinte e seis dias do mês de março do ano dois mil e dez, às nove horas, no Auditório do Centro de Ciências Exatas e da Natureza da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Computação Distribuída*", o Sr. Daniel Charles Ferreira Porto. A comissão examinadora composta pelos professores doutores: Glêdson Elias da Silveira (DI - UFPB), Orientador e Presidente da Banca Examinadora, Guido Lemos de Souza Filho (DI-UFPB) e Nelson Souto Rosa (UFPE), como examinador externo. Dando início aos trabalhos, o Prof. Glêdson Elias da Silveira, cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "*MLSD: Um Protocolo de Divulgação de Estados dos Enlaces para Redes em Malha Sem Fio Infraestruturadas*". Concluída a exposição, o candidato foi argüido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, a professora Tatiana Aires Tavares, Sra. Coordenadora do PPGI, lavrou a presente ata, que vai assinada por ela, e pelos membros da Banca Examinadora. João Pessoa, 26 de março de 2010.

Tatiana Aires Tavares

Prof. Dr. Glêdson Elias da Silveira
Orientador (DI-UFPB)

Prof. Dr. Guido Lemos de Souza Filho
Examinador Local (DI-UFPB)

Prof. Dr. Nelson Souto Rosa
Examinador Externo (UFPE)

MLSD: Um Protocolo de Divulgação de Estados dos Enlaces para Redes em Malha Sem Fio Infraestruturadas

RESUMO

O aumento da popularidade e da demanda das redes sem fio ao longo dos últimos anos tem levado a novos desafios que estimulam o desenvolvimento da tecnologia na busca de melhores padrões e protocolos. Neste contexto, surgem as Redes em Malha Sem Fio - RMSF (*Wireless Mesh Networks* – WMN), que são redes de baixo custo, que tem a capacidade de se organizar e se configurar automaticamente, de fácil implantação e com capacidade de tolerância a falhas. As RMSFs buscam alcançar melhor desempenho, maior confiabilidade e flexibilidade quando comparadas a outras redes sem fio. Estas redes podem fornecer uma infraestrutura formada por dispositivos dedicados à tarefa de roteamento, com seu posicionamento planejado para obter cobertura satisfatória, proporcionando conectividade dentro da área de cobertura para os clientes móveis. Por isto, estas redes precisam se adaptar a alterações de topologia, que podem ocorrer a todo o momento. Para permitir esta adaptação é necessário um protocolo de roteamento. Um componente importante de um protocolo de roteamento proativo é um protocolo de descoberta de topologia da rede, no qual as modificações na topologia da rede são divulgadas através de mensagens de atualização. Entretanto, protocolos proativos possuem uma alta carga de mensagens. Uma carga elevada de mensagens tem grande impacto no desempenho do protocolo de roteamento e podem levar a problemas de escalabilidade. Neste contexto, a principal contribuição deste trabalho é a construção de um protocolo de divulgação dos estados dos enlaces com garantia de entrega, chamado *Mesh Network Link State Dissemination Protocol* - MLSD, projetado para RMSFs, cujos objetivos são reduzir a carga de mensagens de atualização na rede e reduzir o total de mensagens enviadas para divulgar as atualizações.

Palavras-chave: Redes em Malha sem Fio, escalabilidade, protocolo de roteamento, topologia e infraestrutura.

MLSD: Um Protocolo de Divulgação de Estados dos Enlaces para Redes em Malha Sem Fio Infraestruturadas

ABSTRACT

The increasing popularity and demand of wireless networks, over the past few years, has led to new challenges that stimulate the development of technology to build better standards and protocols. In such a context arises the Wireless Mesh Networks (WMN) which are low cost, self-organized and self-configurable networks that are easy to deploy and are fault tolerant. The WMN are aiming to achieve better performance, greater reliability and flexibility when compared to other wireless networks. These networks can provide an infrastructure consisting of dedicated devices for routing, strategically placed to achieve adequate coverage, providing connectivity within the coverage area for mobile clients. Therefore, these networks need to adapt to topology changes that can occur at any time. To allow this adjustment a routing protocol is required. An important component of a proactive routing protocol is a network topology discovery protocol, which disseminates link-state updates messages over the network. The proactive protocols usually have a high message overhead. However, a high message overhead has a major impact on the performance of a routing protocol and can lead to scalability problems. In such a context, the main contribution of this work is to build a reliable link-state dissemination protocol, called MLSD specially designed for RMSF whose goals are to reduce the messages overhead and reduce the total messages sent to advertise topology changes.

Keywords: Wireless Mesh Networks, scalability, routing protocol, topology and infrastructure.

Tenho-vos dito estas coisas, para que em mim tenhais paz. No mundo tereis tribulações; mas tende bom ânimo, eu venci o mundo.

Jesus, em João 16:33

Agradecimentos

Se vi mais longe foi por estar de pé sobre ombros de gigante. Isaac Newton

Agradeço a Deus, por ter me proporcionado a força, a determinação, a sabedoria e a paciência de que eu precisei para lutar pelos meus objetivos e realizá-los, em todos os momentos da minha vida.

Aos meus pais, José Petrônio (*in memorian*) e Maria das Dores, pela educação, formação de caráter e integridade que tenho hoje, e que mesmo na ausência ou na distância, me serviram de estímulo para não desanimar diante das dificuldades e sempre seguir em frente, rumo à vitória.
À minha querida Juliana, por estar sempre ao meu lado e dar um apoio fundamental em todos os momentos, me trazendo muita felicidade, a quem dedico meu amor.
Aos meus irmãos, Daianne, Diego, Danielle, Denise e Débora; aos meus sobrinhos, Maria Júlia, Laura, Miguel, Eduardo, Marina, Maria Clara e Yasmim; aos meus cunhados André Luís, Elierton e Adilson e à minha sogra Maria José, pelas alegrias, apoio e pela maravilhosa família.
Aos meus avós, João e Raimunda Nascimento, pelos quais sinto grande admiração, por serem a base estrutural da nossa família e servirem como referência de retidão.

Ao meu inestimável e saudoso amigo Gustavo Cavalcanti (*in memorian*), a quem eu devo uma enorme gratidão, não só pela parceria, mas principalmente pela confiança, cumplicidade, lealdade e consideração na nossa sincera e inesquecível amizade; e à sua mãe D. Waleska Cavalcante, exemplo de fortaleza, por estar sempre presente. Uma verdadeira lição de força e vida de um irmão de coração, grande ser humano, com quem tive a honra de estudar e trabalhar dividindo momentos felizes, de superação e também muito difíceis, mas sem nunca perder a fé em Deus. *Hasta la muerte, Siempre!*

Ao Prof. Gledson, que acompanhou durante o mestrado, pelos valiosos ensinamentos recebidos, fonte de confiança e responsabilidade. À Lula e Cida, amigos que se tornaram a família do trabalho, a quem tenho uma enorme gratidão e apreço. Também aos amigos do PPGI UFPB, Ramos, Prof. Valéria, por me ajudarem com os processos do mestrado, e Sr. João Mariano, pelo pronto atendimento em resolver os problemas de fornecimento de energia durante a fase crítica das simulações. Aos membros da banca examinadora Guido Lemos, Nelson Souto Rosa pelas importantes sugestões para a versão final desta dissertação.

Aos meus amigos, Luís Antônio, Vital Filho, Aralinda Nogueira, Danielle Bezerra, Antônio Carlos, João Paulo, Jean Felipe, Mateus, Joedson, Uirá, Katherine e à equipe do COMPOSE, por terem me confortado e estimulado nos momentos difíceis e com os quais também compartilhei os momentos alegres.

À Escola Superior de Redes da RNP, renomada instituição na qual tive oportunidade em poder trabalhar, lecionar, aprender e progredir cada vez mais durante o mestrado.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AODV	*Ad Hoc on-demand distance vector*
IDE	*Integrated Development Environment*
IEEE	*Institute of electrical and electronics engineers*
IFS	*802.11 Interframe Space*
IP	*Internet protocol*
IWMRA	*Infrastructured wireless mesh architecture*
MAC	*Media access control*
MANET	*Mobile Ad Hoc network*
MAP	*Mesh access point*
MC	*Mesh client*
MLSD	*Mesh Network Link State Dissemination Protocol*
MP	*Mesh point*
MPP	*Mesh portal point*
MPR	*Multipoint relay*
MR	*Mesh router*
OLSR	*Optimized link state routing*
QoS	*Quality of service*
RA-OLSR	*Radio Aware Optimized Link State Routing*
RM-AODV	*Radio metric Ad Hoc on demand distance vector*
RMSF	Rede em Malha Sem Fio
SNDP	*Scalable Neighborhood Discovery Protocol*
STA	*Station*
UFPB	Universidade Federal da Paraíba
WMN	*Wireless mesh network*

Sumário

Capítulo 1

Introdução

As redes sem fio têm vivenciado um aumento de popularidade ao longo dos últimos anos e, mais recentemente, as tecnologias de WLAN (*Wireless Local Area Network*) tornaram-se meio comum de acesso à Internet para computação móvel. O baixo custo das interfaces sem fio tem permitido o desenvolvimento de uma variedade de dispositivos portáteis que utilizam estas tecnologias, proporcionando acesso ubíquo às pessoas. Esta crescente demanda tem levado a novos desafios que estimulam o desenvolvimento da tecnologia na busca de melhores padrões e protocolos.

A rápida difusão do acesso via IEEE 802.11 [Ieee 2007] e o crescimento da demanda por cobertura, têm levado a um aumento do número de pontos de acesso instalados em ambientes domésticos, pequenas empresas e vários cenários de *hotspots* (locais onde existem pontos de acesso disponíveis), ainda que com cobertura limitada e servindo a poucos usuários. Atualmente, existe um considerável interesse na expansão das redes 802.11 para cenários corporativos de larga escala, para prover acesso de banda larga a um grande número de usuários [Faccin 2006; Zhang 2006].

As redes sem fio podem operar em duas formas: no modo infraestruturado e no modo *Ad Hoc [Royer 1999]*. No modo infraestruturado, a comunicação entre os dispositivos com interface sem fio, chamados nós, deve ser realizada através de um ponto de acesso (*Access Point* - AP) fixo, com posicionamento planejado para prover cobertura aos dispositivos móveis associados. O ponto de acesso é o responsável por encaminhar todos os dados trocados entre os clientes associados, ou entre os clientes e a rede fixa na qual o AP esteja conectado. Mesmo nas situações em que dois dispositivos móveis possam se comunicar diretamente, todo o tráfego obrigatoriamente deve passar pelo ponto de acesso.

O modo *Ad Hoc* é o modo mais simples e permite que os dispositivos possam se comunicar diretamente, se estiverem no raio de alcance, sem a presença de uma infraestrutura

de rede prévia ou de estações fixas. As redes *Ad Hoc* também são conhecidas como MANETs (*Mobile Ad Hoc NETworks*) [Cordeiro 2002; Johnson 2007; Macker 1998].

As redes *Ad Hoc* (Figura 1) são caracterizadas por não dependerem de infraestrutura preestabelecida ou administração centralizada e são compostas de dispositivos como laptops e PDAs. Nas redes *Ad Hoc*, os nós são móveis e livres para se moverem enquanto se comunicam com outros nós, operando de maneira *peer-to-peer*, agindo como um roteador independente formando uma rede temporária e dinamicamente organizada quando os nós entram na área de cobertura um do outro. Assim, os nós podem se comunicar com nós fora do raio de alcance através da cooperação dos vizinhos. Devido a esta mobilidade, a topologia da rede sem fio pode mudar rapidamente e de forma imprevisível [Basagni 2004; Macker 1998].

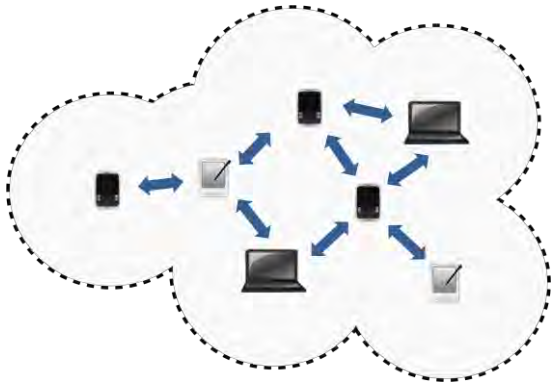


Figura 1 Mobile *Ad Hoc* Network

Com essas características, as redes *Ad Hoc* permitem sua aplicabilidade em diversos cenários [Basagni 2004] como campos de batalhas, emergência em locais de desastres, aplicações educacionais, mas também em feiras de eventos, canal de retorno de TV Digital [Miguel Elias 2007], ou ainda, onde o custo ou o tempo de implantação de uma infraestrutura convencional seja muito elevado.

Como uma promessa de futuro das tecnologias de redes sem fio surge as Redes em Malha Sem Fio - RMSF (*Wireless Mesh Network - WMN*) [Akyildiz 2005; Zhang 2006]. As RMSFs buscam alcançar melhor desempenho, maior confiabilidade e flexibilidade quando comparada a outras redes sem fio, por isto, vêm atraindo a atenção dos pesquisadores, como se percebe pelo surgimento de publicações e desenvolvimento de padrões como o IEEE 802.11s relacionados ao tema [Bahr 2006].

Muito parecidas com uma rede *Ad Hoc*, nas RMSFs também existem nós móveis capazes de se organizar e configurar dinamicamente, mas, diferentemente das redes *Ad Hoc*, também pode existir um conjunto de nós fixos dedicados a rotear os pacotes da rede,

formando um *backbone*. Estas redes possuem capacidade de tolerância a falhas, visto que podem se ajustar à perda de alguns nós, com os nós da rede automaticamente estabelecendo e mantendo conectividade entre si [Akyildiz 2005; Zhang 2006].

Assim como nas redes *Ad Hoc*, nas RMSFs a conectividade entre os nós fora do raio de alcance é obtida com cooperação dos nós da rede através de múltiplos saltos entre a origem e o destino. Para que os roteadores sejam capazes de encaminhar o tráfego entre os nós, as informações necessárias ao roteamento devem ser divulgadas e mantidas entre os nós da rede, e um protocolo de roteamento deve ser usado para este fim.

Em razão das similaridades entre as RMSF e *Ad Hoc*, tem sido muito comum a aplicação de protocolos de roteamento desenvolvidos para redes *Ad Hoc* em RMSF [Akyildiz 2005; Albuquerque 2006]. Por exemplo, o projeto VMesh [Tsarmpopoulos 2005] utiliza o protocolo OLSR [Clausen 2003; Jacquet 2001]. Outro exemplo de aplicação de protocolos para redes *Ad Hoc* em RMSF é o *Microsoft Mesh Networks* [Chen 2006], que é construído baseado no DSR [Johnson 2004] modificando a métrica para calcular os custos dos enlaces.

Entretanto, conforme [Akyildiz 2005], a presença do *backbone* fixo e sem restrições de consumo de energia (pois os nós podem ser conectados diretamente a rede elétrica ao invés de baterias), permite o desenvolvimento de protocolos de roteamento mais eficazes projetados para melhor aproveitar as características das RMSF. Por isto, apesar da grande disponibilidade de protocolos para redes *Ad Hoc*, ainda são necessárias novas propostas e pesquisas para projetos de protocolos de roteamento para redes em malha sem fio em cenários de redes domésticas com acesso de banda larga à Internet e redes corporativas sem fios.

1.1. Protocolos de Roteamento para Redes Sem Fio

Para divulgar as informações na rede, os protocolos de roteamento podem usar duas estratégias [Myung Jong 2006]: vetor de distância ou estado de enlace.

Os protocolos de roteamento que divulgam vetores de distância funcionam mantendo informações dos custos dos enlaces (distância) para cada destino em uma tabela de roteamento (um vetor) e publicando para seus vizinhos que, com isto, podem incluir os destinos aos quais só podem ser acessados através do nó que enviou a tabela. Em função deste processamento, estes protocolos levam bastante tempo para convergir [Tanenbaum 2003], isto é, para que as atualizações alcancem toda a rede. Um exemplo de protocolo com esta abordagem é o RIP [Hedrick 1988].

Os protocolos de roteamento que divulgam estados dos enlaces funcionam coletando informações sobre o estado da conectividade de seus vizinhos, que são enviadas para cada um dos roteadores da rede. Desta forma, cada roteador pode montar uma tabela com toda a topologia da rede e computar os melhores caminhos para cada destino [Tanenbaum 2003]. Um exemplo de protocolo que usa esta abordagem é o OSPF [Moy 1998].

Os protocolos de roteamento para redes sem fio também podem ser classificados observando a forma de manutenção das informações: proativos, reativos e híbridos. Os protocolos proativos tentam manter informações de roteamento consistentes e atualizadas, para cada nó da rede, através da divulgação de mensagens de atualização em resposta à mudanças na topologia da rede [Royer 1999]. O modo de emissão das mensagens pode ser orientado a eventos, quando uma mensagem é emitida se alguma modificação for detectada, ou orientado a tempo, quando a mensagem é emitida em intervalos de tempo regulares [Myung Jong 2006].

Os protocolos proativos tem a vantagem de conhecer uma rota para um destino antes de iniciar a comunicação. A desvantagem destes protocolos é que eles tendem a gerar uma alta carga de mensagens de roteamento [Campos 2005], pois os nós continuamente emitem mensagens, mesmo quando não participam de alguma comunicação. A alta carga de mensagens dos protocolos proativos também impõe problemas de escalabilidade [Chen 2006], pois à medida que a rede cresce, mais nós emitem mensagens. São exemplos de protocolos proativos o DSDV [Perkins 1994], OLSR [Jacquet 2001] e WPR [Campista 2008c].

Os protocolos reativos criam rotas somente quando requeridas pelo nó de origem. Desta forma, quando um nó necessita de uma rota para um destino, ele inicia um processo de descoberta de rota na rede. Quando a rota é estabelecida, um processo de manutenção é iniciado até que algum nó intermediário ao longo do caminho, ou o destino, se torne inacessível ou a rota não ser mais necessária [Royer 1999].

A vantagem dos protocolos reativos é a economia dos recursos da rede, pois os nós só emitem mensagens de roteamento quando precisam se comunicar, evitando divulgações desnecessárias de informações. Entretanto, dado que o processo de descoberta da rota é realizado somente sob demanda, as rotas não estão imediatamente disponíveis [Campos 2005]. Desta forma, os protocolos reativos sofrem um retardo para enviar os dados, até que se complete o processo de descoberta de rota. São exemplos de protocolos reativos o AODV [Perkins 1999], e o DSR [Johnson 2004].

Existem ainda protocolos híbridos que fazem uso de ambas as estratégias proativa e reativa e tentam obter o melhor de ambas as abordagens [Zhang 2006], tais como: ZRP [Haas 1997], DYMO [Chakeres 2006], e o HWMP [Bahr 2006].

Em face destas características, a escolha do protocolo de roteamento mais eficaz está relacionado ao tipo de cenário de aplicação, por exemplo, um protocolo pode ser mais adaptado aos cenários com muita ou pouca mobilidade ou considerar um retardo mínimo para inicio da comunicação.

1.2. Motivação e Objetivos do Trabalho

Embora os protocolos de roteamento desenvolvidos para redes *Ad Hoc* funcionem em RMSF, eles não consideram as características das RMSF e, por isto, tem desempenho muito ruim. Por exemplo, os protocolos proativos desenvolvidos para redes *Ad Hoc* geralmente realizam a divulgação das mensagens de roteamento periodicamente e sem garantia de entrega. Conforme já mencionado, o envio periódico de mensagens eleva a carga de roteamento gerando problemas de escalabilidade. Além disto, a perda dessas mensagens gera tabelas de roteamento incompletas. Assim, existe a necessidade de protocolos mais eficientes adaptados às características das RMSF [Akyildiz 2005; Zhang 2006].

Conforme mencionado, nas RMSF pode existir um conjunto de nós fixos destinados a rotear os pacotes formando um *backbone*. Segundo [Zhang 2006], protocolos de roteamento proativos são melhores adaptados a RMSF com nós fixos. Desta forma, não existe latência de obtenção de rota para um nó que sempre está no mesmo lugar, como *gateways* e roteadores.

Além disso, também podem existir nós com mobilidade (clientes). Neste caso, as modificações da vizinhança podem ser frequentes e precisam ser informadas ao *backbone*, para que as rotas sejam corrigidas. A abordagem de divulgação dos estados dos enlaces tem como vantagem a rápida convergência quando comparada aos protocolos de vetor de distância, alcançando mais rapidamente todos os nós da rede.

Um componente importante de um protocolo de roteamento baseado em estado de enlace é um protocolo de descoberta de topologia da rede (*Network Topology Discovery Protocol* – NTDP) no qual atualizações na topologia da rede são divulgadas através de mensagens chamadas *Link-state Advertisement* (LSA) [Nezhad 2008]. Os protocolos podem divulgar as LSAs periodicamente ou quando for percebida alguma atualização dos enlaces, isto é, orientada a eventos [Zou 2002].

Por um lado, os protocolos que divulgam LSA periodicamente podem deixar rotas incorretas durante o intervalo de atualização. Se este intervalo for longo, o tempo para que as mensagens alcancem todos os nós da rede também aumenta. Se este intervalo for muito curto, as rotas são corrigidas mais rapidamente, mas, provocam um aumento da carga de mensagens, consumindo muitos recursos da rede.

Por outro lado, os protocolos que divulgam LSA orientados a eventos respondem mais rapidamente a mudanças na topologia, pois a mensagem é enviada no momento em que o evento ocorre. No entanto, nós móveis podem provocar eventos frequentes o que leva a um aumento da carga de mensagens, consumindo recursos da rede.

Neste contexto, esta dissertação propõe o protocolo MLSD (*Mesh Network Link State Dissemination Protocol*), um protocolo de divulgação dos estados dos enlaces com garantia de entrega, projetado para RMSF que tem como objetivos:

- Reduzir o total de mensagens de atualização de topologia emitidas na rede, através de uma estratégia proativa, orientada a eventos e empregando uma estratégia controle de envio para evitar sobrecarga de mensagens.
- Reduzir a carga de mensagens enviadas, através de uma estratégia de divulgação incremental de atualizações, da adoção de um formato compacto das LSA, juntado informações sempre que possível e eliminando informações desnecessárias.

Com isto, busca-se um protocolo mais escalável e robusto, através uso efetivo infraestrutura fornecida pelas RMSF, emprego de uma estratégia de entrega confiável e uso eficiente recursos da rede.

1.3. Organização do Trabalho

Esta dissertação está organizada em sete capítulos, incluindo esta introdução, divididos da seguinte maneira: a descrição das Redes em Malha Sem Fio é realizada no Capítulo 2. O Capítulo 3 mostra os trabalhos relacionados destacando a estratégia de controle da topologia. O Capítulo 4 descreve a especificação do protocolo MLSD detalhando cada uma das suas estratégias. No Capítulo 5, é apresentado o ambiente de simulação que destaca as ferramentas e parâmetros utilizados no trabalho. O Capítulo 6 apresenta os resultados da avaliação de desempenho e no Capítulo 7 são realizadas as considerações finais e delineado os trabalhos futuros.

Capítulo 2

Redes em Malha Sem Fio

As Redes em Malha Sem Fio (RMSF) são redes de baixo custo, que tem a capacidade de se organizar e configurar automaticamente, de fácil implantação, com capacidade de tolerância a falhas, isto é, de se adaptar a perda de alguns nós [Akyildiz 2005].

As RMSF vêm suprir uma necessidade dos usuários de aplicações militares e civis, com maior largura de banda e acesso à Internet [Bruno 2005; Zhang 2006]. Outras aplicações de RMSF podem ser em segurança pública (polícia, bombeiros, guardas rodoviários, câmeras de vigilância), cobertura para redes locais sem fio (*WLAN*), em ambientes fechados (*Indoor*), como escritórios e laboratórios, ou abertos (*Outdoor*) [Sichitiu 2005], permitindo acesso ubíquo aos usuários móveis e fixos [Bruno 2005; Zhang 2006].

A arquitetura da RMSF introduz uma hierarquia dos dispositivos que a compõe [Hossain 2008]. Nesta arquitetura, existem dispositivos dedicados com funções específicas de roteamento chamados *mesh routers* (MRs) e, também, existem os clientes que podem ou não executar o roteamento, chamados *mesh clients* (MCs). Dependendo das funcionalidades dos dispositivos, as RMSF podem ser classificadas em três grupos: Infraestruturada/*Backbone*, Cliente ou Híbrida [Akyildiz 2005]

O MLSD, protocolo para disseminação de mensagens de topologia proposto nesta dissertação, foi desenvolvido especialmente para a RMSF Infraestruturada, utilizando as características deste tipo de RMSF para divulgar eficientemente os eventos ocorridos na rede. Por isto, neste capítulo será apresentado uma descrição das RMSF, mostrando as arquiteturas existentes, as características destas redes e os principais problemas e desafios envolvidos.

2.1 Arquitetura das RMSF

Conforme citado, a arquitetura das RMSF é composta por dois tipos de nós: *Mesh Routers* (MRs) e *Mesh Clients* (MCs), e podem ser classificadas em três grupos, de acordo

com o papel exercido pelos nós: Infraestruturada/*Backbone*, Cliente ou Híbrida [Akyildiz 2005].

De maneira geral, os MRs são dispositivos responsáveis por realizar o roteamento na rede e geralmente possuem mais recursos computacionais. Estes MRs não possuem limitações quanto ao consumo de energia, pois geralmente estão fixos em locais providos de alimentação elétrica, com seu posicionamento planejado para obter cobertura satisfatória, formando uma infraestrutura sem fio (*backbone mesh*) [Zhang 2006].

Os MRs também possuem a capacidade de atuar como *gateway* para a integração com outras RMSF, outros tipos de redes, ou ainda, a *Internet*. Em adição, os MRs podem ser equipados com mais de uma interface de rede sem fio funcionando em diferentes canais, para melhorar a flexibilidade da rede [Akyildiz 2005].

Os MCs são dispositivos com recursos computacionais limitados (ex: baterias e processamento), que geralmente possuem uma única interface de rede sem fio e são portáteis como laptops, PDAs, telefones celulares e assim por diante. Os MCs acessam a rede através dos MRs e também podem usar outros MCs para encaminhar seus pacotes, isto é, também podem executar a função de roteadores [Akyildiz 2005]. As seções seguintes apresentam um detalhamento de cada um dos grupos de RMSF (Infraestruturado, Cliente e Híbrido), destacando o papel exercido pelos MRs e MCs em cada grupo.

2.1.1 RMSF Infraestruturada

A rede em malha sem fio infraestruturada (*Infrastructure/backbone Wireless Mesh Network*) é a mais comum. Nela, os *Mesh Routers* formam um *backbone* sem fio (Figura 2), responsável por uma área de cobertura para conectar os *Mesh Clients*. Os *Mesh Routers* também podem exercer funções de *gateway*, conectando a RMSF à *Internet*, ou com outras redes já existentes. Nesta arquitetura os clientes não encaminham pacotes de outros nós.

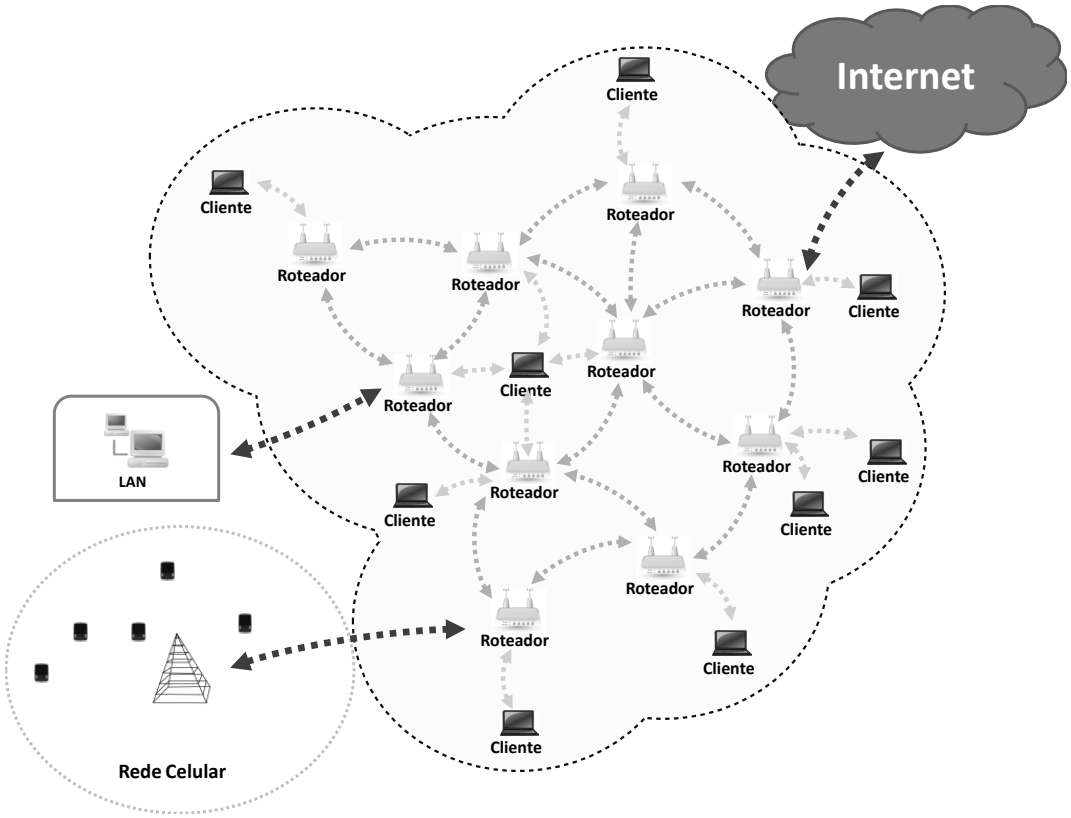


Figura 2 Arquitetura de RMSF Infraestruturada

Uma aplicação para as RMSF infraestruturadas é a cobertura de grandes áreas, como um bairro ou campus universitário, onde MRs instalados nos telhados de casas ou prédios formam o *backbone* que permita acesso à *Internet* aos clientes. Como exemplo desta aplicação existe o projeto *roofnet* [Bicket 2005]. A Figura 3 demonstra esta aplicação, destacando inclusive a capacidade de tolerância a falhas de uma RMSF. Nela, podemos ver que a ocorrência de uma falha de um *link* em uma das casas, provoca uma reconfiguração na rede permitindo encaminhamento dos pacotes por um caminho alternativo (tracejado).

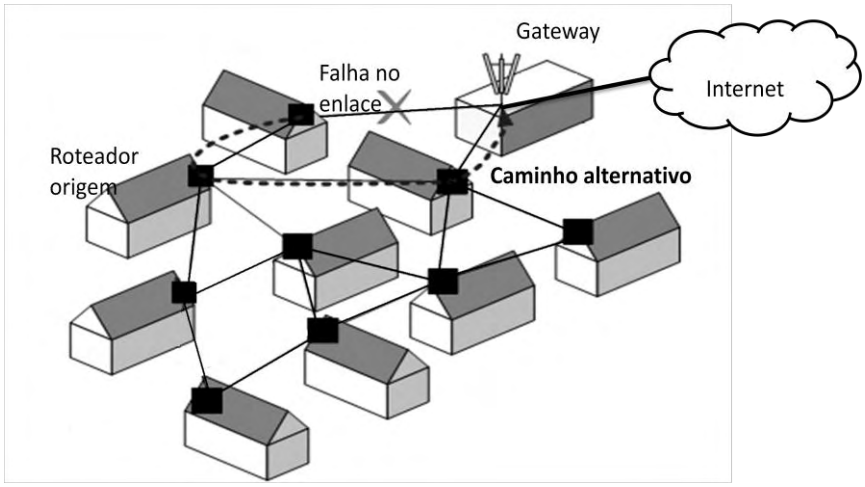


Figura 3 Infraestrutura de uma RMSF em um bairro

2.1.2 RMSF Cliente

Na RMSF Cliente (*Client Wireless Mesh Network*), os MCs formam uma rede *peer-to-peer* (Figura 4), e devem ser capazes de realizar tarefas de configuração e roteamento para si e para os outros, tal como uma rede *Ad Hoc* convencional. Neste tipo de RMSF não existe MR e cada pacote destinado a um nó na rede salta de cliente em cliente, até alcançar o destino.

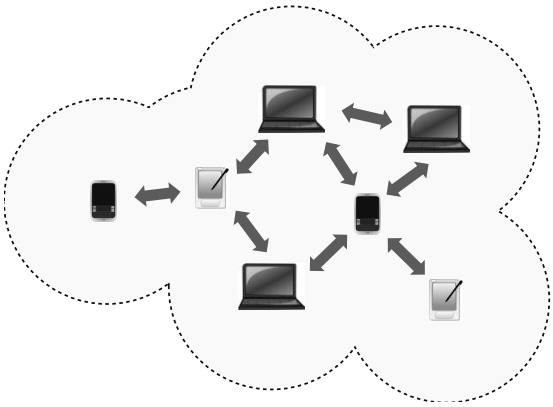


Figura 4 Arquitetura de RMSF Cliente

2.1.3 RMSF Híbrida

A RMSF Híbrida é uma combinação da RMSF infraestruturada e da RMSF cliente. Os MCs podem tanto acessar a rede através dos MRs como através de outros MCs, enquanto os MRs proveem acesso à *Internet* e as redes com outras tecnologias (Figura 5), além de melhorar a cobertura de conectividade dentro do *backbone mesh*.

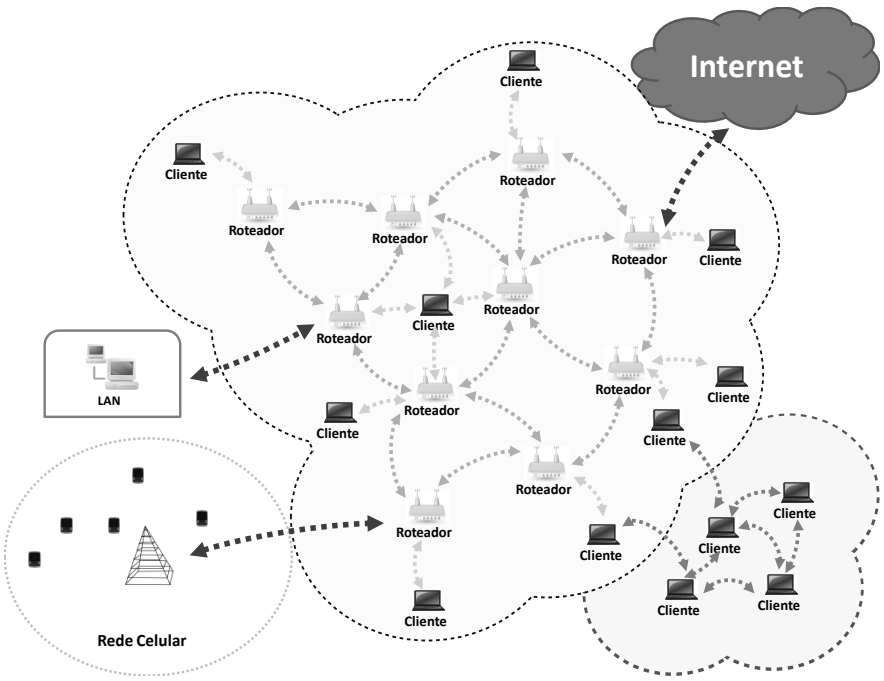


Figura 5 Arquitetura Híbrida de RMSF

2.2 Características

A presença de um *backbone*, juntamente com a utilização de protocolos apropriados, conferem às RMSF algumas características gerais [Akyildiz 2005; Held 2005]:

- **Autoconfiguração –** Os nós de uma RMSF devem utilizar protocolos que permitam descobrir seus vizinhos e as rotas para os outros nós da rede [Held 2005].
- **Confiabilidade –** Com a presença de um *backbone* estático, os clientes podem acessar mais do que um roteador para se comunicar com algum outro nó da rede. Esta característica é importante para os clientes móveis ou para o caso de falha em algum roteador [Held 2005; Jun 2003].
- **Interoperabilidade –** Uma das características das RMSF é a capacidade de se conectar a outros tipos de redes sem fio existentes [Akyildiz 2005], como por exemplo, rede de telefonia celular, WiMax [Andrews 2007] ou Zigbee [Zigbee 2008].
- **Múltiplos tipos de acesso –** Como as RMSFs podem ser integradas com outras redes existentes, a comunicação dos nós de uma RMSF pode ser interna à rede (comunicação ponto-a-ponto) ou através dos outros tipos de rede que estejam conectadas à RMSF [Akyildiz 2005].

2.3 IEEE 802.11s

Conforme citado, novos padrões e protocolos vêm sendo desenvolvidos para RMSF. Em 2003, o *Institute of Electronics and Electrical Engineering* (IEEE) criou o grupo "S", para especificação de um padrão para RMSF. Atualmente, o padrão 802.11s possui um rascunho (*draft*) na versão 4.0 e está em processo de revisão dos comentários [802.11s-Tg 2008].

O padrão 802.11s define uma arquitetura para a RMSF similar à arquitetura infraestruturada definida em [Akyildiz 2005]. A Figura 6 ilustra a arquitetura da RMSF e seus componentes para o 802.11s [Bahr 2006]:

- ***Station* (STA) –** Corresponde a todo nó que utiliza a rede e, mais especificamente, algum nó que não tenha capacidade de acessar diretamente a RMSF, isto é, um cliente legado que não execute o protocolo de roteamento.
- ***Mesh Point* (MP) –** é uma STA que tem capacidade de acessar diretamente o *backbone mesh*, isto é, pode executar o protocolo de roteamento e encaminhar pacotes dentro da rede. Os MPs podem ser desde dispositivos clientes, como laptops, até dispositivos de infraestrutura, como roteadores [Bahr 2007].

- ***Mesh Access Point* (MAP) –** São MPs com funcionalidade de *Access Point(AP)*, isto é, este tipo de nó faz a conexão dos STAs atuando como um *AP* das redes 802.11.
- ***Mesh Portal Point* (MPP) –** São MPs com funcionalidade de *gateway* ou *bridge*, fazendo a interconexão com outras redes IEEE 802, como por exemplo, a Ethernet.

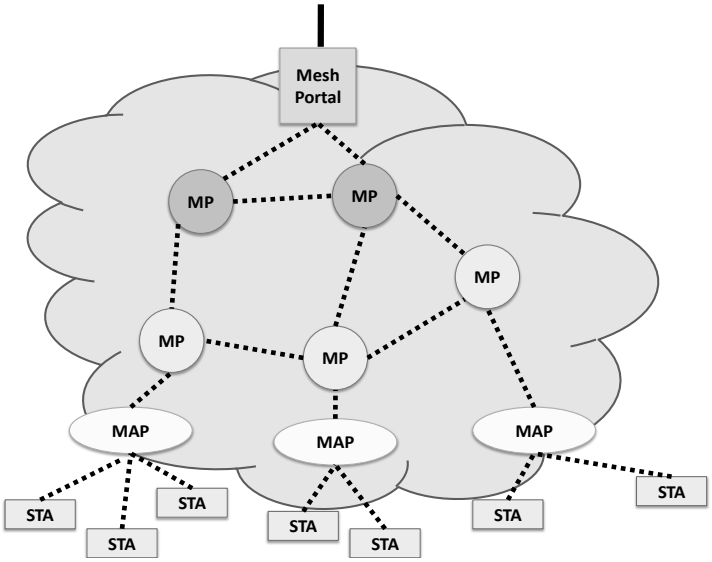


Figura 6 Arquitetura de uma rede em malha sem fio 802.11s

O 802.11s também inclui os padrões 802.11e [Mangold 2002] e 802.11i [802.11i-Tg 2004]. O padrão 802.11e adiciona suporte à qualidade de serviço (*Quality of Service* - QoS) na camada MAC para as redes sem fio. Já o 802.11i define mecanismos de segurança para as redes sem fio.

O 802.11s padroniza uma métrica de roteamento denominada *Airtime Routing Metric* [Bahr 2006], que reflete a utilização do canal para a transmissão de um quadro através de um determinado enlace [Zhang 2006]. Essa métrica leva em consideração o tamanho do quadro, a taxa de erros, a carga de mensagens e a taxa de transmissão dos enlaces envolvidos.

Além da métrica padronizada, nas primeiras versões da especificação 802.11s, foi definido um protocolo de roteamento padrão e outro opcional: o HWMP (*Hybrid Wireless Mesh Protocol*) [Bahr 2006] e o RA-OLSR (*Radio Aware Optimized Link State Routing Protocol*) [Bahr 2006]. Ambos operam utilizando informações da camada de enlace.

Atualmente o RA-OLSR foi retirado do 802.11s em favor de um *framework* de extensão que permita selecionar outras implementações de protocolos e métricas [Carrano 2009]. Neste *framework* de extensão, através de mensagens especiais de anúncio (*beacons*), emitidas pelos MP, é informado aos nós da rede o protocolo de roteamento e a métrica utilizada [Bahr 2006].

Desta forma, enquanto assegura a interoperabilidade, obrigando qualquer dispositivo em conformidade com o padrão 802.11s a implementar o protocolo HWMP com a métrica *Airtime*, também permite evolução e adaptação do protocolo com métricas que possam descrever melhor um determinado cenário. Os protocolos RA-OLSR e HWMP serão descritos nas Seções 3.2 e 3.4 respectivamente.

2.4 Desafios em RMSF

As RMSF são uma promessa de futuro para prover serviços de acesso sem fio, ubíquo, escalável e de banda larga à Internet. Entretanto, ainda existem muitos desafios em se construir uma rede sem fio de larga escala e alto desempenho. Conforme [Akyildiz 2005; Bruno 2005; Zhang 2006], alguns destes desafios podem ser entendidos em forma de camada, da camada física à de aplicação.

Na camada física, a aplicação de sistemas MIMO (*Multiple input multiple output*) tem sido realizada para melhorar a eficiência das redes sem fio. O MIMO consiste no envio de várias réplicas do sinal em frequências diferentes, através de múltiplas antenas no transmissor para um mesmo número de antenas no receptor. De maneira simplificada, esta técnica baseia-se no fato de que sinais emitidos por antenas independentes têm atenuação diferente. Assim, existe uma alta probabilidade de que pelo menos um sinal correto possa ser recebido. O MIMO, é atualmente, uma das principais tecnologias do 802.11n.

As comunicações entre os nós utilizando múltiplas interfaces sem fio apresentam desafios tanto para os protocolos de enlace quanto aos protocolos de roteamento. Desta forma, os protocolos da MAC precisam ser projetados para utilizar mais eficientemente múltiplas antenas operando em múltiplos canais.

Na camada de rede, para explorar a capacidade alcançada pelos potenciais avanços das camadas física e MAC, os protocolos de roteamento também precisam se preocupar em encontrar rotas de melhor qualidade de acordo com as condições de rede.

Protocolos de roteamento para redes que utilizam comunicação através de vários saltos costumam escolher a melhor rota em termos da menor quantidade de saltos, tempo de vida da rota ou nível de energia dos dispositivos, em detrimento da qualidade do enlace. Por isto, é um desafio o desenvolvimento de métricas mais eficazes que considere a qualidade do enlace para um melhor desempenho fim a fim. Por exemplo, uma métrica que leve em consideração a utilização do canal no qual os nós estão se comunicando pode ajudar o protocolo na definição de rotas que melhorem a vazão da rede [Campista 2008a].

Outra característica importante para um protocolo de roteamento é o uso efetivo do *backbone* criado pelos nós fixos da RMSF, buscando o consumo mínimo de recursos para configurar os caminhos e ter capacidade de se corrigir rapidamente.

A camada de transporte representa outro grande desafio. À medida que aumenta a quantidade de saltos, aumenta também a variação do RTT (*Round-trip time*), que faz o desempenho do TCP degradar rapidamente. Além disto, a perda de pacotes, colisões, falhas de enlace e assimetria da rede também contribuem para degradação do desempenho do TCP. Por isto, os protocolos de transporte precisam ser refinados para serem aplicados de maneira eficaz nas RMSF.

Um dos cenários de aplicação das RMSF é fornecer acesso de banda larga à Internet. Portanto, para que aplicações comuns da Internet, como VoIP e *Streaming Video*, funcionem bem, é necessário que as RMSF suportem requisitos de qualidade de serviço (QoS). Assim, outras métricas como *delay* (retardo), *jitter* (variação do retardo), e vazão por nó, também precisam ser consideradas pelos protocolos de comunicação.

Somando-se aos desafios citados, ainda é necessário bastante esforço de desenvolvimento na área de segurança, interoperabilidade e facilidade de uso. Um detalhamento dos principais desafios das RMSF pode ser encontrado em [Bruno 2005], [Akyildiz 2005] e [Zhang 2006].

2.5 Considerações Finais

Neste capítulo foi apresentada a descrição das redes em malha sem fio, expondo a arquitetura, os principais grupos, características e desafios existentes. Torna-se essencial conhecer a arquitetura, os principais grupos e as características das RMSF para compreender o MLSD, protocolo de divulgação de mensagens de topologia apresentado nesta dissertação.

Dentre os desafios apresentados neste capítulo, o MLSD se propõe a melhorar a camada de rede, usando efetivamente o *backbone* e buscando reduzir o consumo de recursos (total de mensagens e carga das mensagens), através de uma nova estratégia de divulgação orientada a eventos, com garantia de entrega de mensagens e um formato compacto das mensagens de atualização de topologia.

Capítulo 3

Trabalhos Relacionados

Para melhor contextualização deste trabalho, neste capítulo, serão apresentados alguns protocolos de roteamento que estão sendo usados em RMSF, destacando as estratégias adotadas na topologia para que possa ser contrastada com a estratégia do MLSD que será proposta. Os protocolos que serão mostrados têm características híbridas e proativas. Os protocolos reativos não mantêm informações de topologia e por este motivo não serão descritos.

Segundo [Chen 2006], devido a RMSF ser uma área de pesquisa bastante nova, muitos projetos existentes ainda utilizam protocolos desenvolvidos para redes *Ad Hoc*, em especial, as estratégias como as utilizadas pelo AODV[Perkins 1999] e OLSR[Jacquet 2001].

O OLSR é um protocolo proativo, desenvolvido para redes *Ad Hoc*, cujo processo de controle da topologia é bem definido e consiste da escolha de um conjunto de nós, através de uma heurística, responsáveis por emitir mensagens de controle periodicamente. O OLSR também é usado em RMSF e seu funcionamento é base para alguns protocolos de roteamento desenvolvidos para RMSF (ex. RA-OLSR), por isto, sua descrição detalhada será realizada na Seção 3.1 e a descrição do RA-OLSR na Seção 3.2.

Além disso, como uma aplicação comum para RMSF é o acesso à *Internet*, é muito usual que nestes cenários as rotas para *Internet* sejam as mais utilizadas. Por isto, mesmo protocolos híbridos desenvolvidos especificamente para RMSF, geralmente utilizam a parte proativa para informar o *backbone* a respeito das rotas para *Internet*. São exemplos de protocolos projetados para RMSF que utilizam esta abordagem o WPR [Campista 2008b] e o HWMP [Bahr 2006].

O WPR [Campista 2008b] tem seu funcionamento baseado no OLSR. A diferença é que o WPR possui um algoritmo de controle de inundação (*flooding*) e utiliza uma heurística

própria para calcular um conjunto de nós encaminhadores, responsáveis por divulgar as informações da rede, chamados AMPRs. O protocolo WPR será detalhado na Seção 3.3.

O HWMP [Bahr 2006] é um protocolo híbrido, cuja parte reativa é baseada no protocolo AODV, que é um protocolo desenvolvido para redes *Ad Hoc*. A diferença é que o HWMP é implementado na camada de enlace enquanto o AODV é implementado na camada de rede.

Outra diferença do HWMP é o comportamento proativo para divulgar as rotas para alguns nós. Além disso, o HWMP usa uma métrica diferente do AODV para selecionar o melhor caminho. Esta métrica, chamada *Airtime*, leva em consideração características mais sensíveis do meio de transmissão a rádio (*radio-aware*). O protocolo HWMP será detalhado na Seção 3.4.

Os protocolos WPR, RA-OLSR e HWMP (parte proativa) possuem estratégias de controle de topologia que possibilitam a comparação com o MLSD. Contudo, para a avaliação de desempenho que será realizada no capítulo 6, a comparação do MLSD será realizada somente com o OLSR, porque não foi encontrada nenhuma implementação disponível do RA-OLSR, WPR e HWMP para o simulador utilizado.

3.1 Protocolo OLSR

O OLSR (*Optimized Link State Routing Protocol*) [Clausen 2003] é um protocolo de roteamento proativo, baseado em estado de enlace, que utiliza uma estratégia de divulgação de mensagens periódicas para atualizar as informações da topologia da rede.

A estratégia de divulgação dos estados dos enlaces convencional é realizada com uma inundação na rede (*flooding*). Nesta estratégia, todo nó que recebe uma mensagem deve repassá-la uma única vez, ou seja, todos os nós da rede são encaminhadores. O OLSR reduz a quantidade de mensagens destinada à atualização dos estados dos enlaces através de uma estratégia otimizada para escolha dos nós encaminhadores.

Cada nó escolhe um conjunto de nós, dentre os seus vizinhos, para que sejam os responsáveis por propagar mensagens de atualização dos estados dos enlaces. Os nós que propagam as mensagens em benefício dos outros são chamados MPRs (*Multipoint Relays*).

A escolha dos MPRs é realizada com as informações obtidas da vizinhança do nó. Para um nó determinar quais são seus vizinhos, o OLSR envia periodicamente, em *broadcast*,

mensagens *HELLO*. As mensagens *HELLO* não são repassadas por nenhum nó. Quando um nó recebe um *HELLO,* ele adiciona o emissor da mensagem como sendo seu vizinho.

Cada mensagem *HELLO* carrega consigo uma lista com todos os vizinhos do nó. Desta forma, um nó A que recebe uma mensagem *HELLO* de um nó B, toma conhecimento de todos os vizinhos em comum com o nó B, e também todos os vizinhos que o nó A pode alcançar através do nó B, isto é, vizinhos a dois saltos do nó A. Os MPRs do nó A são escolhidos entre os vizinhos do nó A que podem alcançar todos os vizinhos a dois saltos de A.

As mensagens *HELLO* também servem para identificar quais enlaces são simétricos. Isto ocorre quando um nó que recebe uma mensagem *HELLO* encontra seu próprio endereço na lista de vizinhos enviada. Assim, o nó sabe que o emissor do *HELLO* também tem a capacidade de ouvi-lo. Apenas os nós com enlaces simétricos podem ser escolhidos MPRs.

Quando um nó A escolhe um nó B como MPR, ele passa a informar na sua mensagem *HELLO* quem são seus MPRs. Assim, o nó B ao receber uma mensagem *HELLO* do nó A, sabe que foi escolhido MPR de A e passa a divulgar, periodicamente, as mensagens de atualização dos estados dos enlaces, chamadas no OLSR de mensagens de controle de topologia (*Topology Control* - TC).

Através das mensagens TC os nós recebem as informações da topologia da rede e podem, usando um algoritmo de menor caminho, computar a tabela de roteamento. Todas as rotas calculadas possuem um tempo de expiração definido. A medida que as mensagens TC chegam, as rotas são atualizadas. A métrica de roteamento utilizada pelo OLSR é a quantidade de saltos (*hops*) até o destino.

3.1.1 Divulgação da Topologia no OLSR

Conforme mencionado, o controle na divulgação das mensagens TC é realizado utilizando *Multipoint Relays* (MPRs). As mensagens TC são emitidas em *broadcast*, sem garantia de entrega, mas apenas os MPRs do nó emissor repassam as TCs para seus próprios MPRs, e assim sucessivamente na rede. Cada nó escolhe independentemente seu conjunto de MPRs. A estratégia de divulgação de mensagens de controle de topologia (TC) apenas pelos MPRs de um nó reduz o total de mensagens na rede.

Os MPRs são escolhidos entre os vizinhos que permitem conhecer a topologia a dois saltos. Por exemplo, na Figura 7 o nó central possui oito vizinhos e através das mensagens *HELLO* emitidas por quatro deles (com hachura), o nó central toma conhecimento de todos os

vizinhos a até dois saltos. Desta forma, na visão do nó central, os quatro vizinhos são escolhidos MPRs.

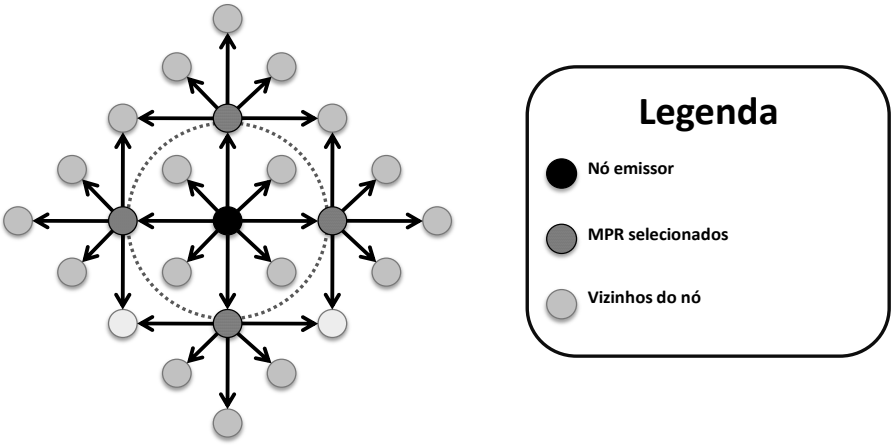


Figura 7 divulgação de mensagens de *broadcast* pelos MPRs de um nó

Um nó que foi selecionado como MPR passa a publicar periodicamente as mensagens de controle de topologia (TC). Estas mensagens são encapsuladas no pacote do OLSR. O pacote do OLSR pode carregar qualquer mensagem relacionada ao protocolo (ex: *TC* ou *HELLO*), e permite o transporte de várias mensagens em um único pacote.

O pacote do OSLR é encapsulado utilizando o protocolo UDP e IP. Por padrão, os MPRs divulgam os estados dos seus enlaces a cada 5 segundos. Assim as informações dos enlaces do próprio MPR só são reenviadas na rede após este intervalo.

Entretanto, quando um MPR recebe uma mensagem de outro MPR vizinho ele repassa em um intervalo de no máximo 0,5 segundos. Além disto, se durante esse intervalo de 0,5 segundos do repasse forem recebidas mais TCs que o pacote do OLSR pode transportar, outros pacotes são enviados em rajada, isto é, imediatamente. Assim, a medida que aumenta a quantidade de MPRs, aumenta o número de repasses, e aumenta a quantidade de mensagens enviadas em um tempo menor.

3.2 RA-OLSR

O RA-OLSR (*Radio Aware Optimized Link State Routing Protocol*) é uma adaptação do OLSR para o ambiente definido pelo 802.11s [Bahr 2006]. Assim como o HWMP, o RA-OLSR realiza a construção dos caminhos da rede na camada de enlace, utilizando endereços MAC ao invés de endereços IP. Outra adaptação importante é que o RA-OLSR também utiliza uma métrica mais sensível ao meio de transmissão via rádio (*radio-aware*) [Bahr

2006]. Esta informação é propagada através de um campo adicional na mensagem de informação de topologia.

O RA-OLSR usa o mesmo mecanismo de divulgação da topologia do OLSR, utilizando o conjunto de nós *Multipoint Relays* (MPRs). Todos os MP participam do processo de escolha dos MPR. Além disso, o protocolo utiliza a estratégia do *Fisheye State Routing* [Pei 2000] para a divulgação das informações dos estados dos enlaces. Assim, no RA-OLSR, as informações dos nós mais próximos são mais frequentemente divulgadas do que nós mais distantes, manipulando o campo TTL das mensagens.

3.3 WPR

O WPR (*Wireless-mesh-network Proactive Routing*) [Campista 2008b] é um protocolo proativo, baseado em estado de enlace, projetado para as RMSFs. O WPR utiliza um mecanismo de controle de inundação das informações dos estados dos enlaces da rede, adaptando a estratégia de *Multipoint Relays* (MPRs) do OLSR a um determinado cenário de RMSF (detalhado mais a frente). Com isso, o WPR consegue diminuir a quantidade de mensagens de divulgação dos estados dos enlaces que são reencaminhadas na rede.

O cenário para o qual o WPR foi projetado considera que a maior parte do tráfego na RMSF é para Internet. Além disso, o protocolo assume que todos os roteadores da rede conhecem previamente os endereços dos *gateways* para a rede cabeada. Por fim, é definido que o protocolo é executado apenas pelos roteadores do *backbone* da RMSF. Assim, os clientes que desejam se conectar à RMSF usam os nós do *backbone* como *Access Points*.

3.3.1 A Divulgação da Topologia no WPR

O WPR possui um mecanismo de controle de inundação para as informações de topologia. Esse mecanismo de controle define um conjunto de nós responsáveis pelo encaminhamento das mensagens, semelhante ao conjunto de nós MPRs definido pelo OLSR. Estes nós são denominados AMPRs (*Adapted MultiPoint Relays*) e, assim como os MPRs, reduzem o número de mensagens TC na rede.

O conjunto de nós AMPRs diferencia-se do conjunto de nós MPRs do OLSR pelo algoritmo de seleção dos nós AMPRs. Este algoritmo é uma adaptação do algoritmo de seleção dos MPRs para o cenário de uso da rede que é assumido pelo WPR. No cenário de uso assumido pelo WPR a maioria das aplicações da rede são aplicações de acesso à *Internet*.

Assim, o mecanismo de controle de inundação do WPR realiza atualização mais frequente do caminho dos roteadores para os *gateways*.

O algoritmo do mecanismo de inundação controlada define dois tipos de mensagens de divulgação dos estados dos enlaces: as mensagens de inundação controlada e as mensagens de inundação não controlada. Estas mensagens possuem o mesmo formato, mas frequências de publicação diferentes.

Desta forma, a parte da topologia que é mais frequentemente atualizada com as informações de topologia fica semelhante a uma árvore, com a raiz sendo um dos *gateways* para a rede cabeada. A Figura 8 destaca a área que possui maior frequência de atualização das informações de topologia de um roteador R em uma RMSF. Este algoritmo será explicado a seguir.

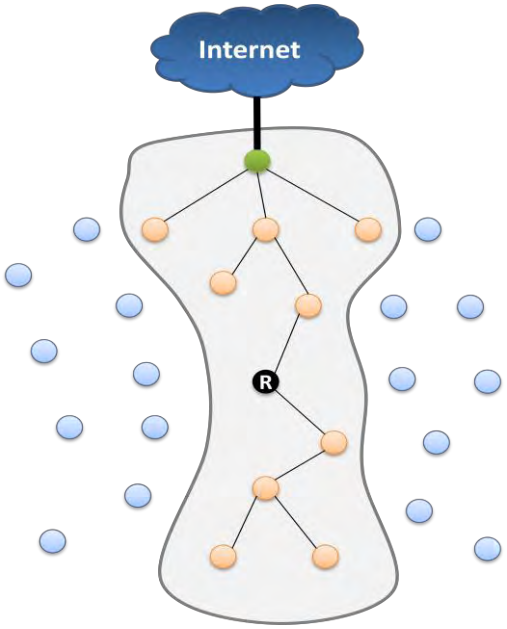


Figura 8 Área com maior frequência de atualização para um roteador R

As mensagens de inundação controlada são publicadas mais frequentemente do que as mensagens de inundação não controlada. Para encaminhamento destas mensagens, o WPR define que todos os AMPRs que receberem uma mensagem de inundação não controlada devem realizar o encaminhamento da mensagem normalmente. É através deste encaminhamento que as informações dos estados dos enlaces do nó emissor podem chegar a todos os roteadores da rede.

Todavia, ao receberem uma mensagem de inundação controlada, os AMPRs devem realizar o encaminhamento apenas caso a mensagem tenha sido enviada por um nó ascendente ou descendente de sua árvore.

A definição dos nós ascendentes de um roteador acontece simplesmente efetuando o cálculo da rota até o *gateway*. Já a definição dos nós descendentes de um roteador acontece através das mensagens trocadas para descoberta de vizinhança.

A descoberta de vizinhança entre os roteadores do WPR acontece através do envio periódico de mensagens *HELLO*. Estas mensagens *HELLO* são similares às mensagens *HELLO* do OLSR. Entretanto, os roteadores do WPR acrescentam para cada vizinho incluído nas mensagens *HELLO*, a informação se este vizinho foi escolhido pelo roteador emissor como ascendente ou não. Estas mensagens *HELLO* permitem que cada roteador defina como seus descendentes os nós que o escolheram como ascendente, possibilitando assim o encaminhamento de mensagens de inundação controlada.

3.4 O Protocolo HWMP

O HWMP (*Hybrid Wireless Mesh Protocol*) [Bahr 2006] é um protocolo híbrido, isto é, possui estratégia reativa e proativa para definição de rotas na rede. Este é o protocolo de roteamento padrão adotado no 802.11s e é utilizado, por exemplo, no projeto "*One Laptop Per Child*" [Bahr 2007].

O HWMP utiliza o RM-AODV (*Radio Metric AODV*) [Aoki 2006] para a definição de rotas de maneira reativa. O RM-AODV é uma modificação do protocolo reativo AODV [Perkins 1999] que constrói os caminhos da rede na camada de enlace, isto é, usa endereços MAC ao invés de IP e utiliza uma métrica *radio-aware* [Aoki 2006].

Na obtenção de rotas, o mecanismo utilizado pelo RM-AODV é similar ao do AODV. Vale salientar que o RM-AODV, diferentemente do AODV, não permite que nós intermediários respondam as requisições de rota, mesmo que possuam uma rota já calculada para o destino. Isto acontece pelo fato da métrica utilizada pelo RM-AODV ser mais sensível ao enlace dos vizinhos do que a métrica de saltos do AODV. Desse modo, respostas de nós intermediários podem gerar valores errados para as métricas dos caminhos [Zhang 2006]. Além disso, o RM-AODV permite que sejam feitas requisições de rotas para múltiplos destinos de uma vez, enquanto o AODV só permite que um único destino seja configurado em cada requisição [Perkins 1999].

Em alguns cenários de uso de RMSF, uma grande quantidade do tráfego é direcionada para um único *Mesh Portal Point* (MPP) [Bahr 2006]. Por exemplo, um MPP que conecte a RMSF à Internet. Neste caso, a parte proativa do HWMP mantém o caminho para este nó.

3.4.1 A Divulgação da Topologia no HWMP

A extensão proativa do HWMP pode ter sua utilização configurada nos *Mesh Points* (MP)/*Mesh Access Points* (MAP) e *Mesh Portal Points* (MPP). Habilitando a extensão proativa, um MPP passa a periodicamente publicar anúncios (*mesh portal announces*) através da rede. Desse modo, uma árvore contendo os caminhos para o MPP raiz é criada. Caso exista mais de um MPP publicando anúncios, apenas um deles pode ser configurado ou selecionado, através de um processo de eleição, como o nó raiz. Através dos anúncios dos MPPs os MP podem ser configurados para operar em dois modos:

- **Sem registro** – é considerado o modo leve da extensão proativa, pois não gera tantas mensagens de controle. Neste modo, os MPs ou MAPs não divulgam mensagens para realizar seu registro no MPP raiz, mas usam anúncios recebidos para criar e atualizar uma entrada para o MPP raiz na sua tabela de roteamento.
- **Com registro** – modo da extensão que gera muitas mensagens de controle. Neste modo, todo MP ou MAP que deseja se comunicar com o MPP raiz deve enviar uma mensagem para realizar seu registro e de suas estações (STAs).

Os dois modos de funcionamento da parte proativa do HWMP possibilitam a escolha da utilização da estratégia híbrida para obtenção de rotas definida pelo protocolo. Caso seja utilizado o modo sem registro, a extensão proativa apenas possibilita que caminhos para o MPP raiz sejam previamente calculados. Isto permite que os MPs possuam rotas previamente definidas para as redes externas ou para nós presentes no caminho para o MPP raiz. As rotas para outros MPs da rede devem ser obtidas através da maneira reativa do protocolo.

A utilização do modo com registro também permite o conhecimento prévio de rotas para o MPP raiz e para os nós pertencentes ao caminho para o MPP raiz. Além disto, permite que o HWMP reduza a latência para obtenção de rotas para qualquer outro MP da rede. Isto ocorre porque quando o MP fonte deseja se comunicar com outro, ele envia dados diretamente para o MPP raiz indicando com qual MP destino deseja se comunicar. Como o MPP raiz possui registro de todos os MPs da rede, ele encaminha a mensagem para o MP destino sinalizando ao MP destino para iniciar o processo de requisição de rotas para o MP origem. Enquanto o processo de obtenção da melhor rota é realizado, os dois MPs (origem e destino)

podem utilizar o caminho através do MPP raiz, reduzindo assim a latência para iniciar a comunicação.

É importante destacar que, com o modo de registro e os anúncios do MPP raiz habilitados o HWMP penaliza a comunicação entre nós móveis dentro do *backbone mesh*, pois, como pode haver quebras mais frequentes do enlace, a comunicação pode prioritariamente usar os caminhos definidos na árvore para a raiz. Deste modo, os caminhos escolhidos podem não ser as melhores rotas, e todo tráfego vai ter que passar pelo portal.

O HWMP é um protocolo cuja especificação ainda está em desenvolvimento. Em face deste estágio de evolução, ideias como suporte (ou não) a múltiplos MPP e seu funcionamento, ainda estão sendo discutidas e não estão especificadas. No presente momento, o que se tem definido é: caso exista mais de um MPP na rede, um processo de eleição é iniciado para que apenas um deles assuma o papel de *gateway*.

3.5 Considerações Finais

Este capítulo apresentou os principais protocolos de roteamento, com características proativas, utilizados em Redes em Malha Sem Fio. Além disto, foi destacado como se dá a estratégia de controle da topologia neles.

Vale destacar que todos os protocolos apresentados neste capítulo realizam a divulgação dos seus estados de enlace de maneira periódica. Isto ocorre porque as mensagens são emitidas sem garantia de entrega. Assim os protocolos esperam que, em algum momento, todos os nós recebam a informação de topologia e desta forma o protocolo consiga manter suas informações de topologia consistentes.

Capítulo 4

O Protocolo MLSD

Este capítulo apresenta a especificação do protocolo MLSD (*Mesh Network Link State Dissemination Protocol*), um protocolo para divulgação dos estados dos enlaces para RMSF infraestruturada. Os principais objetivos do protocolo MLSD são reduzir a carga das mensagens empregadas na atualização das informações da topologia e reduzir o total de mensagens enviadas para divulgar as atualizações.

Para reduzir o total de mensagens de atualização das informações da topologia, o MLSD adota uma estratégia orientada a eventos. Nesta abordagem, uma mensagem é emitida somente quando alguma modificação da topologia for detectada, por exemplo, quando algum *Mesh Client* entra na área de cobertura e é detectado por um *Mesh Router*. Entretanto, quando a movimentação dos *Mesh Clients* aumenta, quantidade de eventos de detecção e perda de enlaces também aumenta, consequentemente, o total de mensagens tende a se elevar. Para tratar este efeito da abordagem orientada a eventos, o MLSD controla o intervalo de tempo entre emissões de mensagens de topologia, buscando reduzir o excesso de mensagens na rede enviadas em um curto intervalo de tempo.

Para reduzir a carga das mensagens enviadas, o MLSD adota uma abordagem incremental na qual são divulgadas, de forma confiável, apenas as atualizações ocorridas na topologia ao invés de divulgar todos os estados dos enlaces, garantindo a consistência das informações e reduzindo a quantidade de dados em cada mensagem. Além disso, o MLSD adota um formato compacto de pacote para transportar as informações, agrupando as mensagens sempre que possível e eliminando informações desatualizadas.

O protocolo MLSD foi desenvolvido para ser utilizado na camada de topologia da arquitetura de roteamento para as RMSF infraestruturada, denominada IWMRA (*Infrastructure Wireless Mesh Routing Architecture*) [Porto 2009]. A IWMRA separa as

funcionalidades do roteamento em três camadas: vizinhança, topologia e roteamento. Desta forma, um protocolo deve ser projetado para tratar os problemas de cada uma das camadas.

O protocolo MLSD assume as características das RMSF infraestruturadas que foram apresentadas no Capítulo 2. Além disso, o MLSD possui também uma premissa sobre os tipos de enlaces utilizados. Sendo assim, os requisitos exigidos para correta operação do protocolo MLSD estão listados abaixo:

- Infraestrutura de *Mesh Routers* posicionado para permitir cobertura total à rede;
- Disponibilidade de uso irrestrito de energia pelos *Mesh Routers*;
- Encaminhamento de mensagens realizado exclusivamente pela infraestrutura de *Mesh Routers*; e
- Os enlaces entre os dispositivos que compõem a rede devem ser bidirecionais.

O cenário de RMSF assumido pelo MLSD e seus componentes pode ser visto na Figura 9, em que se percebe a área de cobertura fornecida pelo *backbone* formada pelos *Mesh Routers* fixos (MR) e que os *Mesh Clients* (MC) podem estar movimentando-se através de uma região de cobertura ou permanecer fixos.

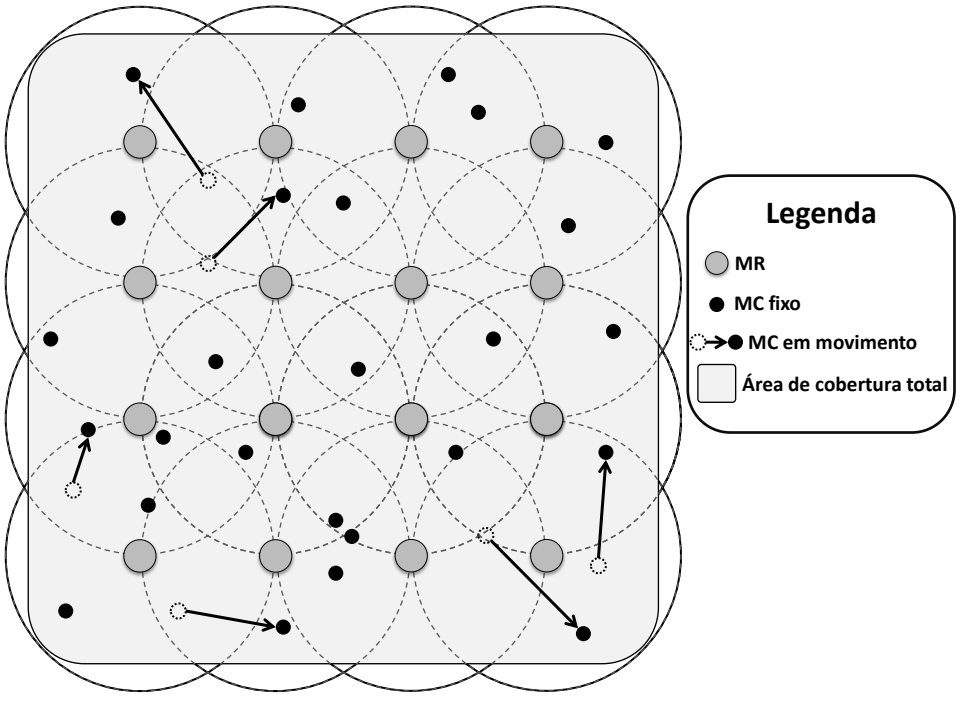


Figura 9 Cenário de rede em malha sem fio com cobertura total

Diferentes topologias e tecnologias podem exigir adaptações dos protocolos utilizados para que se obtenha um melhor desempenho. Por isto, o protocolo MLSD foi desenvolvido de uma maneira que pudesse ser configurável, através de alguns parâmetros, para se adequar a

diversas situações em que possa ser aplicado. Dependendo dos valores especificados, estes parâmetros podem modificar o comportamento do protocolo. Assim, o MLSD tem seus parâmetros configurados com valores padrões para obter um bom desempenho nos cenários de maior utilização.

A especificação do MLSD apresentada neste capítulo está organizada da seguinte forma: na Seção 4.1 apresenta a IWMRA, arquitetura para qual o MLSD foi construído; a Seção 4.2 mostra uma visão geral do protocolo e de seu funcionamento; as estruturas usadas pelo MLSD para manter e divulgar as informações topológicas serão apresentadas nas Seções 4.3, 4.4 e 4.5; na Seção 4.6 é detalhado como é realizada a divulgação dos estados dos enlaces; a Seção 4.7 descreve as operações que podem ser executadas em um enlace; a Seção 4.8 trata do controle de envio de mensagens; a Seção 4.9 descreve a sincronização das bases; o processo pelo qual inconsistências são detectadas e corrigidas é visto na Seção 4.10; o controle dos números de sequência é explicado na Seção 4.11.

4.1. IWMRA – Infrastructure Wireless Mesh Routing Architecture

A IWMRA [Porto 2009] é uma arquitetura de roteamento em três camadas desenvolvida levando em consideração as características específicas das redes em malha sem fio infraestruturadas. Um cenário de aplicação, já apresentado na Figura 9, inclui um conjunto de *Mesh Routers* fixos com posição planejada para proporcionar uma área de cobertura contínua, além de um conjunto de *Mesh Clients*, fixos ou móveis. Na versão inicial da arquitetura, todos os nós têm apenas uma interface sem fio e as ligações são bidirecionais.

Como já mencionado, na arquitetura da RMSF infraestruturada os *Mesh Routers* e *Mesh Clients* desempenham papéis diferentes, em que apenas os *Mesh Routers* são responsáveis por construir um *backbone* sem fio e encaminhar o tráfego dos nós, enquanto *Mesh Clients* usam os recursos da rede. Como os *Mesh Routers* são dispositivos fixos, eles podem ser conectados diretamente à fonte de energia, ao contrário dos *Mesh Clients* que são dispositivos móveis e tem fonte de energia limitada e fornecida através de baterias. Estas características das RMSF infraestruturadas são exploradas pela IWMRA para reduzir a carga de mensagens e o total de mensagens e com isto melhorar a escalabilidade da rede.

De acordo com [Tanenbaum 2003], cinco tarefas são executadas por protocolos de roteamento baseados em estado de enlace:

- Descoberta dos vizinhos;

- Detecção da métrica local para alcançar cada vizinho;
- Geração de mensagem contendo novas informações de vizinhança e métrica (estados dos enlaces);
- Envio destas informações para os outros roteadores da rede; e
- Cálculo do caminho mais curto até cada um dos outros roteadores.

A IWMRA divide as funções do roteamento em três camadas: vizinhança, topologia e roteamento. Desta forma, permite uma melhor divisão de responsabilidades em cada camada. Além disto, também define que um protocolo especializado deve ser construído para executar as funções de cada camada (Figura 10). Cada protocolo é independente e fornece um conjunto de serviços bem definidos para as camadas superiores. Assim, separando as funcionalidades em camadas, a IWMRA permite futuras adaptações.

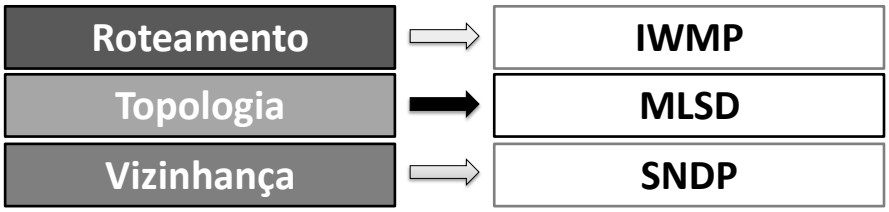


Figura 10 Camadas da IWMRA e os protocolos correspondentes

A camada de vizinhança é responsável por detectar a presença e estado dos vizinhos diretamente conectados, acompanhar as mudanças na vizinhança e alertar a camada da topologia sempre que for detectada uma alteração. A camada de vizinhança também pode detectar a métrica do enlace que é utilizada pelas camadas superiores para calcular o custo total do caminho e selecionar as melhores rotas.

A camada de topologia utiliza inundações para divulgar de forma eficiente as informações obtidas da camada de vizinhança (estados dos enlaces) para todo o *backbone*, permitindo aos *Mesh Routers* construir um mapa topológico da rede. A camada de topologia é responsável por manter as informações topológicas precisas e sincronizar dados entre todos os *Mesh Routers*. Também é função da camada de topologia alertar a camada de roteamento sempre que uma mudança no mapa topológico for detectada, assim as rotas podem ser recalculadas e atualizadas.

Finalmente, adotando uma abordagem proativa e reativa, a camada de roteamento constrói as melhores rotas para todos os nós, sendo responsável pelo cálculo e configuração dos caminhos entre os nós e pelo algoritmo de roteamento.

É importante notar que, na IWMRA, as informações de topologia são mantidas apenas pelos *Mesh Routers*, que são os nós que compõem o *backbone mesh*. Desta forma, os *Mesh Clients* não precisam manter as informações da topologia da rede, economizando recursos (memória, energia e processamento), e, quando precisam se comunicar, usam os serviços da camada de roteamento para requisitar uma rota aos *Mesh Routers* do *backbone*. Como os *Mesh Routers* possuem as informações da topologia, eles podem responder as requisições dos *Mesh Clients* prontamente.

Na IWMRA, a camada de vizinhança é implementada pelo SNDP (*Scalable Neighborhood Discovery Protocol*) [Santos 2008], que adota uma estratégia de sinalização híbrida e colaborativa, nos qual os *Mesh Routers* empregam uma estratégia proativa e orientada a tempo, enquanto os *Mesh Clients* fazem uso de uma estratégia de sinalização reativa e orientada a eventos.

A camada de topologia é implementada pelo MLSD (*Mesh Network Link State Dissemination Protocol*), que é um protocolo proativo e orientado a eventos, é responsável pela geração das mensagens contendo as informações dos estados dos enlaces, bem como a divulgação destas informações, de forma confiável, através do *backbone* sem fio. Além disto, o MLSD implementa uma estratégia de controle das inundações afim de reduzir a sobrecarga de mensagens.

Na camada de roteamento, o IWMP (*Infrastructure Wireless Mesh Protocol*), é um protocolo de roteamento híbrido, de múltiplas rotas. O IWMP faz uso das informações fornecidas pela camada de topologia para construir um grafo e calcular os melhores caminhos usando o algoritmo SPF [Dijkstra 1959]. Assim, o IWMP especifica como efetuar a configuração das rotas, isto é, nos *Mesh Clients* requisitando as rotas e nos *Mesh Routers* respondendo as requisições, calculando os melhores caminhos e realizando o encaminhamento dos datagramas.

Conforme já citado, a IWMRA permite que estas camadas possam ser vistas como interfaces para componentes, já que estes protocolos são independentes entre si. Portanto, podem ser trocados por outros mais adaptados a algum outro cenário, permitindo que o protocolo possa evoluir sem quebrar a compatibilidade.

O MLSD recebe as informações da camada de vizinhança, por este motivo uma descrição sucinta do SNDP será realizada com intuito de favorecer o entendimento.

4.1.1 Protocolo SNDP (Camada de vizinhança)

Os nós que executam protocolos baseados em estado de enlace precisam conhecer os enlaces que formam a vizinhança dos nós do *backbone* e de seus dispositivos vizinhos [Tanenbaum 2003]. Deste modo, na IWMRA, a camada de vizinhança tem como objetivo gerenciar as informações da vizinhança de um nó, para que possam ser usadas pela camada de topologia.

Em cenários onde os nós são móveis, eles podem utilizar mensagens periódicas como forma de anúncio de presença para seus vizinhos. Estas mensagens são conhecidas como "*HELLO"* e permitem que vizinhos do nó, quando receberem ou deixam de receber as mensagens *HELLO*, detectem a presença ou ausência de vizinhos que as emitiram. Entretanto, o envio de mensagens periódicas por todos os nós resulta em uma grande carga de mensagens na rede gerando problemas de escalabilidade [Santos 2008].

Visando reduzir a carga de mensagens *HELLO* na rede, a camada de vizinhança requer uma estratégia baseada na arquitetura planejada das RMSF. O SNDP é um protocolo de vizinhança que efetivamente utiliza as características da RMSF, empregando uma estratégia de colaboração entre *Mesh Client* e *Mesh Router* para informação de detecção ou perda de vizinhança, reduzindo a carga de mensagens e aumentando a escalabilidade da rede.

No SNDP, apenas os *Mesh Routers*, que são os nós que compõem o *backbone* sem fio, enviam mensagens *HELLO* periodicamente. Esta abordagem difere de outros protocolos como OLSR [Clausen 2003], em que todos os nós da rede enviam mensagens *HELLO*. As mensagens *HELLO* do SNDP contêm o endereço do nó emissor, um número de sequência da mensagem e outros campos utilizados para detecção e perda de vizinhança de um nó. Estas mensagens são enviadas em *broadcast* e não são repassadas por nenhum nó. Assim, quando um *Mesh Router* detecta um *HELLO* de outro ele pode adicionar o *Mesh Router* emissor da mensagem como seu vizinho.

Em complemento, um *Mesh Client* que entrar na área de cobertura de um *Mesh Router* e receber uma mensagem *HELLO* deve adicionar o *Mesh Router* em sua tabela de vizinhança e então também emitir uma mensagem *HELLO* para que o *Mesh Router* tenha conhecimento de sua presença. Vale salientar que a mensagem enviada pelo *Mesh Client* ao receber uma mensagem *HELLO* do *Mesh Router* não é periódica, mas gerada por causa da descoberta do *Mesh Router*.

Quando um *Mesh Router* recebe uma mensagem *HELLO* de *Mesh Client*, ele deve adicioná-lo em sua tabela de vizinhança. Além disso, o *Mesh Router* deve indicar, em sua próxima mensagem *HELLO*, que aquele cliente foi descoberto. Assim, ao perceber que foi adicionado pelo *Mesh Router*, o *Mesh Client* não precisa mais enviar nenhuma mensagem *HELLO*. Desta forma, tanto o *Mesh Router* quanto o *Mesh Client* detectam suas vizinhanças. É importante destacar que, como na RMSF infraestruturada não é permitida a comunicação direta entre clientes, o SNDP não detecta os enlaces dos *Mesh Clients* com outros *Mesh Clients*.

Para detectar a perda da vizinhança entre os nós o SNDP também utiliza estratégias complementares quando a vizinhança é estabelecida entre *Mesh Routers* ou entre um *Mesh Router* com um *Mesh Client*.

Por um lado, como os *Mesh Routers* publicam mensagens periodicamente, eles podem perceber a perda da vizinhança com outro *Mesh Router* quando deixar de receber mensagens *HELLO* deste vizinho. Assim, quando um *Mesh Router* vizinho deixar enviar mensagens *HELLO* o *Mesh Router* assume que o enlace foi rompido e remove a vizinhança com o nó da tabela de vizinhança.

Por outro lado, os *Mesh Clients* não publicam mensagens periodicamente. Assim, para um *Mesh Router* perceber a perda de vizinhança com um *Mesh Client* é necessária a colaboração deste e de outros *Mesh Routers* vizinhos. Sempre que recebe uma mensagem *HELLO* de um *Mesh Router*, o *Mesh Client* define um tempo de expiração da informação na vizinhança. Desta forma, quando uma mensagem *HELLO* do *Mesh Router* é recebida, este tempo para expiração é atualizado.

Entretanto, quando um *Mesh Client* se movimenta ele pode sair do raio de alcance do *Mesh Router* e deixar de receber as mensagens. Assim, caso *Mesh Client* deixe de receber alguma mensagem de um *Mesh Router*, o tempo de expiração para a entrada na tabela de vizinhança é alcançado, então o *Mesh Client* assume que houve perda da vizinhança com este *Mesh Router*. Contudo, ainda é necessário que o *Mesh Router* também perceba que também perdeu a vizinhança. Por isto, o *Mesh Client* envia uma mensagem *HELLO* notificando a perda de vizinhança com o *Mesh Router*. Naturalmente esta notificação não pode ser recebida diretamente pelo *Mesh Router* destino, mas ainda pode ser recebida por outros *Mesh Routers* vizinhos, que são responsáveis por encaminhar a notificação de perda para o roteador indicado pelo *Mesh Client*.

Com isto, o *Mesh Router* ao receber a notificação de perda do *Mesh Client*, encaminhada pelos outros *Mesh Routers*, também remove as informações da vizinhança como *Mesh Client*. Para não gerar tráfego extra, esta notificação segue encapsulada dentro das mensagens *HELLO* dos *Mesh Routers*.

A comunicação com a camada de topologia se dá sempre que for detectada alguma modificação da vizinhança. Isto é, sempre que for detectada a presença ou percebida a perda de um nó, a camada de vizinhança alerta a camada de topologia informando qual modificação foi realizada e em qual nó.

Como já mencionado, a especificação da camada de vizinhança é definida pelo protocolo de descoberta de vizinhança SNDP. Mais detalhes do funcionamento do protocolo podem ser encontrados em [Santos 2008].

4.2. Protocolo MLSD – *Mesh Network Link State Dissemination Protocol*

O Protocolo MLSD especifica a camada de topologia do IWMRA. Ele é encarregado de divulgar os estados dos enlaces para todo o *backbone* através de mensagens de topologia emitidas quando ocorre um evento. Os eventos são quaisquer operações nos enlaces informados pela camada de vizinhança. Por exemplo, quando um *Mesh Router* adiciona um *Mesh Client* como um novo vizinho.

Conforme a arquitetura da IWMN, o MLSD assume que somente os *Mesh Routers* compõem o *mesh backbone*. Portanto, apenas os *Mesh Routers* enviam e processam as mensagens de atualização de topologia, chamadas LSU (*Link State Update*). Os *Mesh Clients* não enviam nem processam LSUs Isto é, para os *Mesh Clients*, as informações de topologia são somente os dados recuperados diretamente pela camada de vizinhança, sendo composta apenas dos *Mesh Routers* vizinhos.

O MLSD gerencia a emissão de LSUs para reduzir a carga e o total de mensagens de topologia e com isto melhorar a escalabilidade, permitindo a rede possuir um grande número de nós. Além disto, o MLSD elimina mensagens desatualizadas para evitar inconsistências na base topológica. A base topológica é a estrutura que permite que cada *Mesh Router* construa uma visão completa da topologia da rede e é igual para todos os roteadores.

Conforme mencionado no Capítulo 3, os protocolos de roteamento proativos para RMSF utilizam uma estratégia de *flooding* em que as mensagens de atualização são emitidas sem garantia de entrega e carregam todos os enlaces detectados. Por isto, a consistência das

informações da topologia se dá através do envio periódico de mensagens, obedecendo a uma frequência de envio. Desta maneira, caso algum nó perca uma mensagem, ele terá a chance de corrigir a informação numa próxima mensagem.

Entretanto, a forma de divulgação periódica possui importantes desvantagens: pode aumentar o total de mensagens na rede – caso a frequência de publicação seja alta – ou pode também gerar uma latência no tempo de convergência da rede – caso a frequência de publicação seja baixa e as mensagens demorem a serem enviadas [Lee 1999].

A principal característica do MLSD é a geração de mensagens orientada a eventos e a divulgação das informações empregando uma estratégia de entrega confiável e incremental. Assim, a consistência das informações da topologia é mantida através de dois processos: divulgação das atualizações dos estados dos enlaces utilizando um *flooding* de forma confiável e incremental, e a sincronização das bases topológicas.

Quando a movimentação dos *Mesh Clients* aumenta, uma grande quantidade de eventos pode ser gerada, o que também pode resultar num aumento do total de mensagens. O MLSD controla o envio de mensagens acumulando vários eventos para serem enviados de uma vez. Além disto, o MLSD controla um intervalo entre as retransmissões das mensagens em função da quantidade de eventos, reduzido o total de mensagens. Ainda, através de um formato compacto para representar as operações, o MLSD também reduz a carga das mensagens enviadas no anúncio de diversos eventos. As estruturas do protocolo MLSD utilizadas nesses processos serão apresentadas a seguir.

4.3. Formato da Mensagem LSU

A mensagem LSU (*Link State Update*) é o pacote usado pelo MLSD para divulgar as atualizações no *backbone*. Cada LSU pode carregar um ou mais anúncios de eventos ocorridos na topologia, que no MLSD é chamado de LSA (*Link State Advertisement*). Cada LSA representa um conjunto de estruturas que condensam as atualizações ocorridas nos estados dos enlaces de cada *Mesh Router*.

A mensagem LSU é enviada diretamente no quadro da camada MAC e emitida em *broadcast*. Todos os *Mesh Routers* que recebem a LSU devem avaliar cada atualização de topologia presente no pacote individualmente. Assim, cada *Mesh Router* decide se deve ou não repassar a atualização da topologia em sua própria LSU. A mensagem LSU é exibida na

Figura 11, onde os números representam os tamanhos dos campos em bits. Os campos com tamanho variável não possuem tais números associados.

type (8)	version (8)	src_addr (32)	
num_forwarders(8)	forwarders (32)	forwarders (32)	...
total_updates (16)	LSAs	LSAs	...
bitmap			

Figura 11 Mensagem LSU

Os campos da mensagem LSU (*Link State Update*) são:

- ***type:*** identifica que a mensagem é um LSU do protocolo MLSD;
- ***version:*** identifica a versão do protocolo (atualmente 1);
- ***src_addr:*** identifica o endereço de origem da LSU (endereço IP);
- ***num_forwarders:*** informa a quantidade de *Mesh Routers* presentes na lista de encaminhadores;
- ***forwarders:*** lista de endereços dos *Mesh Routers* vizinhos que são encaminhadores de alguma atualização de topologia presente no pacote LSU;
- ***total_updates:*** quantidade de anúncios de atualizações dos estados dos enlaces presentes no pacote;
- ***LSAs:*** lista de anúncios das atualizações dos estados dos enlaces de um *Mesh Router* (cada elemento da lista é um LSA); e
- ***bitmap:*** mapa de bits que realiza a correspondência entre as diversas atualizações dos estados dos enlaces informadas nas LSAs e os elementos da lista de encaminhadores (*forwarders*) do pacote.

Em linhas gerais, os campos *num_forwarders*, *forwarders*, e *bitmap* definem como as atualizações dos estados dos enlaces informadas nas LSAs devem ser processadas por cada *Mesh Router* da lista de *forwarders*.

O tamanho do *bitmap* é variável e seu valor, em bits, é obtido pela multiplicação do *num_forwarders* com o *total_updates*. Estes campos serão descritos no processo de divulgação dos estados dos enlaces com garantia de entrega na seção 4.6.

A LSA contém todas as atualizações dos estados dos enlaces de um dado *Mesh Router*. Na LSA, os nós que compõem os enlaces com o *Mesh Router* são informados agrupados de acordo com a operação que deve ser executada sobre o enlace. A LSA é mostrada na Figura 12.

MR_addr (32)		num_operations (16)	
LSA_operation	LSA_operation	LSA_operation	...

Figura 12 LSA - *Link State Advertisement*

A LSA é composta dos seguintes campos:

- ***MR_addr:*** informa o endereço (IP) do *Mesh Router* que gerou a LSA;
- ***num_operations:*** informa a quantidade de atualizações dos enlaces do *Mesh Router* com endereço indicado no campo *MR_addr*; e
- ***LSA_operation:*** estrutura que define a operação que será realizada agrupando por tipo de nó.

O campo *LSA_operation* define um cabeçalho para agrupar uma lista de nós do mesmo tipo (*Mesh Client/Mesh Router*) e as operações do mesmo tipo. As operações que podem ser informadas nas LSA são: ADD, para adicionar um enlace; e REM, para anunciar a perda de um enlace. Através desta representação, por exemplo, todas as adições de *Mesh Clients* na vizinhança de um *Mesh Router* podem ser informadas de uma vez, agrupando uma lista de endereços com o cabeçalho informando a operação ADD e tipo *Mesh Client*. A *LSA_operation* é apresentada na Figura 13.

neighbor_type(4)		operation_type(4)	
num_updates (16)	neighbor_data	neighbor_data	...

Figura 13 *LSA_Operation*

A *LSA_operation* possui os seguintes campos:

- ***neighbor_type:*** tipo do vizinho (*Mesh Router*/*Mesh Client*);
- ***operation_type:*** informa a operação no enlace (ADD/REM);
- ***num_updates:*** informa a quantidade de operações para um mesmo tipo de vizinho e tipo de operação; e

- ***neighbor_data:*** informa o endereço do vizinho que compõe o enlace que foi atualizado com a métrica e um número de sequência.

O campo *neighbor_type* determina se a execução da operação exige um processamento adicional em função do tipo do nó que compõe o enlace, por exemplo, simplesmente adicionando ou removendo um enlace caso o vizinho seja um *Mesh Client* ou mesmo iniciando um processo de sincronia ou de limpeza dos nós inalcançáveis da base topológica caso o vizinho seja um *Mesh Router*. Além disto, no campo *neighbor_data* são informados os detalhes do vizinho que compõe o enlace, como apresentado na Figura 14.

neighbor_addr(32)	metric(8)	seq_number

Figura 14 *Neighbor_data*

A *Neighbor_data* possui os seguintes campos:

- ***neighbor_addr:*** endereço (IP) do vizinho que compõe o enlace atualizado;
- ***metric:*** métrica do enlace
- ***seq_number:*** número de sequência da atualização.

O valor do campo *metric* é informado pela camada de vizinhança e utilizado pela camada de roteamento para calcular o custo total do caminho através das somas dos custos dos enlaces intermediários. O MLSD não realiza qualquer processamento sobre este dado, pois sua função é de apenas disseminar a informação através do *backbone*. Desta maneira, qualquer métrica representada neste campo pode ser utilizada sem restrições, por exemplo, a métrica ETX [Couto 2005] ou ETT [Draves 2004].

Vale ressaltar que o protocolo SNDP implementa a camada de vizinhança e informa ao MLSD a métrica *hop-count*, que descreve apenas o estado da conectividade entre os vizinhos (se é vizinho ou não). Portanto, o valor da métrica é igual a 1, caso seja informada a operação de adição do enlace. Na operação de remoção o valor deste campo não é avaliado.

O campo *seq_number* informa o número da atualização do enlace. Este número é mantido pelo *Mesh Router* (*MR_addr* informado na LSA) e é incrementado em qualquer atualização dos enlaces. O tamanho deste campo também é variável, podendo ocupar 16 bits ou 8 bits. A forma como o tamanho do número de sequência é computado e o processo realizado quando o número alcançar o valor limite e for reiniciado é descrito na Seção 4.11.

É importante observar que o MLSD determina a quantidade de atualizações podem ser carregadas em cada pacote LSU através de um parâmetro configurável. Este parâmetro pode ser modificado respeitando o tamanho máximo do quadro da camada MAC. Por exemplo, para o 802.11, a configuração padrão do MLSD define que é permitido enviar até 128 atualizações de estados dos enlaces, que podem estar presentes em uma ou mais LSAs. Caso o número de atualizações seja superior a 128, outra LSU deve ser gerada.

Também vale salientar que, a carga de dados do pacote depende do conjunto de atualizações a serem enviadas na LSU como efeito do formato compacto adotado pelo MLSD. Por exemplo, quando 128 atualizações de topologia são enviadas para quatro *Mesh Routers* vizinhos definidos como encaminhadores, no pior caso, cada LSA vai carregar apenas uma atualização e o tamanho do pacote, resultante da soma de todos os campos, é de 2137 bytes. Contudo, se as mesmas 128 atualizações estiverem discriminadas em apenas uma única LSA, por exemplo, um único *Mesh Router* realizou a adição de um conjunto de *Mesh Clients*, então o tamanho do pacote resultante da soma de todos os campos é de 867 bytes.

4.4. *Buffer* de Envio

O *buffer* de envio é uma estrutura interna que armazena informações sobre as atualizações de topologia geradas pela vizinhança do nó ou recebida através de outras mensagens LSU e que deverão ser divulgadas para todo o *backbone*. De acordo com as informações existentes no *buffer* de envio, as LSUs podem ser construídas e despachadas para transmissão. Os campos do *buffer* de envio podem ser vistos na Figura 15.

MR_addr	neighbor_addr	neighbor_type	seq_number	metric	operation	send_status	time	forwarders_list

Figura 15 *Buffer* de envio

Os campos do *buffer* de envio são:

- ***MR_addr:*** endereço (IP) do *Mesh Router* que gerou a atualização;
- ***neighbor_addr:*** endereço (IP) do nó vizinho que compõe o enlace;
- ***neigbor_type:*** tipo do nó vizinho (*Mesh Router*/*Mesh Client*);
- ***seq_number:*** número da atualização do enlace do *MR_addr*;
- ***metric:*** métrica do enlace;

- ***operation:*** operação de atualização no enlace a ser executada;
- ***send_status:*** define se a atualização já foi transmitida alguma vez, assume *true* se a mensagem já foi enviada ou *false* se nunca foi enviada;
- ***time:*** registra um tempo mínimo de espera para transmissão da atualização de topologia no *buffer* de envio; e
- ***forwarders_list:*** lista de endereços dos *Mesh Routers* vizinhos que compõe a lista de e encaminhadores (*forwarders*) para esta atualização de topologia;

O campo *send_status* determina se a atualização de topologia já foi emitida pelo menos uma vez pelo *Mesh Router* e o campo *time* determina quando a mensagem deve ser transmitida, ou retransmitida. O uso destes campos é apresentado no processo de divulgação dos estados dos enlaces com garantia de entrega que será descrito nas Seções seguintes.

O campo *forwarder_list* mantém a lista de endereços de *Mesh Routers* vizinhos que devem realizar o encaminhamento desta atualização. O conjunto dos *Mesh Routers* indicados como *forwarder* nas atualizações de topologia que serão enviadas irão formar a lista de *forwarders* da LSU.

4.5. Base Topológica

A base topológica, vista na Figura 16, representa o mapa de topologia da rede e é a partir dela que a camada de roteamento obtém as informações para calcular os caminhos entre os nós. A base topológica mantém as informações da topologia obtidas através das mensagens LSU recebidas da rede e das informações da camada de vizinhança. É importante ressaltar que as informações registradas na base topológica não expiram, só são removidas e adicionadas mediante eventos ocorridos e divulgados através de mensagens na rede.

MR_addr	neighbor_addr	neighbor_type	seq_number	metric

Figura 16 Base Topológica

Os campos da base topológica são:

- ***MR_addr:*** endereço (IP) do *Mesh Router* ;
- ***neighbor_addr:*** endereço (IP) do nó vizinho;
- ***neigbor_type:*** tipo do nó vizinho (*Mesh Router*/*Mesh Client*);

- ***seq_number:*** número da atualização do enlace do *MR_addr*; e
- ***metric:*** métrica do enlace.

A visão da base topológica é a mesma para todos os *Mesh Routers* que compõem o *backbone*. Assim, todos os *Mesh Routers* conhecem os enlaces entre todos os nós da rede.

4.6. Divulgação dos Estados dos Enlaces

O MLSD mantém a consistência das informações topológicas através de dois mecanismos: a divulgação dos estados dos enlaces e o processo de sincronização das bases topológicas.

A divulgação das atualizações dos estados dos enlaces é realizada através das mensagens LSU, que contém as atualizações incrementais da topologia nas LSAs. Estas atualizações, quando processadas por um *Mesh Router*, são reencaminhadas para os *Mesh Routers* vizinhos, até alcançar todos os nós do *backbone* da rede. Logo, a divulgação das atualizações dos estados dos enlaces adota o conceito de inundação (*flooding*).

Como as mensagens LSU são enviadas em *broadcast*, no 802.11, este envio é realizado apenas uma vez, sem recursos da camada MAC como RTS/CTS ou reconhecimento positivo (*Ack*) e, portanto, estão mais sujeitas a perdas.

Para realizar a entrega confiável, o MLSD implementa um mecanismo de reconhecimento positivo implícito com retransmissão. Adicionalmente, o mecanismo de processamento determina se cada uma das atualizações de topologia recebida na mensagem deve ser encaminhada adiante, reconhecida, reenviada ou ignorada. Desta forma, este processo permite a disseminação das atualizações através do *backbone* enquanto assegura que atualizações antigas não prejudiquem a consistência das informações.

Além disto, o MLSD controla o tempo em que as mensagens são retransmitidas para que o excesso de eventos não sobrecarregue a rede. Estes processos serão detalhados a seguir.

4.6.1 Reconhecimento Positivo Implícito com Retransmissão

No mecanismo de reconhecimento positivo implícito, um *Mesh Router* que envia uma atualização de topologia detecta que a atualização foi efetivamente entregue a um *Mesh Router* vizinho indicado como encaminhador, quando o *Mesh Router* vizinho repassar a mesma atualização, em sua própria LSU, para um próximo *Mesh Router*. Como o envio da LSU é realizado em *broadcast*, o *Mesh Router* que primeiramente enviou a atualização de

topologia também recebe a mensagem, e, portanto, esta serve de reconhecimento da recepção do *Mesh Router* vizinho.

A atualização de topologia pode ser gerada pelo próprio *Mesh Router*, quando alguma modificação na vizinhança for detectada, ou pode ser recebida de algum *Mesh Router* vizinho.

Por um lado, quando a atualização de enlace é informada pela camada de vizinhança do próprio *Mesh Router*, todos os *Mesh Routers* vizinhos devem receber e repassar a mensagem. Portanto, todos eles são marcados como encaminhadores (*forwarders*) para a atualização.

Por outro lado, quando um *Mesh Router* recebe uma nova atualização de topologia através da LSU e estiver indicado como encaminhador da atualização, todos os vizinhos do *Mesh Router*, excetuando o *Mesh Router* que lhe enviou a mensagem, devem ser informados que devem repassar a atualização de topologia em sua própria LSU. Para isto, todos os *Mesh Routers* vizinhos são marcados como encaminhadores para a atualização, excetuando o *Mesh Router* de quem a LSU foi recebida.

A lista de encaminhadores da atualização de topologia de um *Mesh Router* é um subconjunto dos *Mesh Routers* vizinhos. Vale ressaltar que várias atualizações de topologia podem ser enviadas em uma mesma LSU e cada atualização contida nas LSAs da LSU possui seu próprio conjunto de encaminhadores.

Se um *Mesh Router* vizinho não repassar a atualização em certo intervalo de tempo, o *Mesh Router* a envia novamente até que perceba o vizinho repassá-la. O *Mesh Router*, ao perceber o repasse da mensagem, internamente reconhece que o vizinho a recebeu com sucesso e não precisará enviar a mensagem novamente.

É importante destacar que o reconhecimento da atualização de topologia é realizado quando o *Mesh Router* percebe o repasse. Por isto, todo *Mesh Router* deve enviar a atualização de topologia pelo menos uma vez, mesmo que não possua outros vizinhos para repassá-la. Isto ocorre porque o envio é necessário ao processo de reconhecimento. Neste caso, o envio é realizado sem indicar nenhum encaminhador na LSU, pois a informação serve apenas para reconhecimento do vizinho que enviou inicialmente.

A Figura 17 apresenta o mecanismo de reconhecimento positivo implícito. Nesta figura, podemos ver que o *Mesh Router* A envia uma LSU contendo uma atualização de topologia para o único vizinho *Mesh Router* B, informando a adição da vizinhança do *Mesh Client* X (Figura 17a). O *Mesh Router* B, por sua vez, repassa a mensagem indicando o

vizinho C como encaminhador (Figura 17b). A LSU de B é recebida tanto por A quanto por C. Assim, o *Mesh Router* A entende a mensagem de B como um reconhecimento. O *Mesh Router* C, que é indicado como encaminhador por B, deve repassar a mensagem. Contudo, C não possui outros vizinhos. Mesmo assim, C envia a mensagem sem indicar qualquer encaminhador para que o *Mesh Router* B possa reconhecer que C a recebeu (Figura 17c).

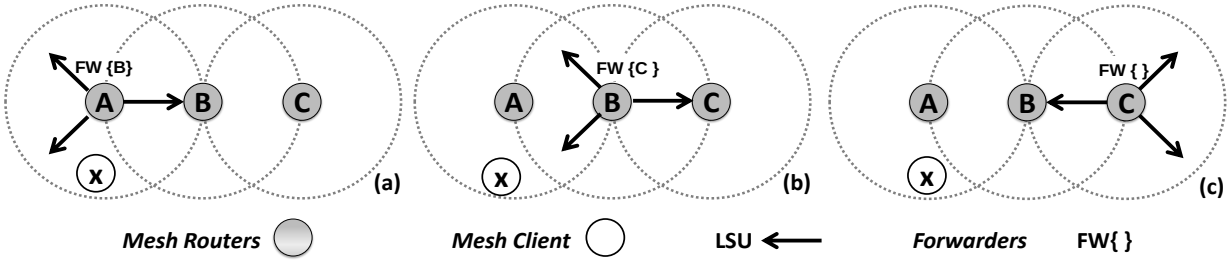


Figura 17 Mecanismo de reconhecimento positivo implícito

É importante enfatizar que, como o reconhecimento é implícito, nenhuma mensagem adicional precisa ser enviada. Assim, quando as atualizações de topologia são recebidas e repassadas com sucesso, cada *Mesh Router* precisa enviá-la somente uma vez. Além disto, se alguma mensagem for perdida, apenas o vizinho que não percebeu o repasse precisa reenviar. Esta estratégia é diferente do processo de *flooding* convencional, em que as mensagens também são enviadas por cada nó apenas uma vez, mas como não há garantia de entrega, e se a mensagem for perdida, um novo *flooding* precisa ser realizado novamente.

Conforme mencionado, quando um *Mesh Router* envia uma mensagem de atualização para um vizinho, ele aguarda um tempo para o vizinho realizar o repasse (este tempo será explicado na Seção 4.8). Se o vizinho não realizar o repasse, o *Mesh Router* envia a mensagem novamente. Se o enlace com o *Mesh Router* vizinho for removido, os repasses pendentes do vizinho deixam de ser esperados. Isto é, não é mais necessário reconhecer os repasses do vizinho e consequentemente nenhuma mensagem indicando o vizinho perdido como encaminhador é transmitida.

Quando existir mais de um *Mesh Router* vizinho marcado como encaminhador para a atualização da topologia, o *Mesh Router* precisa perceber que todos os encaminhadores realizaram o repasse em suas mensagens. Devido a colisões ou outros problemas de transmissão é possível que algum vizinho não receba a LSU enviada e, portanto, não irá repassar mensagem. Neste caso, o *Mesh Router* irá receber os repasses de alguns vizinhos, mas não de todos e por isto deverá enviar a atualização da topologia novamente. Todavia,

apenas o vizinho que não recebeu a mensagem precisa repassa-la, pois os outros já o fizeram. Assim, apenas o *Mesh Router* que não repassou a mensagem é informado como encaminhador da atualização de topologia. Esta situação pode ser visualizada e melhor entendida através da Figura 18.

A Figura 18 apresenta a robustez do mecanismo de reconhecimento positivo implícito, caso alguma mensagem seja perdida. É possível observar o *Mesh Router* B enviando uma LSU, contendo uma atualização de topologia, informando a adição do enlace com o *Mesh Client* Y que é recebida pelo *Mesh Router* A, mas não é recebida pelo *Mesh Router* C (Figura 18a). Então, o *Mesh Router* B percebe que somente o *Mesh Router* A propagou a atualização adiante (Figura 18b). Ao final de certo tempo, o *Mesh Router* B reenvia a atualização de topologia informando, na LSU, que apenas o *Mesh Router* C precisa repassá-la (Figura 18c). Desta forma, apenas o *Mesh Router* C repassa a atualização e o *Mesh Router* B reconhece que C a enviou.

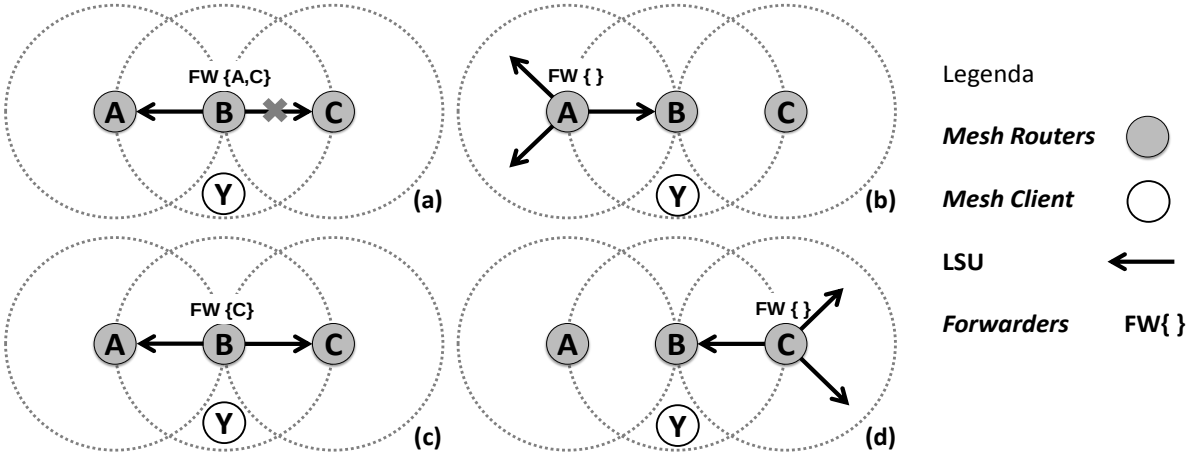


Figura 18 Reconhecimento positivo implícito com retransmissão (perda de mensagem)

Um *Mesh Router* pode receber uma mensagem com uma atualização de topologia indicando-o como encaminhador, mesmo que a mensagem não seja nova. Isto ocorre quando algum vizinho não percebeu o *Mesh Router* realizar o repasse e reenvia a mensagem, ou ainda quando o *Mesh Router* recebe a mesma atualização de mais de um vizinho antes que ele envie sua mensagem. No primeiro caso, a atualização de topologia é reenviada e não precisa ser repassada. Para isto, a atualização não deve ter encaminhador indicado porque ela serve apenas para o reconhecimento pelo vizinho. No segundo caso, o *Mesh Router* reconhece que o vizinho já possui a atualização de topologia, de forma antecipada, assim o vizinho não será indicado como encaminhador da atualização e não precisará realizar o repasse novamente.

A Figura 19 ilustra um reconhecimento antecipado: supondo que os *Mesh Routers* C e D repassem uma atualização de topologia do *Mesh Router* A, indicando como encaminhador o vizinho B, antes que o *Mesh Router* B encaminhasse a mensagem (Figura 19a). Assim, ao receber a primeira mensagem o *Mesh Router* B indicará como encaminhadores os outros vizinhos que ainda não enviaram a mensagem, mas ao receber a segunda mensagem, realiza o reconhecimento antecipado e assim deve enviar a atualização de topologia sem encaminhadores, apenas para permitir que vizinhos C e D (Figura 19b) realizem o reconhecimento da mensagem.

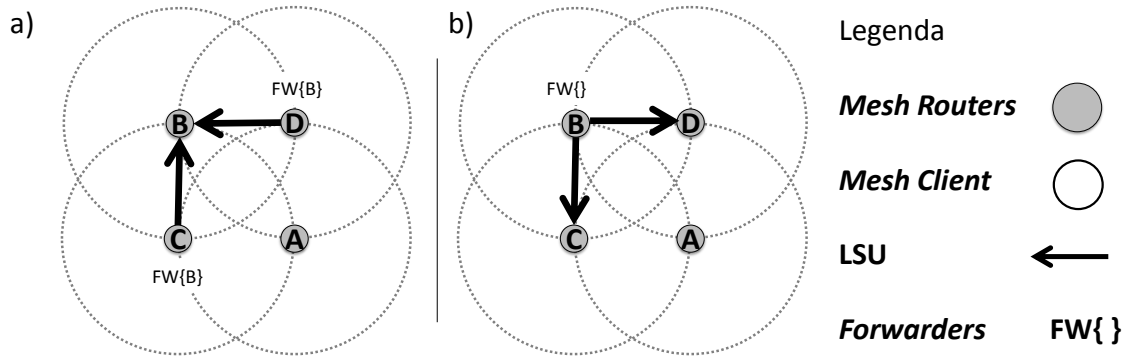


Figura 19 Reconhecimento antecipado

Vale ressaltar que, um *Mesh Router* sabe que é encaminhador para alguma atualização de topologia quando seu endereço está presente na lista de encaminhadores da LSU recebida. Todos os endereços dos encaminhadores das atualizações do pacote são informados nesta lista. Com isto, um *bitmap* relaciona cada atualização de topologia com a lista de encaminhadores. Assim, cada atualização de topologia tem os encaminhadores representados em uma pequena parte do *bitmap*, esta parte informa quais *Mesh Routers* da lista de encaminhadores estão indicados como encaminhadores para a atualização.

O tamanho da parte de cada atualização de topologia no *bitmap* tem um número de bits igual à quantidade de elementos da lista de encaminhadores. Desta forma, é possível realizar a correspondência entre a lista de encaminhadores e cada atualização de topologia, informando quais dos elementos da lista estão marcados como encaminhadores, assumindo o valor 1, ou 0 caso não seja encaminhador.

A sequência de bits do *bitmap* deve ser construída obedecendo à mesma ordem em que as atualizações de topologia correspondentes aparecem na LSU e a ordem em que os endereços dos encaminhadores aparecem na lista de encaminhadores. Por exemplo, no *bitmap* da na Figura 20, para a primeira atualização de topologia (MR-A ADD X #7), o *Mesh Router*

D é encaminhador, mas C não. Para segunda atualização de topologia, (MR-A ADD Y #8), D não é encaminhador, mas C é. Para a terceira atualização de topologia, (MR-A ADD Z #9), ambos são encaminhadores.

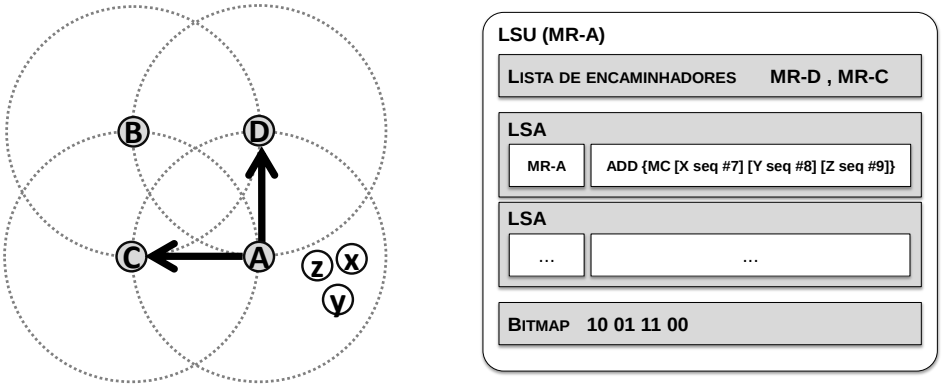


Figura 20 Mapa de bits na LSU

O tamanho total em bits do *bitmap* é calculado multiplicando quantidade de atualizações de topologia a serem enviadas na LSU pela quantidade de elementos da lista de encaminhadores. Por convenção, o tamanho do *bitmap* deve ser múltiplo um de 8 bit. Portanto, o valor resultante da multiplicação, se não for um múltiplo de 8, deve ser arredondado para o primeiro múltiplo de 8 maior que o tamanho do *bitmap*. Os bits acrescentados devem ser definidos com valor 0, conforme também pode ser verificado no exemplo da Figura 20.

A utilização de um *bitmap* evita repetir o endereço de um mesmo encaminhador na LSU quando várias atualizações de topologia forem enviadas para um mesmo *Mesh Router*. Assim, ao invés de informar vários endereços de 32 bits para cada atualização de topologia, um pequeno conjunto de bits com tamanho, no máximo, igual ao número de *Mesh Routers* vizinhos (usualmente, menor ou igual a 4), para cada atualização de topologia, realiza a correspondência com os endereços informados na lista de encaminhadores da LSU.

4.6.2 Processamento das Atualizações de Topologia

Diversos eventos podem ser gerados na vizinhança do *Mesh Router*. Além disto, o *Mesh Router* pode receber diversas atualizações através das LSUs dos vizinhos. Quando uma atualização de topologia é informada, seja através da camada de vizinhança ou do recebimento de uma LSU, o processamento da atualização determina se a atualização deve ser divulgada adiante, reconhecida, reenviada, ignorada ou disparar um processo de correção.

Quando um *Mesh Router* é informado pela camada de vizinhança de alguma modificação no estado dos seus enlaces, ele gera uma atualização de topologia que deve ser registrada com um incremento do seu número de sequência. As atualizações informadas pela camada de vizinhança de um nó são consideradas sempre como novas e, portanto, devem ser inseridas no *buffer* de envio para serem divulgadas adiante, repassando para os vizinhos.

Quando o *Mesh Router* processa uma atualização de topologia recebida através LSU, o processamento realizado na atualização deve verificar ele está marcado como encaminhador na parte do *bitmap* da LSU correspondente à atualização, e se a atualização é nova.

Para toda operação nova, recebida ou detectada, deve ser computada da operação (adicionar ou remover) informada na mensagem, registrando efetivamente as modificações do enlace na base topológica. A computação de cada operação será detalhada na Seção 4.7.

No processamento da atualização de topologia, as atualizações de topologia antigas são removidas do *buffer* de envio e a atualização de topologia mais nova é inserida. Os elementos *<MR_addr, neighbor_addr, seq_number>*, que compõem a atualização de topologia, definem unicamente um anúncio de atualização de um enlace. Assim, a atualização de um enlace denotado por *MR_addr* e *neighbor_addr* com um *seq_number* mais antigo deve ser removida e a informação que representa o estado mais recente deve ser inserida, conforme mostrado na Figura 21.

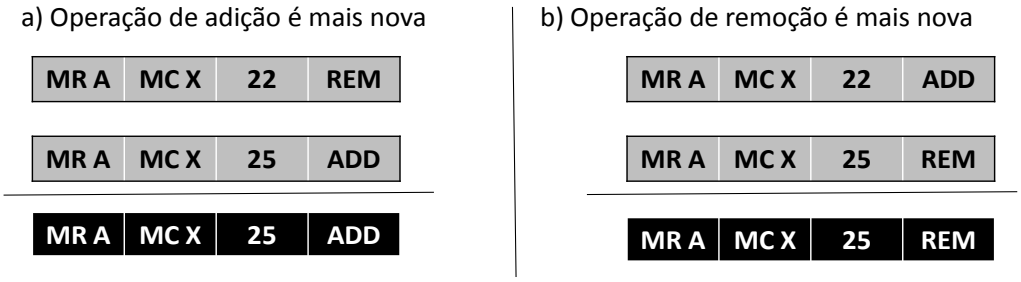


Figura 21 Inserção da atualização de topologia nova no *buffer* de envio

A remoção de uma atualização de topologia antiga devido à inserção de uma atualização mais nova ocorre, por exemplo, quando um *Mesh Client* em movimento retorna rapidamente a vizinhança de um *Mesh Router*, então o anúncio da atualização de topologia informando a adição do enlace (ADD), mais novo, substitui o de remoção (REM), mais antigo, antes dele ser enviado. Ou ainda, durante o *flooding*, quando atrasos de transmissão da rede e colisões podem fazer a LSA mais nova alcançar uma mais antiga.

Enquanto um *Mesh Router* estiver aguardando o reconhecimento de alguma atualização enviada para algum vizinho, uma atualização mais nova sobre o mesmo enlace pode ser recebida. Então, as atualizações antigas do enlace são removidas do *buffer*, mesmo as pendentes de reconhecimento. Neste caso, o *Mesh Router* deixa de esperar os reconhecimentos das mensagens antigas dos vizinhos, pois a atualização mais nova deverá ser repassada para os vizinhos. Quando os vizinhos receberem a atualização nova, também eliminarão a antiga no *buffer* de envio.

É importante destacar que quando ocorre a eliminação de uma atualização de topologia mais antiga no *buffer* de envio, a ordem de ocorrência dos eventos não é mantida. Esta ordem não é necessária ao MLSD, pois cada atualização definida por <*MR_Addr*, *neighbor_addr*, *seq_number*> é avaliada somente com relação ao mesmo enlace (*MR_Addr*, *neighbor_addr*). Assim, não importa a ordem de avaliação de atualizações de enlaces envolvendo pares de vizinhos diferentes. O número de sequência define qual atualização do enlace é a mais nova.

Pra determinar se uma atualização é nova é necessário observar o tipo da operação, o estado do *buffer* de envio e da base topológica. Por exemplo, para classificar uma operação de remoção recebida é preciso verificar se existe alguma informação do enlace na base de dados. Caso não exista, ainda é necessário observar o *buffer* de envio, pois a atualização já pode ter sido processada e estar aguardando reconhecimentos dos vizinhos. Na Seção 4.7 será explicado como através de cada operação, examinando o *buffer* de envio e a base topológica, as atualizações recebidas são classificadas como novas ou não. Para simplificar a explicação é suficiente saber, neste momento, se a atualização é nova ou não.

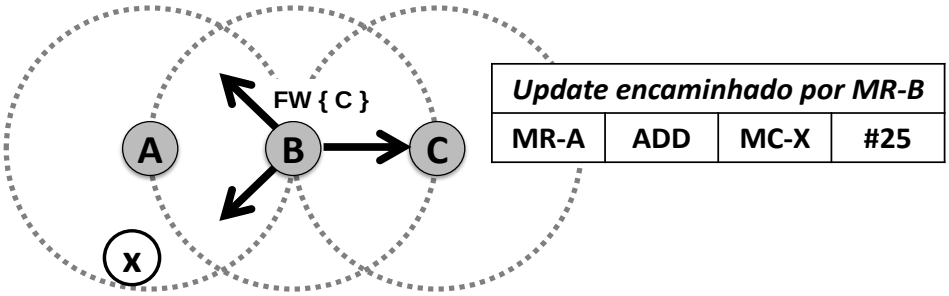
Assim, quando um *Mesh Router* recebe uma atualização de topologia nova e estiver marcado como seu encaminhador, as atualizações antigas do mesmo enlace são removidas do *buffer* de envio, conforme já explicado. Além disto, a modificação do enlace é efetivada (ADD/REM) na base topológica e a nova atualização deve ser inserida no *buffer* de envio para ser divulgada adiante, seguindo o mecanismo de reconhecimento positivo implícito.

Um *Mesh Router* também pode receber uma atualização que não é nova, isto é, a atualização já foi processada. Neste caso, é necessário verificar se o *Mesh Router* é encaminhador para a atualização de topologia recebida na mensagem, para determinar se a atualização trata-se do reconhecimento de algum vizinho, pedido de retransmissão ou deve ser ignorada.

Assim, quando o *Mesh Router* não é encaminhador e a atualização não é nova, então é preciso verificar a atualização trata-se de algum reconhecimento. Para isto, o *buffer* de envio deve ser examinado. Desta forma, se o *Mesh Router* que enviou a mensagem estiver na lista de encaminhadores para a referida atualização, ele deve ser removido da lista e, dessa maneira, o reconhecimento do vizinho é internamente registrado pelo *Mesh Router*, conforme podemos observar no exemplo da Figura 22.

Em complemento, nas situações em o *Mesh Router* não está indicado como encaminhador para a atualização de topologia e não se trata de um reconhecimento, a atualização é ignorada.

A Figura 22 apresenta como o reconhecimento da atualização de topologia é registrado. Supondo o *Mesh Router* A, tendo enviado a atualização de topologia indicando B como encaminhador (note o valor *true* do campo *sent_status* no *buffer* de envio de A), vemos que o endereço de B está na lista de encaminhadores do *buffer* de envio de A. Quando o *Mesh Router* B repassa a atualização para o C, o *Mesh Router* A, que também recebe a mesma mensagem, entende como um reconhecimento de B e pode eliminar B da lista de nós que precisam reconhecer a mensagem.



Update encaminhado por MR-B			
MR-A	ADD	MC-X	#25

Buffer de envio do *Mesh Router* A após o recebimento da mensagem de B								
A	X	MC	25	1	ADD	true	0	

Figura 22 *Mesh Router* A recebe o reconhecimento do *Mesh Router* B

Quando um *Mesh Router* recebe uma mensagem em que é indicado como encaminhador para uma atualização que não é nova, pode ter ocorrido uma de duas situações: algum vizinho não recebeu o reconhecimento do *Mesh Router*, ou tratar-se de um reconhecimento antecipado.

No primeiro caso, a atualização de topologia é simplesmente reenviada. No segundo caso, que já foi mostrado no exemplo da Figura 19, o *Mesh Router* deve ser removido da lista

de encaminhadores e a mensagem é enviada indicando apenas os encaminhadores restantes, ou nenhum encaminhador caso a mensagem ainda não tenha sido enviada nenhuma vez.

Após o processamento de todas as atualizações de topologia da LSU, é realizada uma limpeza no *buffer* de envio para retirada das atualizações de topologia que já foram reconhecidas (lista de encaminhadores vazia) e enviadas pelo menos uma vez (*send_status* igual a *true*). Assim, conforme podemos ver no exemplo da Figura 22, após o recebimento do reconhecimento do *Mesh Router* B, o *Mesh Router* A não possui mais vizinhos para reconhecer e, como a atualização já foi enviada uma vez, poderá ser removida do *buffer* quando for terminado o processamento das atualizações recebidas na LSU enviada por B.

Embora o MLSD assuma que os enlaces sejam bidirecionais, a percepção da adição ou da perda do enlace é informada pela camada de vizinhança de cada nó, podendo ocorrer em instantes diferentes. Por este motivo, é possível um nó perder temporariamente a vizinhança com outro por um curto intervalo, sem que o outro vizinho detecte que houve perda. Quando isto ocorre entre *Mesh Routers*, um dos *Mesh Router* pode deixar de repassar atualizações de topologia para o outro durante este curto intervalo, já que o nó não era mais seu vizinho, o que pode levar a inconsistências na base topológica.

Para corrigir este problema, os *Mesh Routers* podem perceber quando um enlace com algum *Mesh Router* vizinho foi temporariamente perdido: quando uma atualização de topologia é recebida adicionado o enlace com o vizinho que não havia sido removido. Assim, o vizinho pode disparar um procedimento de correção, realizando uma nova sincronização. O processo de sincronização será descrito na Seção 4.9 e um detalhamento do processo de correção de inconsistências na Seção 4.10.

4.7. Operações de Atualização dos Enlaces

As atualizações de topologia carregam as operações que deverão ser executadas nos enlaces. Quando a operação é executada, as modificações da atualização de topologia são efetivamente registradas na base topológica.

O MLSD define dois tipos de operação: a adição e a remoção. O tipo de operação determina como a atualização da topologia é avaliada para verificar se a atualização é nova ou antiga. A definição se alguma atualização é nova ou antiga é realizada observando se existe informação sobre o enlace na base topológica e no *buffer* de envio. A definição se a

atualização de topologia é nova ou antiga é necessária ao mecanismo de reconhecimento e processamento da atualização já explicado.

As Seções 4.7.1 e 4.7.2 apresentam as operações de adição e remoção, respectivamente, destacando como uma atualização de topologia é avaliada e declarada como nova e como a operação é executada.

As operações também podem disparar procedimentos de correção. Por exemplo, a operação de adição pode disparar uma nova sincronização e a operação de remoção pode realizar a limpeza dos nós inalcançáveis da base topológica quando houver particionamento do *backbone*. Para melhor explicar o procedimento de detecção e correção das inconsistências, é necessário entender as operações e o processo de sincronização. Por este motivo, somente após a explicação das operações e da sincronização é que apresentaremos os procedimentos de correção, na seção 4.10.

4.7.1 Operação de Adição

A operação de adição acrescenta ou atualiza um enlace na base topológica. Contudo, apenas atualizações mais recentes podem ser efetivadas na base. Para determinar se uma atualização de topologia contendo uma operação de adição é nova ou não, é necessário verificar a base topológica e o *buffer* de envio a fim de localizar se existe alguma informação anterior acerca do enlace.

Uma operação de adição é dita nova em duas situações. A primeira ocorre se existir dados sobre o enlace <*MR_addr, neighbor_addr*>, na base topológica, com número de sequência mais antigo. Neste caso, é executada uma atualização das informações de métrica e do novo número de sequência do enlace na base topológica. Além disto, conforme já explicado, toda atualização nova deve ser inserida no *buffer* de envio para ser repassada conforme o mecanismo de reconhecimento positivo implícito.

A segunda situação em que uma atualização de topologia pode ser classificada como nova, ocorre se não existir nenhuma informação sobre o enlace na base topológica e também não existir atualização do enlace com número de sequência maior no *buffer* de envio. Neste caso, a não existência de informações na base, por si só, não determina que um enlace deva ser adicionado, pois alguma atualização com número de sequência maior pode ter removido o enlace e ainda estar presente no *buffer* de envio aguardando o repasse ou reconhecimentos.

Quando não existir dados do enlace na base topológica nem atualizações do enlace com número de sequência maior no *buffer* de envio, então a atualização de topologia é

classificada como nova e deve ser executada a adição inserindo as informações do enlace na base topológica. Conforme já dito, a atualização é repassada conforme mecanismo de reconhecimento positivo implícito.

Em qualquer outra situação a atualização é classificada como antiga. Então, de acordo com o processamento, a mensagem pode ser ignorada, ser um reconhecimento ou provocar a o reenvio de um reconhecimento, conforme já explicado na Seção 4.6.2

4.7.2 Operação de Remoção

A operação de remoção elimina a informação e um enlace na base topológica. De forma similar à operação de adição, quando uma atualização de topologia avaliada for uma operação de remoção, a base topológica e o *buffer* de envio também são verificados para localizar alguma informação acerca do enlace. Desta forma é possível verificar se a atualização de topologia é nova ou não.

Uma operação de remoção é dita nova se existir dados sobre o enlace <*MR_addr, neighbor_addr*>, na base topológica, com número de sequência mais antigo. Neste caso, é executada a eliminação das informações do enlace na base topológica. Assim, quando a atualização é nova ela deve ser inserida no *buffer* de envio para ser repassada conforme o mecanismo de reconhecimento positivo implícito.

Em qualquer outra situação a atualização é classificada como antiga e então, conforme já explicado na Seção 4.6.2, o processamento da mensagem irá determinar se a mensagem pode ser ignorada, tratar-se de um reconhecimento ou provocar o reenvio de um reconhecimento, inserindo novamente a atualização no *buffer* de envio.

4.8. Controle de Inundação de Mensagens Baseado em *Time-Slots*

As IWMNs também suportam *Mesh Clients* móveis, consequentemente clientes móveis podem causar muitas mudanças na vizinhança dos *Mesh Routers*. Quando alguma modificação dos enlaces é informada pela vizinhança de um *Mesh Router*, uma mensagem LSU é gerada e transmitida para os *Mesh Routers* vizinhos, segundo o mecanismo de reconhecimento positivo implícito.

Conforme explicado, no reconhecimento positivo implícito, a garantia de entrega de uma atualização de topologia é obtida quando um *Mesh Router* percebe que os *Mesh Router* vizinhos repassaram a atualização de topologia em suas LSUs, pois a mensagem também tem a função de reconhecimento. Assim, enquanto o *Mesh Router* não detectar que todos os

vizinhos realizaram o repasse, a atualização de topologia é retransmitida indicando os vizinhos, dos quais ainda não se tenha detectado o repasse, como sendo encaminhadores da atualização na nova LSU.

É importante notar que podem ocorrer problemas durante a transmissão, levando os *Mesh Routers* a retransmitir mensagens. Contudo, se o *Mesh Router* repassar uma atualização de topologia em uma mensagem para os *Mesh Routers* vizinhos com sucesso, e receber o reconhecimento de todos os vizinhos com sucesso, não haverá necessidade de retransmissões e, consequentemente, menor será o total de mensagens enviadas.

Os problemas da transmissão são atacados principalmente na camada MAC. Entretanto, nem sempre recursos existentes para tratar estes problemas estão disponíveis. Por exemplo, o padrão 802.11 possui o mecanismo de RTS/CTS para reduzir as colisões de mensagens causadas na situação do nó escondido (*hidden node problem*). Todavia, não faz uso de RTS/CTS quando o quadro é emitido em *broadcast*. Este problema é apresentado no cenário da Figura 23, em que existe a possibilidade de ocorrer colisão: após uma emissão de mensagem do nó A (Figura 23a), os vizinhos C e D, escutam o meio e podem iniciar suas transmissões em instantes muito próximos, provocando colisão da mensagem nos receptores A e B (Figura 23b).

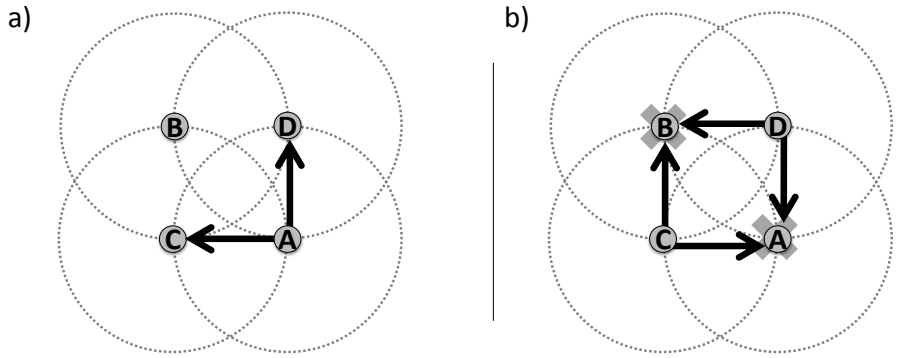


Figura 23 Colisão - nó escondido

As estratégias de divulgação de topologia dos protocolos de roteamento geralmente se preocupam apenas em aplicar um pequeno *jitter* para entregar o pacote à camada MAC, para então enviar as mensagens. O *jitter* é um retardo aleatório, sorteado por cada nó a cada transmissão e tem a função de espalhar o envio das mensagens no tempo. Assim, quando nós diferentes detectam um evento no mesmo instante, eles irão escolher tempos diferentes para iniciar os algoritmos de contenção da camada MAC, auxiliando a reduzir as colisões.

Conforme citado, se o *Mesh Router* repassar a mensagem para os *Mesh Routers* vizinhos com sucesso e receber todos os reconhecimentos dos vizinhos com sucesso, não haverá retransmissões, resultando em menor total de mensagens enviadas. No MLSD, as colisões no repasse ou no reconhecimento da mensagem levam a retransmissões. Por isto, o MLSD emprega uma estratégia diferenciada para espalhar o envio das mensagens antes de entregar para a camada MAC e, com isto, auxiliar na tarefa de reduzir as colisões e consequentemente a necessidade de retransmissões, conforme será explicado a seguir.

4.8.1. *Slots* de tempo

O MLSD define *slots* de tempo, que são intervalos em que o *Mesh Router* pode realizar a transmissão. Estes *slots* são configurados dinamicamente entre os encaminhadores através de cada LSU enviada. A LSU indica em qual *slot* cada encaminhador deve realizar o repasse, de modo que cada *Mesh Router* seja configurado para realizar o repasse em um *slot* distinto, auxiliando a camada MAC a reduzir a chance de colisão.

O tempo definido para o *slot* é um parâmetro configurável do protocolo. Este parâmetro deve ser determinado conforme as características da tecnologia sem fio, considerando a taxa de transmissão e o tamanho da maior mensagem do MLSD possível para a tecnologia adotada. Uma vez definido este valor, o parâmetro deve ser igual para todas as estações utilizando a mesma tecnologia.

Por exemplo, no 802.11b, um valor padrão sugerido é definido pela constante SLOT_INTERVAL, que é igual a 0,03125s. Este valor é calculado dividindo o segundo em potências de 2, conforme apresentado na Figura 24. Assim, para escolher o tamanho do *slot* adequado, devemos calcular o tempo de transmissão do maior quadro do MLSD considerando os tempos de transmissão do preâmbulo e do pacote, além dos IFS do 802.11.

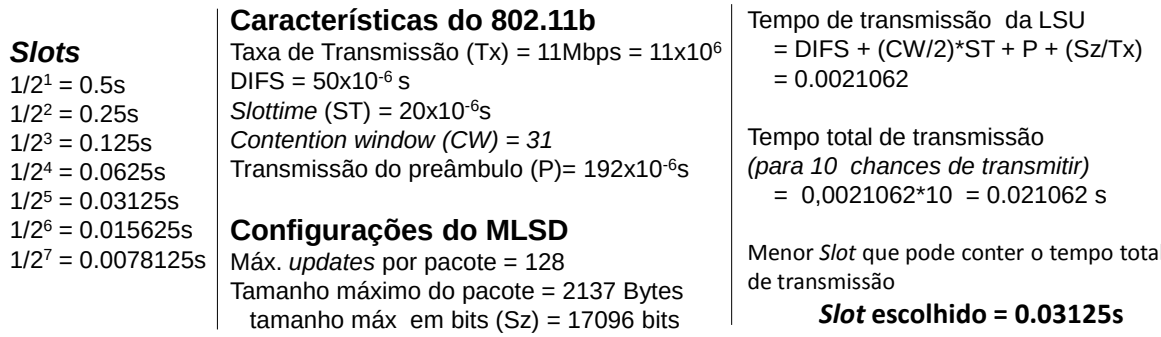


Figura 24 Cálculo do tempo do *slot* para o 802.11b

É importante notar que, durante o período agendado para o *slot* também podem ocorrer transmissões de dados entre os nós. Por isto, o MLSD deve considerar um período maior do que apenas o tempo de transmissão. Assim, para obter o tempo do *slot*, o tempo de transmissão deve ser multiplicado por uma constante. Desta forma, se o meio sem fio estiver ocupado durante o inicio do tempo de transmissão, ainda haverá tempo suficiente, no mesmo *slot*, para que o *Mesh Router* tenha outra chance de enviar a LSU. Então, o tempo do *slot* escolhido deve ser o mínimo necessário para conter o tempo total de transmissão, com um intervalo suficiente para mais de uma tentativa.

4.8.2. Configuração dos *Slots* alocados para envio das mensagens

Um *Mesh Router* que detecta uma atualização de topologia em algum dos seus enlaces deve agendar a transmissão da atualização para ser enviado, em uma LSU, o mais brevemente possível. Em complemento, um *Mesh Router* que recebe uma LSU deve considerar as configurações indicadas na LSU para agendar o envio da LSU no *slot* correspondente.

Como já visto, quando o processamento da atualização de topologia determina que a atualização deva ser enviada, seja para repasse adiante ou apenas para reconhecimento, a atualização é primeiramente inserida no *buffer* de envio. Para inserir uma atualização no *buffer* de envio, além de calcular a lista de encaminhadores para a atualização, dois outros parâmetros precisam ser definidos: o *status* de envio da atualização e um tempo de espera para transmissão. Estes dois parâmetros (tempo de espera e *status* de envio), permitem distinguir, no *buffer*, quais atualizações podem ser recuperadas para ser enviadas na LSU, em um determinado instante.

O *status* de envio (campo *send_status*) é definido sempre que a atualização é inserida no *buffer* de envio com o valor *false*, que indica que a referida atualização nunca foi transmitida. Quando a atualização é recuperada do *buffer* para ser empacotada em uma LSU, o *send_status* deve ser atualizado para *true*.

O tempo de espera para transmissão é computado quando a atualização de topologia é informada pela vizinhança ou recebida através de uma LSU e então é inserida no *buffer*.

Por um lado, se a atualização de topologia for informada pela vizinhança ou, se recebida através de LSU, tratar-se de um pedido de reenvio para reconhecimento, a atualização é agendada para envio no primeiro *Slot*. Em outras palavras, tanto para anunciar alguma modificação da vizinhança quanto para corrigir alguma mensagem perdida, o tempo

de espera para transmissão (campo *time*) que é configurado para atualização de topologia a ser inserida no *buffer* de envio é igual a 0.0 segundos. Vale salientar que isto não significa necessariamente que a LSU seja enviada imediatamente, mas que a atualização já esta pronta para ser enviada na próxima vez que o *buffer* de envio for lido. Desta forma, a atualização será enviada tão logo a próxima LSU seja enviada.

Por outro lado, quando uma atualização de topologia nova é recebida através de uma LSU, o tempo de espera para transmissão é calculado conforme o *slot* alocado para o encaminhador. O encaminhador que recebe a LSU determina em qual *slot* está alocado para realizar o repasse através da posição em que seu endereço aparece na lista de encaminhadores da LSU. O primeiro elemento da lista de encaminhadores é indicado na posição 1.

Desta maneira, conforme podemos ver na Figura 25, quando os *Mesh Routers* C e D recebem uma LSU enviada por A (Figura 25a), eles recuperam a posição do seu endereço na lista de encaminhadores (D=1 e C=2). Cada *Mesh Router*, ao inserir uma atualização de topologia no *buffer* de envio, calcula o tempo de espera de transmissão (campo *time*) para um *slot* diferente (Figura 25b). Assim, enquanto o primeiro da lista (MR-D), agenda seu repasse para o próximo *slot*, o segundo na lista (MR-C), agenda seu repasse para o *slot* seguinte. Desta forma, a atualização de topologia de cada *Mesh Router* somente será despachada em uma LSU no *slot* configurado, reduzindo a chance de colisões.

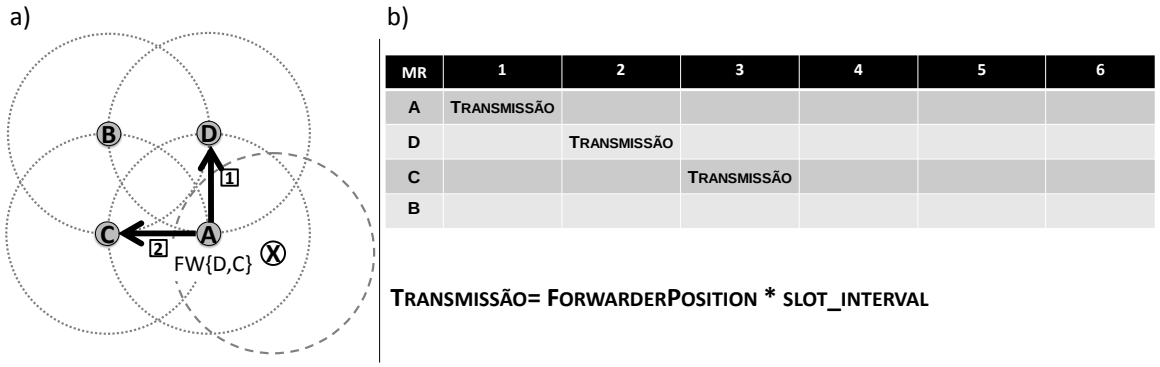


Figura 25 Configuração do *slot* dos encaminhadores pela LSU

É importante perceber também que um nó pode receber a mesma atualização de diversos vizinhos. Por exemplo, na Figura 25a, quando os *Mesh Routers* C e D publicarem as mensagens que devem ser recebidas pelo *Mesh Router* B, eles podem informar uma posição diferente para o *Mesh Router* B, na lista de encaminhadores de suas respectivas LSUs. Neste caso, o *slot* efetivamente configurado para repasse da atualização por B é o da primeira LSU

recebida, pois, ao receber a segunda mensagem, a atualização de topologia já não será mais nova e servirá apenas de reconhecimento antecipado.

Para enviar as mensagens é necessário recuperá-las do *buffer* de envio. Por isto, um temporizador continuamente dispara a leitura do *buffer* de envio para verificar se existem atualizações de topologia a serem enviadas e então gerar a LSU.

Quando o *buffer* de envio está vazio, é necessário que o *buffer* seja verificado com maior frequência para evitar atrasos para enviar mensagens que devem ser publicadas no primeiro *slot*. Desta forma, o temporizador é reajustado para disparar novamente em um tempo obtido pela divisão do valor de SLOT_INTERVAL por um parâmetro, cujo valor padrão sugerido é definido na constante CHECK_BUFFER_FREQUENCY que é igual a 10. Como efeito, durante o tempo de um *slot*, o *buffer* é verificado diversas vezes para detectar se existe alguma mensagem a ser enviada.

Assim, enquanto o *buffer* estiver vazio, uma nova leitura do *buffer* para verificar se foram inseridas novas atualizações de topologia para serem enviadas é agendada para SLOT_INTERVAL/CHECK_BUFFER_FREQUENCY segundos, acrescido de pequeno *jitter*. O *jitter* acrescido ao intervalo do temporizador é um valor aleatório sorteado entre 0 e 0,25*(SLOT_TIME_INTERVAL/CHECK_BUFFER_FREQUENCY), e é aplicado para reduzir a chance de colisão quando *Mesh Routers* detectam vizinhos no mesmo instante.

Desta maneira, enquanto o *buffer* de envio estiver vazio, nenhuma mensagem é enviada e no temporizador é agendada uma nova leitura do *buffer* em um intervalo aleatório de 0,00312500 a 0,00390625 segundos a partir do tempo atual. Como efeito, os eventos ocorridos durante o intervalo configurado no temporizador são acumulados e serão enviados na mesma mensagem LSU.

Sempre que o temporizador é disparado, a leitura do *buffer* de envio também atualiza o campo *time* em cada atualização de topologia, decrementando o seu valor atual pelo tempo transcorrido desde a última vez que temporizador disparou, para então recuperar as atualizações de topologia que serão enviadas. Somente são enviadas as atualizações de topologia com campo *time* igual ou inferior a 0.0 segundos. Para as atualizações que serão enviadas o valor do campo *send_status* é atualizado para *true*.

A LSU é formada com as atualizações recuperadas do *buffer*. Conforme explicado na Seção 4.6.2, a ordem de ocorrência das atualizações do *buffer* não é necessária ao MLSD. Por isto, antes de recuperar as atualizações de topologia, uma reordenação do *buffer* é realizada. A

reordenação sequencia as atualizações dos enlaces para cada *Mesh Router*, primeiramente listando as adições e as remoções de *Mesh Clientes* e em seguida as adições e remoções de *Mesh Routers*. Desta forma, consegue-se melhor aproveitamento do formado do pacote LSU, reduzindo o tamanho do pacote.

Com as atualizações reordenadas recuperadas do *buffer*, uma nova leitura é realizada para construir a lista de encaminhadores, e preencher as estruturas da LSU (LSAs). A partir da lista de encaminhadores da LSU e das atualizações recuperadas do *buffer*, é gerado o *bitmap* definindo para cada atualização se o encaminhador da lista deve ou não estar marcado.

Conforme mostrado na Seção 4.3, o parâmetro configurado para o número máximo de atualizações por LSU é igual a 128. Este valor é definido na constate MAX_UPDATE_IN_LSU, e deve ser calculado conforme a tecnologia usada observando o tamanho da MTU. Se a quantidade de atualizações de topologia a serem enviadas for superior a MAX_UPDATE_IN_LSU, será preciso enviar tantas LSUs quanto forem necessárias para enviar todas as atualizações. Este envio é realizado em rajada. Isto é, todas as LSU são entregues a camada MAC imediatamente. Neste caso, a lista de encaminhadores deve ser igual em todas as LSUs da rajada. No final, após o envio das LSUs é realizada a limpeza do *buffer*, eliminando mensagens já reconhecidas (lista de encaminhadores vazia) e enviadas ao menos uma vez (campo *send_status* igual a *true*).

As atualizações de topologia recuperadas para serem enviadas (*send_status* igual a *true*) que ainda a aguardam reconhecimentos pendentes, não devem ser removidas durante a limpeza do *buffer*. No entanto, o valor do campo *time* é redefinido com um tempo de espera para retransmissão, que por simplicidade será chamado apenas de tempo de retransmissão.

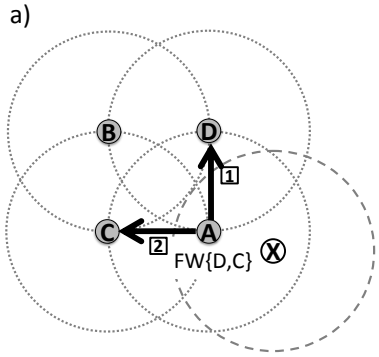
O tempo de retransmissão deve ser suficiente para receber os reconhecimentos de todos os encaminhadores. Se todos os encaminhadores realizarem o repasse, após o recebimento da última mensagem de reconhecimento, conforme já explicado na seção 4.6.2, a atualização de topologia é automaticamente limpada do *buffer*. Caso contrário ela é retransmitida.

Além disto, quando o número de atualizações por mensagem for superior ao limite configurado, como em situações de grande ocorrência de eventos, outras LSUs devem ser criadas e despachadas imediatamente, em rajada. Assim, o tempo do *slot* pode não ser suficiente para despachar todas as LSUs. Neste caso, um multiplicador pelo número de

mensagens aumenta o tempo do *slot*. Consequentemente, com maior tempo disponível, os *Mesh Routers* podem enviar mais mensagens, em rajada.

Assim, o tempo de retransmissão é calculado com base no número de encaminhadores da LSU enviada (para determinar o número de *slots* que serão aguardados) e o número de mensagens necessárias para enviar todas as atualizações recuperadas do *buffer*. Este tempo para retransmitir a atualização é suficientemente grande para conter os repasses de todos os vizinhos. Como podemos ver na Figura 26b, o MR-A reagenda sua retransmissão para um *slot* após todos os vizinhos realizarem seus repasses.

a)



b)

MR	1	2	3	4	5	6
A	TRANSMISSÃO				RETRANSMISSÃO	
D		TRANSMISSÃO				
C			TRANSMISSÃO			
B						

TRANSMISSÃO= FORWARDERPOSITION * SLOT_INTERVAL

RETRANSMISSÃO = (NUM_FORWARDERS +2) * SLOT_INTERVAL *#LSUS

Figura 26 Configuração do *slot* para retransmissão da atualização de topologia

O número de *slots* aguardados deve considerar que a LSU só deverá ser repassada por pelos encaminhadores a partir do próximo *slot* e pode ser recebida em qualquer momento do último *slot.* Logo, o número de *slots* que devem ser aguardados é igual a NUM_FORWARDERS+2.

O número de LSUs enviadas funciona como um multiplicador do *slot* quando for necessário enviar mensagens em rajada, pois neste caso o *Mesh Router* precisa de mais tempo para enviar mais mensagens e de maior tamanho e assim reduzir as colisões. Entretanto, um parâmetro configurável define um valor limite para o multiplicador do *slot*, cujo valor padrão é definido na constante MAX_SLOT_MULTIPLIER e é igual a 5. Evitando, assim, que o tempo do *slot* torne-se muito longo, ao mesmo tempo em que permite a divulgação de um grande número de atualizações.

Desta forma, o tempo de retransmissão é dado por (NUM_FORWARDERS +2)*SLOT_TIME_INTERVAL*(número de LSUs), se o número de LSUs enviadas for menor que MAX_SLOT_MULTIPLIER ou (NUM_FORWARDERS+2)*SLOT_TIME_INTERVAL*MAX_SLOT_MULTIPLIER caso contrário. Com isso, considerando o cenário usual com 4 encaminhadores, caso o número de

LSUs a serem enviadas seja superior a 5 (i.e. mais de 512 atualizações de topologia), então tempo de retransmissão para a atualização de topologia será igual a 0,9375 segundos.

Além do campo *time*, o temporizador é reagendado para disparar novamente também para o tempo de retransmissão. Isto é, nenhuma outra LSU é enviada durante este tempo, e as atualizações de topologia recebidas serão acumuladas.

É importante observar que desta forma, mesmo as atualizações informadas pela vizinhança que são inseridas no *buffer* com *time* igual a 0.0 só serão enviadas quando o temporizador disparar novamente. Contudo, é importante também notar que durante o tempo de espera do temporizador outros *Mesh Routers* estão enviando mensagens. Portanto, ao retardar o repasse, evitam-se as colisões que poderiam provocar ainda mais perdas, mais atrasos e aumento do total de mensagens.

4.9. Sincronização

Se um *Mesh Router* for adicionado ao *backbone* é necessário que ele conheça todos os enlaces da rede. Por isto, quando um *Mesh Router* detecta outro *Mesh Router* ele deve realizar a sincronização da base topológica.

A sincronização é iniciada pelo *Mesh Router* que detecta um enlace com um *Mesh Router* vizinho. O *Mesh Router* que detectou o enlace reconstrói todas as atualizações de topologia registradas na base e as insere no *buffer* informando o *Mesh Router* descoberto como encaminhador das atualizações e campos *send_status* definido como *false* e *time* definido como 0.0. A reconstrução da atualização de topologia é possível porque o *Mesh Router* armazena o número de sequência registrado com a atualização na base.

As atualizações reconstruídas possuem apenas a operação de adicionar, pois nenhuma informação é mantida sobre enlaces removidos. Elas seguem o mesmo processo de entrega das atualizações convencionais. É importante destacar que ao receber as atualizações na LSU, o *Mesh Router* vizinho avaliará cada uma conforme o mecanismo de reconhecimento para saber se deve ou não divulgar adiante, atualizando outros *Mesh Routers* do *backbone*.

É possível que o envio da LSU seja realizado antes que o outro *Mesh Router* detecte o vizinho. Isto é, o *Mesh Router* vai receber uma LSU que tem seu endereço na lista de encaminhadores, mas de um *Mesh Router* que não é seu vizinho, ainda. Neste caso, antes de processar as atualizações da LSU recebida, o MLSD se antecipa a detecção da vizinhança: cria e processa uma atualização de topologia de seu enlace com o *Mesh Router* vizinho e

insere no *buffer* de envio junto com as atualizações existentes em sua base, indicando o vizinho novo como encaminhador. Desta forma, as atualizações recebidas pela LSU que já existirem na base do *Mesh Router* pode servir de reconhecimento antecipado.

A Figura 27 apresenta o processo de sincronização: quando o *Mesh Router B* é informado da vizinhança com o *Mesh Router A*, ele *insere* no *buffer* de envio todas as atualizações registradas na base topológica indicando *A* como encaminhador e envia através da LSU (Figura 27a). Assim, quando *A* recebe a LSU de *B* mesmo que este ainda não seja seu vizinho, *A* se antecipa à camada de vizinhança adicionando *B* e inserindo as atualizações da própria base no *buffer* para enviar junto com os reconhecimentos das atualizações enviadas por *B*(Figura 27b). Quando *B* receber a LSU enviada por *A*, *B* reconhece as atualizações enviadas e envia o reconhecimento das atualizações recebidas.

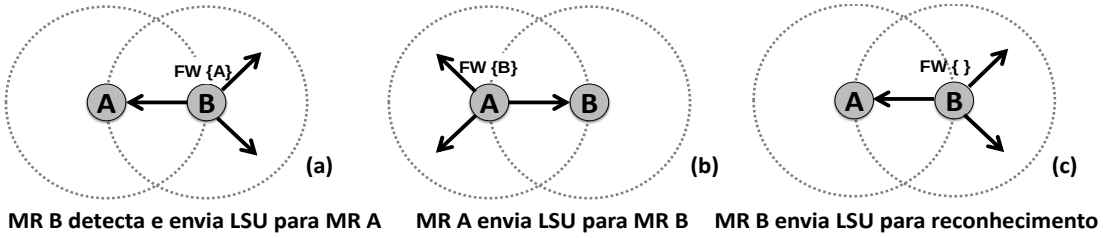


Figura 27 Sincronização das bases topológicas

Se durante o processo de sincronia o *Mesh Router* detectar um novo *Mesh Router* vizinho, além da atualização nova que deve ser gerada e enviada informando o novo enlace entre os vizinhos, as atualizações da base são recuperadas e inseridas no *buffer* de envio, no entanto, as atualizações já existentes no *buffer* não são reinseridas, apenas a lista de encaminhadores é atualizada, acrescentando o novo nó como encaminhador e os campos *send_status* e *time* são redefinidos para *false* e 0.0 novamente, garantindo com isto que a atualização será enviada tão logo a próxima LSU seja enviada.

4.10. Detecção e Correção de Inconsistências

Quando ocorre a ruptura de enlace entre *Mesh Routers* pode ocorrer particionamento do *backbone*, se os *Mesh Routers* envolvidos forem os únicos responsáveis por conectar uma região do *backbone* a outra. Neste caso, as informações dos nós inalcançáveis precisam ser removidas da base topológica.

Para detectar a ruptura do *backbone*, sempre após ser executada uma atualização de topologia com operação de remoção de um enlace entre *Mesh Routers*, um teste de

conectividade deve ser realizado. Este teste é realizado criando um conjunto de *Mesh Routers* conectados com as informações existentes na base topológica, a partir do próprio nó, incluindo recursivamente todos os vizinhos. Assim, os *Mesh Routers* da base topológica que não são parte do conjunto conectado são declarados inalcançáveis e todos os seus enlaces são removidos. Desta forma, uma única atualização de topologia de remoção limpa toda a base topológica dos nós inalcançáveis em todos os *Mesh Routers* do *backbone*.

Quando a camada de vizinhança de cada nó perceber a adição ou a perda do enlace com o vizinho em instantes diferentes, conforme dito na seção 4.6.2, algumas inconsistências podem ser geradas: se um *Mesh Router* perder o enlace com um *Mesh Router* vizinho, mesmo que por um intervalo pequeno, ele deixará de esperar reconhecimentos do vizinho e não o indicará como encaminhador de nenhuma atualização nova.

Se o *Mesh Router* vizinho for o responsável por conectar uma região do *backbone*, toda a região deixará de receber as atualizações durante este período. Um exemplo deste problema pode ser visto na Figura 28a e Figura 28b em que o *Mesh Router B* perde temporariamente o enlace com *C* e, por isto, a atualização de remoção do enlace do *Mesh Router A* com *Mesh Client X* não foi repassada indicando *C* como encaminhador.

Neste caso, quando o *Mesh Router B* recuperar a vizinhança com o *Mesh Router C* ele irá iniciar o processo de sincronização enviando a base para o *Mesh Router* vizinho. Entretanto, como *C* não perdeu a vizinhança com *B*, é necessário outra forma de fazer *C* perceber que o enlace foi perdido e disparar a sincronização.

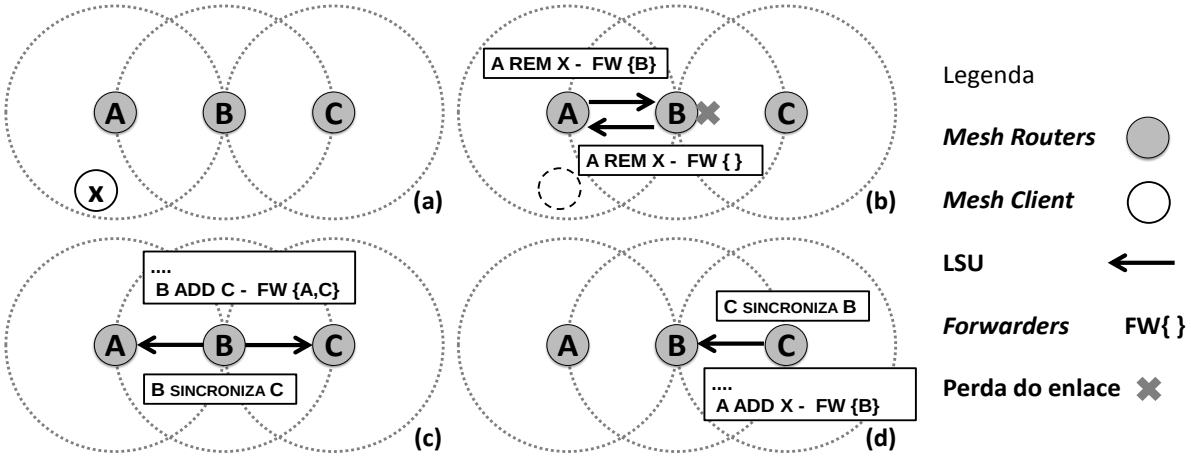


Figura 28 Troca das bases

Para o *Mesh Router C* perceber se perdeu temporariamente a vizinhança com o *Mesh Router B* e por esta razão pode ter deixado de receber alguma atualização de topologia, basta

que *C* perceba uma atualização de topologia de um vizinho readicionando o enlace. Assim, quando for recebida uma atualização de topologia nova, com operação de adição de um enlace com algum *Mesh Router* vizinho, e já houver na base a informação do enlace com número de sequência mais antigo, então o vizinho está sendo readicionado, como podemos ver na Figura 28c, em que *C* recebe uma mensagem de *B* readicionando o enlace.

Desta forma, *C* percebe que *B* perdeu o enlace temporariamente. Então um novo processo de sincronização deve ser disparado. Isto é, toda a base de *C* deve ser enviada de volta ao *Mesh Router B* (Figura 28d). Desta maneira, ambas as bases serão trocadas.

Contudo, se houver alguma remoção durante o intervalo em que a vizinhança é temporariamente perdida a atualização não é repassada, pois no processo de sincronização apenas as atualizações de topologia com operação de adição são enviadas. Portanto, podem restar informações de um enlace que deveria ter sido removido na base do vizinho. Quando a troca das bases for realizada, o enlace que não foi removido da base será reenviado ao vizinho. Como podemos ver ainda na Figura 28d, como não existe informação do enlace na base de B, a atualização do enlace incorreta enviada por C será processada como nova e repassada, provocando inconsistência nas bases, como podemos ver na Figura 29a.

Este problema deve ser consertado pelo *Mesh Router* que gerou a atualização do enlace. A correção é realizada quando um *Mesh Router* receber uma atualização de topologia sobre um de seus enlaces (o endereço do *Mesh Router* esta no campo *MR_addr* da atualização), em que esta marcado como encaminhador, como apresentado na Figura 29a. Assim, a atualização não deve ser executada. O *Mesh Router* deve avaliar se existe informação sobre o enlace e gerar uma nova atualização de topologia de remoção e envia-la para correção do *backbone* (Figura 29b, Figura 29c e Figura 29d).

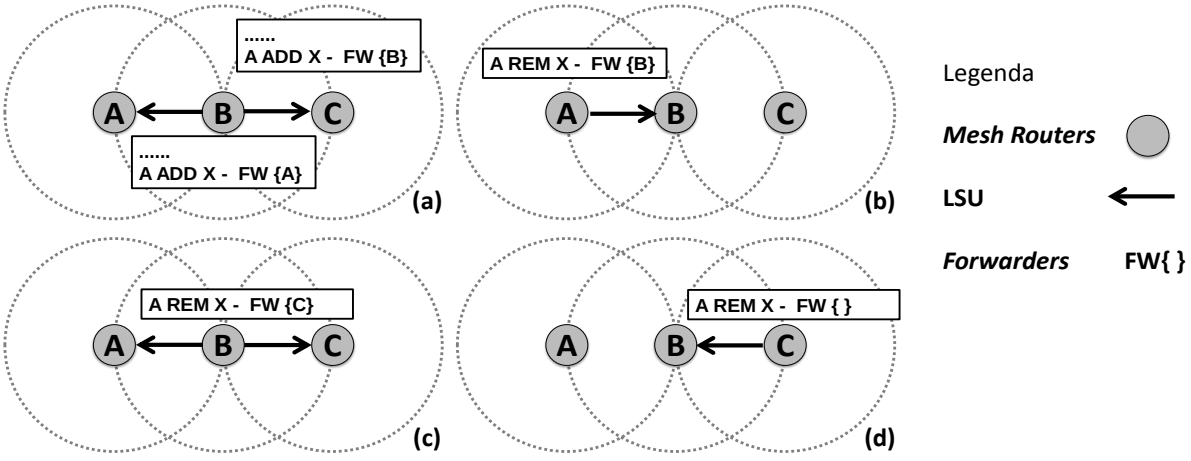


Figura 29 Operação de adição dispara processo de correção

Vale ressaltar que as situações em que ocorrem estes eventos que provocam inconsistências são raras, pois nas RMSF os enlaces entre os *Mesh Routers* são considerados mais estáveis. Ainda, se não houve nenhuma remoção durante o intervalo em que a vizinhança foi reestabelecida, nenhuma inconsistência é gerada e nenhuma atualização precisa ser repassada. O *Mesh Router* vizinho irá receber e enviar as mensagens para reconhecimento, normalmente. Além disto, para que a situação da correção do enlace aconteça é preciso que ocorra uma operação de remoção em um instante muito pequeno, dependendo apenas do tempo em que a camada de vizinhança percebe a perda do enlace. Sendo assim, estas estratégias adicionais permitem que o MLSD mantenha a consistência das informações, e as bases topológicas permaneçam iguais em todos os *Mesh Routers* do *backbone*.

4.11. Números de Sequência

No MSLD, os números de sequência registram a ordem em que as atualizações foram percebidas pelo *Mesh Router*. O número de sequência é divulgado na LSU em conjunto com a informação do enlace <*MR_addr, neighbor_addr*> para cada atualização de topologia e são usados no processamento para determinar se a atualização é nova para um enlace específico. Assim, as mensagens antigas podem ser substituídas e descartadas.

Cada *Mesh Router* mantém um número de sequência que é inicializado com 0 quando o dispositivo é ligado. Ao perceber a primeira modificação, uma atualização é gerada com o número 1 para ser divulgada, e assim sucessivamente.

O valor máximo para o número de sequência é definido na constante MAX_SEQ_NUMBER, cujo valor é 32768. No entanto, quando é montado o pacote LSU, uma estratégia adotada permite reduzir a quantidade de bits gastos com a representação do número de sequência em cada *neighbor_data*.

Conforme explicado na seção 4.8, para montagem da LSU o *buffer* de envio é reordenado. A reordenação sequencia as atualizações dos enlaces para cada *Mesh Router*, primeiramente listando as adições e as remoções de *Mesh Clientes* e em seguida as adições e remoções de *Mesh Routers*. Esta ordenação reflete a estrutura da LSU, isto é, como as LSAs são organizadas. Desta forma, para as LSA de adição de *Mesh Client*, que serão as primeiras da lista, todas as atualizações (*neighbor_data*) estão ordenadas por número de sequência.

Assim, ao construir a LSU, a primeira *neighbor_data* da *LSA_operation* o número de sequência *seq_number* é representado por 16 bits. Todavia, para as *neighbor_data* da mesma

LSA_operation seguintes, o valor de *seq_number* representa o deslocamento do número de sequência calculado com relação a última *neighbor_data* inserida. Se o deslocamento for menor que 127, então o tamanho do campo *seq_number* ocupa apenas 8 bits, se for maior que 127 então o campo *seq_number* ocupará 16 bit e apenas os 15 bits mais a direita representam o deslocamento do número de sequência.

De forma análoga, ao receber a LSU o número de sequência da atualização é facilmente reconstruído.

Quando o *Mesh Router* alcançar o valor limite do número de sequência (MAX_SEQ_NUMBER), ele deve ser reiniciado. Para isto, o *Mesh Router* deve divulgar todos os enlaces com seus vizinhos informando o número de sequência igual a 0, incluindo todos os vizinhos como encaminhadores das atualizações. O valor zero, quando recebido através de uma atualização de topologia, representará a informação mais nova para cada enlace e irá substituir a atualização registrada nas bases topológicas dos outros *Mesh Routers* do *backbone*. Por isto, a única situação em que o valor 0 é divulgado em uma atualização de topologia é quando o número de sequência do *Mesh Router* é reiniciado.

4.12. Considerações finais

O MLSD proposto nesta dissertação é um protocolo construído para a camada de topologia da IWMRA e como tal, preocupa-se especificamente em como divulgar e manter consistentes as informações da topologia da rede da melhor forma possível. Os principais objetivos do MLSD são reduzir o total de mensagens empregada nas atualizações dos enlaces e reduzir a carga das mensagens transmitidas, quando comparado a protocolos que divulgam informações periodicamente.

Para alcançar seus objetivos o MLSD utiliza uma estratégia orientada a eventos, enviando mensagens somente quando necessário, nas quais são divulgadas apenas as atualizações incrementais. Além disto, o MLSD adota um formato compacto para representar as informações na mensagem.

As mensagens são enviadas com garantia de entrega e em conjunto com uma estratégia de sincronização asseguram a consistência das informações do *backbone*. Em adição, o MLSD implementa uma estratégia de *flooding* mais eficiente, separando os momentos de transmissão entre vizinhos em *slots* de tempo distintos para auxiliar a camada MAC a evitar colisões.

Capítulo 5

O Ambiente de Simulação

Este capítulo descreve como foi realizado o desenvolvimento do MLSD e a metodologia adotada para analisar o comportamento dos protocolos MLSD e OLSR. Além disto, serão apresentados: o simulador, a implementação de referência do OLSR, o cenário de simulação e definidos os parâmetros quantitativos da avaliação de desempenho dos protocolos.

Para analisar o comportamento dos protocolos de rede tradicionalmente são usadas duas estratégias [Johnson 2007]: uma é a utilização de cenários reais, chamados "plataforma de testes de redes" (*live network testbeds)* e a outra é a adoção de emuladores de redes, chamada "plataforma de testes de emulação" (*emulation testbeds*).

A plataforma de testes de rede fornece condições reais para realização do experimento, com resultados concretos. No entanto, a aplicação desta metodologia tem um custo elevado, os experimentos não são facilmente reproduzíveis e, ainda, permitem menor controle do usuário sobre o experimento. Por sua vez, a plataforma de testes de emulação consiste em simular a operação da rede, em condições artificiais, em ambiente de software. Esta estratégia tem como vantagens o custo menor, permitir que os experimentos sejam mais facilmente reproduzíveis e fornecer ao usuário grande controle das condições do ambiente da rede e do *host*. A plataforma de testes de emulação é a estratégia que será adotada para este trabalho.

Entre os vários simuladores disponíveis para realizar testes através de emulação, o *Network Simulator 2* (NS-2) foi escolhido por disponibilizar uma extensa documentação, ter um grupo de desenvolvimento bastante ativo no esforço de desenvolvimento de correções e funcionalidades e por ser muito utilizado em diversos trabalhos científicos.

5.1. *Network Simulator 2*

O NS-2 é um simulador de eventos discretos projetado para a pesquisa em redes de computadores, desenvolvido como parte do projeto VINT (*Virtual InterNetwork Testbed*) e

mantido pela agência DARPA (*Defense Advanced Reserach Projects Agency*), do governo americano. O Projeto VINT é composto por um grupo de pesquisadores da UC Berkeley, LBL, USC/ISI e Xerox PARC [Fall 1997].

O NS-2 é baseado em duas linguagens [Altman 2003]: a linguagem C++, usada para desenvolver um simulador orientado a objetos, e a linguagem OTcl, usada em um interpretador que executa os *scripts* do usuário. A linguagem OTcl é uma extensão orientada a objetos da linguagem de *scripts* Tcl. Através de uma biblioteca rica em protocolos, tipos de redes, aplicações, elementos da rede e padrões de tráfego, que são chamados objetos de rede, o NS-2 pode compor diversos cenários de rede. Estes objetos estão descritos em duas hierarquias de classes. Uma é compilada em C++ e outra descrita em OTcl. Para cada classe C++ da API do NS-2, existe uma correspondente em OTcl. A ligação entre um objeto C++ e um objeto OTcl é realizada através da interface TclCL.

A implementação do simulador em linguagem C++ proporciona eficiência através da execução rápida de tarefas como manipulação de grande quantidade de dados, processamento de pacotes e de eventos da simulação. A interface com o usuário na linguagem OTcl proporciona flexibilidade ao simulador, permitindo a fácil modificação da configuração da simulação, alteração dos parâmetros dos protocolos, topologia, dispositivos e tipos de tráfego.

A Figura 30 mostra uma visão geral do uso do simulador NS-2, em que é possível observar que o *script* de configuração da simulação, desenvolvido em OTcl, é processado pelo interpretador que realiza o casamento dos objetos OTcl com o simulador em C++, instanciando os elementos da rede e agendando os eventos discriminados no *script*. O processamento realizado pelo simulador resulta na geração dos arquivos contendo todos os rastros dos eventos gerados durante a simulação (*Trace*) e o arquivo NAM.

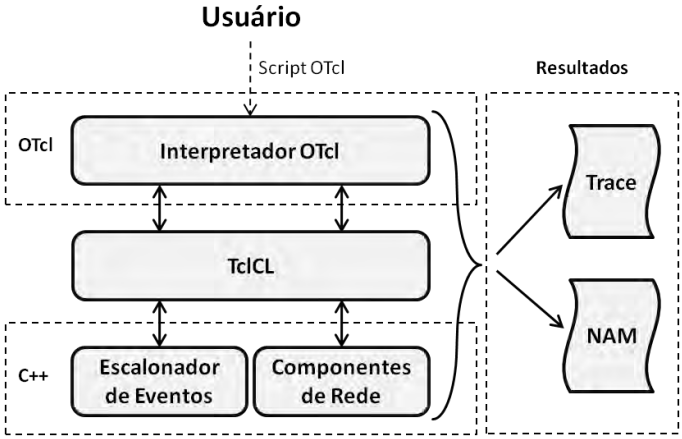


Figura 30 Visão geral do NS2

O arquivo *Trace* gerado deve ser processado via *scritps* criados pelo usuário para obter os dados para análise e com isto poder realizar a avaliação. Os *scripts* criados pelo usuário podem ser desenvolvidos em qualquer linguagem. Neste trabalho, foram desenvolvidos *scripts* em PHP para analisar os arquivos de rastro e Bash para automatizar a execução das simulações.

O arquivo NAM é utilizado pelo programa *Network Animator* (NAM) para reproduzir a simulação, permitindo observar os eventos em uma linha de tempo num ambiente gráfico. O NAM é usado basicamente para depuração. Através dele é possível observar a topologia da rede, o movimento realizado pelos nós, e as transmissões de pacotes na rede. O NAM também é uma ferramenta desenvolvida no projeto VINT e pode ser obtido junto com o NS-2.

5.2. Implementação do MLSD

A criação de um novo protocolo no NS-2 se inicia estendendo a classe *Agent* da API do NS-2 [Ros 2004]. Através da classe *Agent* é possível recuperar os dados da camada MAC, para poder realizar algum processamento e decidir se entrega os dados para camadas superiores ou reenvia através da rede.

A classe *IWMRAAgent* estende a classe *Agent* sobrecarregando dois métodos herdados: o método *recv*, que é o ponto de entrada dos pacotes recebidos pelo nó, e o método *command*, que executa comandos agendados no escalonador descritos no *script* OTcl de configuração da simulação. Através do método *recv*, a instância do agente *IWMRAAgent* recupera o quadro da camada MAC e examina o datagrama recebido para determinar a que camada da IWMRA ele pertence. Quando o pacote do MLSD é identificado, o *handler* do MLSD realiza o restante do processamento.

Além da extensão da classe *Agent*, também é necessária a modificação no código do NS-2 para que o simulador reconheça o protocolo criado. As alterações no código do NS-2 necessárias ao reconhecimento do novo protocolo são: a adição do tipo do novo pacote na lista de pacotes conhecidos pelo simulador; adição do tipo do novo pacote na lista de pacotes na fila de prioridade; formatação da impressão dos registros de eventos de envio e recebimento de pacotes no arquivo de saída (*Trace*); criação do mapeamento da classe que implementa o protocolo em C++ com a interface OTcl e edição do *Makefile* para compilar o novo protocolo junto com o NS-2.

O protocolo MLSD é implementado em quatro classes principais: *MLSDHandler*, *MLSDLSAHandler*, *MLSDSyncHandler e MLSDTopologyTable*. Quando um nó recebe um pacote, o método *recv* da classe *IWMRAAgent* verifica se é destino da mensagem e, neste caso, determinar a que camada pertence. Os datagramas da camada de topologia (LSUs), são entregues a classe *MLSDHandler*.

A classe *MLSDHandler* é responsável por receber, gerar e enviar as LSUs, receber as atualizações da camada de vizinhança, executar as operações de topologia e atualizar os tempos de transmissão e retransmissão. A classe *MLSDLSAHandler* é a responsável por realizar o processamento das atualizações de topologia, o mecanismo de reconhecimento positivo implícito e gerenciar o *buffer* de envio. A classe *MLSDSyncHandler* é a responsável por recuperar as atualizações da base topológica utilizadas na sincronização. A classe *MLSDTopologyTable* implementa a base que armazena as informações topológicas além de realizar a manutenção da base, por exemplo removendo nós inalcançáveis.

O ambiente de desenvolvimento é implementado através de máquina uma virtual *VMWare*, executando o sistema operacional Linux *Debian Lenny* (5.03), a IDE Eclipse CDT, controle de versões realizado com *subversion*, a suíte de compiladores *gcc/g++* versão 4.01, depurador *gdb* e ferramenta de inspeção de memória *Valgrind*. Este ambiente também é usado também para realizar as simulações para avaliação de desempenho. Desta forma, adotamos uma estratégia que pode ser muito facilmente reproduzida por qualquer pesquisador que queira avaliar o protocolo, bastando para tanto iniciar a máquina virtual e executar o simulador dispensando procedimentos complicados de instalação.

5.3. Implementação de Referência do OLSR

Conforme dito, o NS-2 possui uma vasta biblioteca de objetos de rede, incluindo protocolos de roteamento. No entanto, os protocolos atualmente disponíveis não são usados em RMSF ou não possuem uma estratégia que possa ser comparada ao MLSD.

Por este motivo, é necessário inserir no NS-2 uma implementação de referência do OLSR para que a avaliação possa ser realizada. A implementação do OLSR mantida pelo projeto MASIMUM (*MANET Simulation and Implementation at the Univeristy of Murcia*) chamada de UM-OLSR [Ros 2010], foi escolhida porque está em conformidade com a RFC 3626, suportando todas as funcionalidades do OLSR. Além disto, esta implementação já foi também usada em vários outros trabalhos, por exemplo, em [Santos 2008], [Pore 2006] e [Huang 2006].

A versão do UM-OLSR utilizada foi a 0.8.8. Nesta implementação o OLSR permite por padrão a divulgação de até 4 mensagens no mesmo pacote, e assim, podem ser enviadas até 4 mensagens TC. Cada TC divulga os enlaces do MPR com até 64 vizinhos. Ou seja, o OLSR pode divulgar enlaces de até 4 MPRs com 64 enlaces cada, totalizando 256 atualizações de enlaces em um único pacote. Se o número de TCs necessárias para divulgar e repassar as atualizações de topologia for superior a quatro, as mensagens devem ser divulgadas em outros pacotes do OLSR em rajada.

5.4. Cenário de Simulação

Para realizar as simulações dos protocolos MLSD e OLSR, foi projetado um cenário de uma rede em malha sem fio infraestruturada. Este cenário define uma topologia composta de *Mesh Routers* posicionados estrategicamente para prover uma área de cobertura contínua onde os *Mesh Clients* podem mover-se livremente, sem perder conectividade com o *backbone*.

A disposição dos *Mesh Routers* que compõem o *backbone* corresponde a uma topologia na forma de matriz, em linhas e colunas, em que os dispositivos estão separados pela distância do alcance máximo das suas interfaces de rede sem fio, que é igual a 100m.

Os *Mesh Routers* são vizinhos dos *Mesh Routers* da vertical e da horizontal, mas não são vizinhos dos *Mesh Routers* das diagonais, pois a distância entre eles é maior que o alcance da interface de rede, impedindo-os de se comunicar diretamente. O número de *Mesh Routers* que compõe o *backbone* é igual a 100, isto é, a topologia dos *Mesh Routers* tem a forma de uma matriz 10x10. Este *backbone* foi utilizado em todas as simulações realizadas.

Dentro da área do *backbone* formado pelos *Mesh Routers* é definida uma área de cobertura total onde os *Mesh Clients* podem movimentar-se, provocando os eventos que irão gerar os dados do arquivo *Trace* para serem posteriormente analisados. Esta área tem a forma de um quadrado de 1,04Km de lado, ou seja, a área total é igual a 1,0816Km^2. Este cenário pode ser visualizado na Figura 31, em que é possível destacar a disposição dos nós que constituem o *backbone* na forma de matriz e a definição da área de cobertura total formada pelos *Mesh Routers*. É possível notar também a presença de *Mesh Clients* parados ou em movimento, dentro da área de cobertura total.

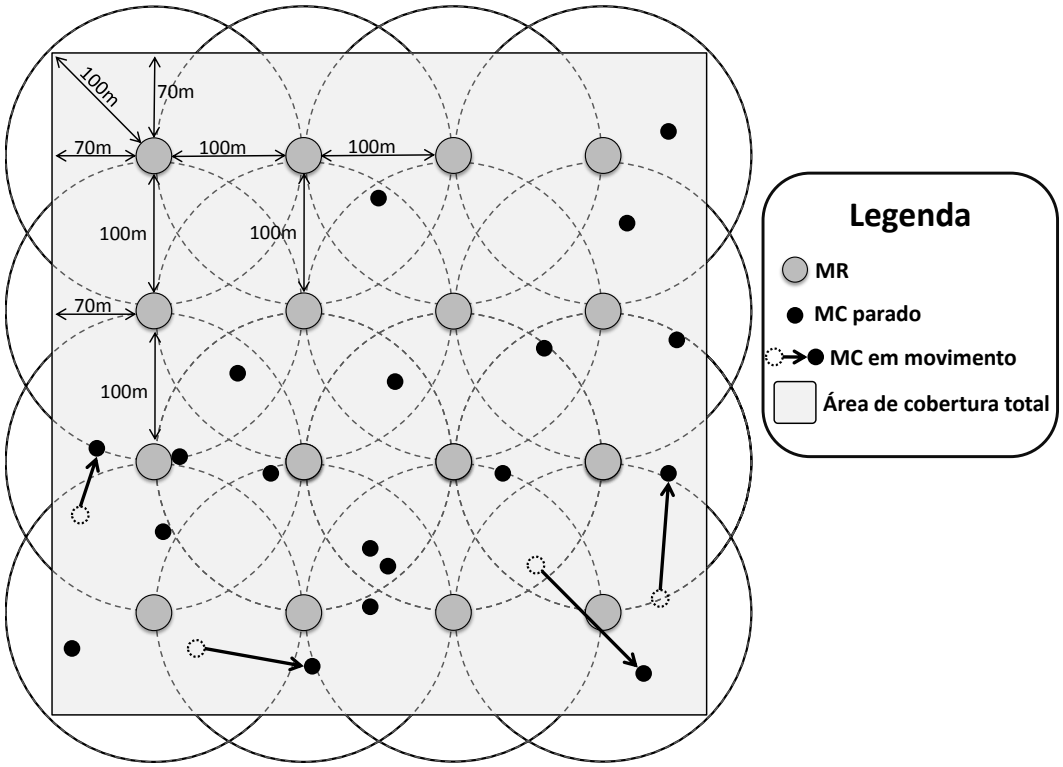


Figura 31 Cenário utilizado nas simulações

Todos os nós da malha possuem uma única interface sem fio que são configuradas conforme as especificações da interface Cisco Aironet 360 [Cisco 2004]: protocolo de acesso ao meio IEEE 802.11; taxa de transmissão de 11 Mpbs; o raio de alcance de 100 m; antenas omnidirecionais e frequência de operação de 2,437 GHz (canal 6). Além disto, a transmissão das ondas de rádio obedecem ao modelo de propagação *TwoRayGround*, que é adequado para representar a atenuação do sinal para as distâncias adotadas.

Para uma análise de desempenho mais completa é necessário avaliar o comportamento dos protocolos quando o número de eventos aumenta, isto é, quando ocorrem mudanças na topologia. Estas mudanças são mais frequentes quando os nós estão em movimento, pois os *Mesh Clients* podem sair do alcance de um *Mesh Router*, perdendo o enlace e entrar na área de outro *Mesh Router* estabelecendo um novo enlace. Por esta razão, foram utilizados cenários com baixa e alta mobilidade dos *Mesh Clients*. Isto é, com baixa velocidade e alta velocidade de movimentação dos clientes.

O modelo de mobilidade descreve o padrão de movimento dos nós móveis determinando como a velocidade e a localização podem mudar com o tempo. No modelo de mobilidade adotado, os *Mesh Clients* permanecem em movimento contínuo, sem paradas (*think-time*), selecionando um novo destino aleatoriamente sempre que a localização for

alcançada. Este modelo permite que os nós possam movimentar-se em velocidade constante ou escolhendo uma velocidade variável. Desta forma, o modelo adotado favorece a geração de eventos e assim permite uma análise mais completa.

As simulações foram realizadas aumentando gradativamente a velocidade e a quantidade de *Mesh Clients*, possibilitando verificar a evolução do comportamento dos protocolos a até situações extremas. Neste trabalho, foram realizadas simulações com as velocidades constantes 0, 0.1, 1, 10 e 20m/s. Além disto, também foram realizadas simulações em velocidade variável. A velocidade variável é obtida escolhendo um valor aleatório variando de 0 a 20m/s, independentemente por cada nó, a cada novo destino selecionado.

Cada uma dessas velocidades foi simulada com a quantidade de *Mesh Clients*: 0, 10, 50 e 100. Todas as simulações tiveram duração de 3000 segundos. Com este tempo, os *Mesh Clients* podem percorrer todo o cenário gerando diversos eventos permitindo melhor observar o desempenho dos protocolos.

5.5. Parâmetros Quantitativos de Avaliação

Através dos parâmetros quantitativos é possível realizar a avaliação, em termos de eficiência de uso da rede, dos protocolos MLSD e OLSR analisados. Todavia, estes parâmetros não são fornecidos diretamente pelo simulador. Os parâmetros são obtidos através do processamento do arquivo de rastro (*trace*), gerado pelo simulador, que registra os eventos ocorridos durante a simulação.

Para recuperar estes parâmetros, conforme já citado, *scripts* devem ser desenvolvidos para processar o arquivo *trace*, levando em consideração o formato da saída de cada protocolo. Neste trabalho, o processamento do arquivo *trace* é realizado através de *scripts* desenvolvidos em PHP [Php 2010]. Os parâmetros quantitativos adotados para a avaliação dos protocolos simulados são:

- Total de mensagens: é a quantidade de mensagens de atualização geradas para divulgação das modificações da topologia durante toda a simulação;
- Carga das mensagens: é a quantidade de bytes transmitidos na rede para o envio das mensagens.

Dentre estes parâmetros, o mais importante é a carga das mensagens. A quantidade de bytes é obtida da camada MAC e considera todos os bytes necessários para transmissão da informação de topologia. Todavia, devido ao fato do OLSR eventualmente juntar mensagens

HELLO, com mensagens *TC*, em um mesmo pacote transmitido, é necessário remover os bytes relacionados às informações da vizinhança do OLSR para comparar com o MLSD apenas a carga de mensagens relacionadas à topologia, de forma justa.

O total de mensagens, por sua vez, permite ver a adaptação dos protocolos quando ocorre grande quantidade de eventos, já que ambos podem acumular e divulgar diversos enlaces em uma única mensagem, e utilizam rajadas quando a quantidade de atualizações é superior ao limite permitido para transmissão.

Conjuntamente, ambos os parâmetros dão uma visão melhor da eficiência dos protocolos avaliados.

5.6. Simulação dos Cenários

Para a plataforma de testes de emulação adotada é necessário que o processo de avaliação de desempenho siga uma metodologia para obter resultados com confiabilidade e credibilidade [Andel 2006]. Para obter resultados confiáveis, para cada configuração de cenário é necessário realizar um conjunto de simulações, utilizando sementes aleatórias diferentes. Através de cada simulação é possível obter os valores dos parâmetros investigados, e o resultado da avaliação é a média dos valores para um dado nível de confiança e nível de erro definidos. Em [Law 2007] é possível encontrar um guia de como isto pode ser realizado.

Cada cenário pode exigir uma quantidade diferente de simulações. A cada simulação, o conjunto dos resultados obtidos a precisa ser reavaliado. Assim, verifica se a média de cada um dos parâmetros desejados atende ao nível de confiança de 95% e margem de erro aceitável de 5%, considerados adequados para estas simulações.

Durante o período de inicialização da simulação, os nós começam a estabelecer os enlaces com os vizinhos, trocando uma grande quantidade de mensagens. Neste momento inicial, chamado *warm-up period*, o comportamento de qualquer protocolo diverge muito do restante do tempo da simulação e produz resultados que poluem a amostra. Por exemplo, podem ser realizadas muitas sincronizações ou ocorrer muitas colisões. Por isto, é necessário realizar ajustes para corrigir distorções da amostra.

A correção é realizada pela remoção do *warm-up period*. Inicialmente são executadas simulações para observar o a duração do *warm-up period*. Neste trabalho, foi verificado o tempo de 160 segundos é suficiente para que todos os protocolos envolvidos realizem as trocas de mensagens e estabeleçam seus enlaces iniciais, passando assim, a operar de maneira

correta. Desta forma, todos os eventos ocorridos até 160 segundos devem ser desconsiderados para análise. Além disto, a execução longa da simulação, por 3000s, auxilia a reduzir o efeito de qualquer distorção.

Os protocolos analisados, MLSD e OLSR, somaram 48 configurações de cenários diferentes. Cada um destes cenários foi simulado até que os parâmetros quantitativos usados na avaliação atingissem o nível de confiança de 95% e margem de erro de 5%, totalizando 2237 simulações.

A realização de todas as simulações especificadas, obedecendo aos parâmetros de confiabilidade adotados, demandou bastante tempo de processamento, devido ao grande número de simulações. Por este motivo, mesmo utilizando 48 máquinas disponibilizadas 24 horas por dia para realizar as simulações, foi necessário um período de quase dois meses para finalizar todas as simulações.

5.7. Considerações finais

Para avaliar o comportamento dos protocolos de rede podem ser usadas duas abordagens: a plataforma de testes em redes e a plataforma de testes de emulação. A plataforma de teste de emulação foi adotada para este trabalho. Ela demanda uma metodologia para realizar os experimentos para que os resultados possam ser obtidos com confiabilidade.

Para realizar as simulações, foi usado o *Network Simulator-2* (NS-2), que é um simulador de eventos discretos amplamente utilizado na pesquisa de redes. O NS-2 recebe um script com a configuração do cenário e realiza a simulação gerando um arquivo *Trace*. Este arquivo é processado por *scripts* para gerar os parâmetros de avaliação quantitativos especificados.

O cenário de RMSF infraestruturada que foi projetado busca explorar o máximo do comportamento dos protocolos simulados através de configurações com diversas quantidades de *Mesh Clients* em diversas velocidades.

Capítulo 6

Avaliação de Desempenho

Este capítulo apresenta a avaliação dos resultados obtidos a partir das simulações dos protocolos MLSD e OLSR, utilizando os seguintes parâmetros quantitativos: carga das mensagens e total de mensagens. Os resultados correspondem aos parâmetros obtidos a partir de diversas simulações utilizando diversas configurações de quantidade de *Mesh Clients* e velocidades.

Os dados obtidos representam o comportamento médio dos protocolos, em cada configuração, dentro do intervalo de confiança adotado. Através destes dados são gerados os gráficos que serão apresentados. Estes gráficos sintetizam as simulações de diversas configurações de cenários sob a perspectiva de um único parâmetro de avaliação.

Em um mesmo gráfico, são mostradas diferentes quantidades de *Mesh Clients* (0, 10, 50, 100) em que todos os dispositivos estão com velocidade variável, ou todos estão com uma mesma velocidade constante de: 0, 0.1, 1, 10, 20 m/s. Assim, um gráfico exibe o comportamento dos protocolos para um parâmetro avaliado, em redes com diferentes quantidades e velocidades de dispositivos.

É importante destacar que as informações da topologia provêm da troca de mensagens de topologia, mas também da camada de vizinhança. Isto é, no OLSR a escolha dos MPRs se dá através de mensagens de controle da vizinhança. Por sua vez, o MLSD também envia as atualizações de topologia quando o *Mesh Router* é informando de alguma modificação em seus enlaces.

Desta forma, é necessário que os processos de controle da vizinhança estejam funcionando para fornecer as informações à camada de topologia. Os processos de controle de informações da vizinhança utilizados por ambos os protocolos já foram comparados em [Santos 2008]. Neste trabalho, a separação da camada de topologia se faz através da análise

das mensagens de topologia de forma isolada, não considerando o total de mensagens e carga envolvidas nos processos de controle da vizinhança.

6.1. Total de Mensagens

O total de mensagens de topologia representa a quantidade de pacotes enviados por todos os nós da rede, em cada protocolo, para divulgar as informações de topologia. O total de mensagens permite perceber como os protocolos se comportam quando ocorrem frequentes mudanças nos enlaces.

Quanto menor o total de mensagens mais eficiente e escalável é o protocolo, isto é, ele acaba sendo mais adequado a redes maiores, com mais dispositivos. No MLSD são contabilizadas todas as LSUs. No protocolo OLSR, foram considerados somente os pacotes que continham ao menos uma TC. Além disto, para o OLSR, mesmo pacotes com mais de uma TC são contabilizados como uma única mensagem de topologia.

Observando o gráfico da Figura 32, que apresenta o comportamento do parâmetro total de mensagens quando todos os *Mesh Clients* estão parados, pode-se constatar que o protocolo OLSR sofre influência bem maior que o MLSD quando se aumenta a quantidade de *Mesh Clients* da rede. É possível verificar um ganho de desempenho do MLSD de praticamente 100% em relação ao OLSR para qualquer quantidade de *Mesh Clients*.

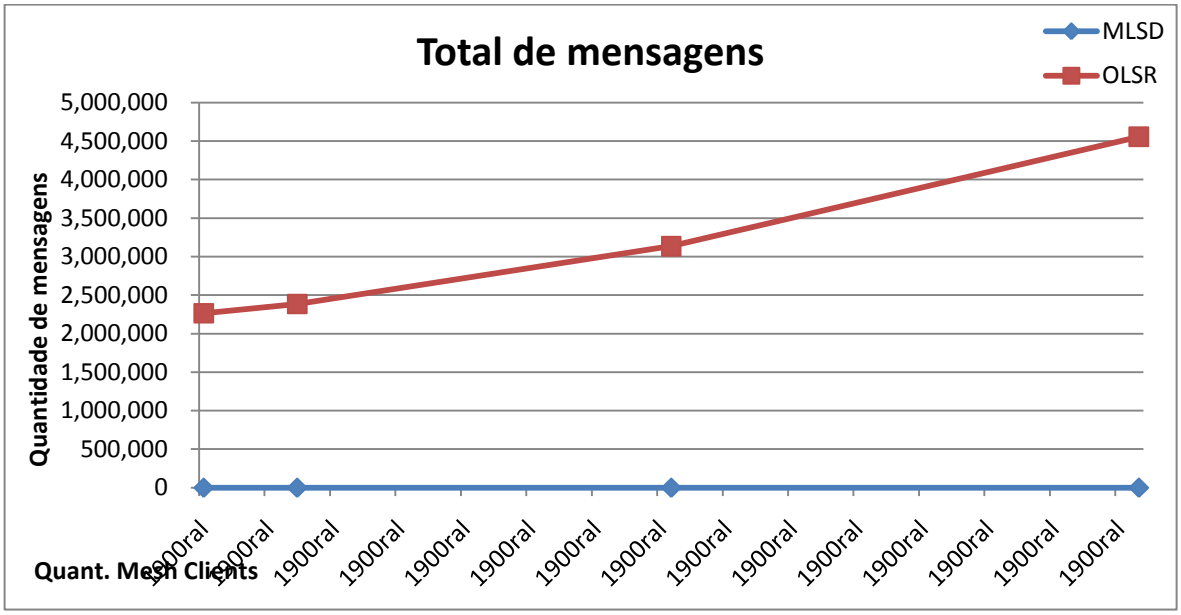


Figura 32 Total de mensagens com os *Mesh Clients* parados

O bom desempenho do MLSD ocorre porque praticamente não se tem modificações dos enlaces, portanto, nenhuma mensagem precisa ser enviada. Todavia, existe uma pequena

quantidade de mensagens enviadas quando o número de *Mesh Clients* é igual a 100. Isto ocorre quando a perda de um enlace é informada para o MLSD, em uma situação muito particular da camada de vizinhança: devido a colisões, um *Mesh Router* pode temporariamente perder o enlace com algum *Mesh Client*. Mesmo sendo rapidamente corrigida, a percepção da perda do enlace faz o *Mesh Router* divulgar a LSU. Contudo, no gráfico da Figura 32 esta quantidade de mensagens é imperceptível devido à escala, pois o total de mensagens dessas correções é extremamente pequeno em relação ao total de mensagens do OLSR.

O elevado total de mensagens enviadas pelo OLSR ocorre porque nele qualquer nó, e não somente os que compõem o *backbone*, podem ser eleitos MPRs, e por consequência enviam mensagens TC. Além disto, cada nó escolhe independentemente seu conjunto de MPRs. Assim, a heurística do OLSR acaba por eleger mais nós que o necessário para cobrir todo o *backbone* e, com isto, envia e repassa mais mensagens.

Vale ressaltar que a quantidade de mensagens enviadas pelo OLSR também é uma função do tempo, visto que o envio é periódico. Os valores do gráfico são referentes ao tempo de análise de 2840 segundos, isto é, tempo total da simulação é 3000s removido os 160s iniciais do *warm-up period*.

É importante destacar no gráfico da Figura 32, que mesmo sem nenhum *Mesh Client*, o que implica um total de mensagens gerado apenas pelo *backbone* de *Mesh Routers*, o OLSR possui um total de mensagens muito elevado. Isto ocorre porque nesta topologia praticamente todos os nós do *backbone* são declarados MPRs e, além de enviar as próprias mensagens, devem realizar os repasses das mensagens dos vizinhos em um intervalo de no máximo 0,5 segundos.

Por fim, outro motivo contribui para o elevado total de mensagens do OLSR: embora cada pacote do OLSR possa divulgar até 256 enlaces, este valor na verdade corresponde a apenas 4 TCs com 64 enlaces. Portanto, caso o MPR receba pacotes contendo TCs dos MPRs vizinhos, e essas TCs sejam originadas de mais de 4 MPRs diferentes (ex.: nos repasses), o MPR precisará enviar mais pacotes OLSR para divulgar todas as mensagens. Este envio é realizado em rajada, enviando tantos pacotes quantos forem necessários para divulgar todas as TCs não repassadas imediatamente.

Observando o gráfico da Figura 33, que apresenta o comportamento do total de mensagens quando os nós estão em movimento. É possível ver que o total de mensagens

também é influenciado pela mobilidade dos nós. Isto é evidenciado pela posição das curvas do mesmo protocolo em velocidades diferentes. Quando os nós estão com velocidade de 0.1m/s, o ganho do MLSD em relação ao OLSR é de no mínimo 97,3%. A medida que a velocidade aumenta, este ganho diminui porque o MLSD começa a enviar mais mensagens. Contudo, sempre apresenta um desempenho melhor que o OLSR, mesmo quando 100 *Mesh Clients* movem-se a 20m/s, alcançando um ganho de 69,4% em relação ao OLSR.

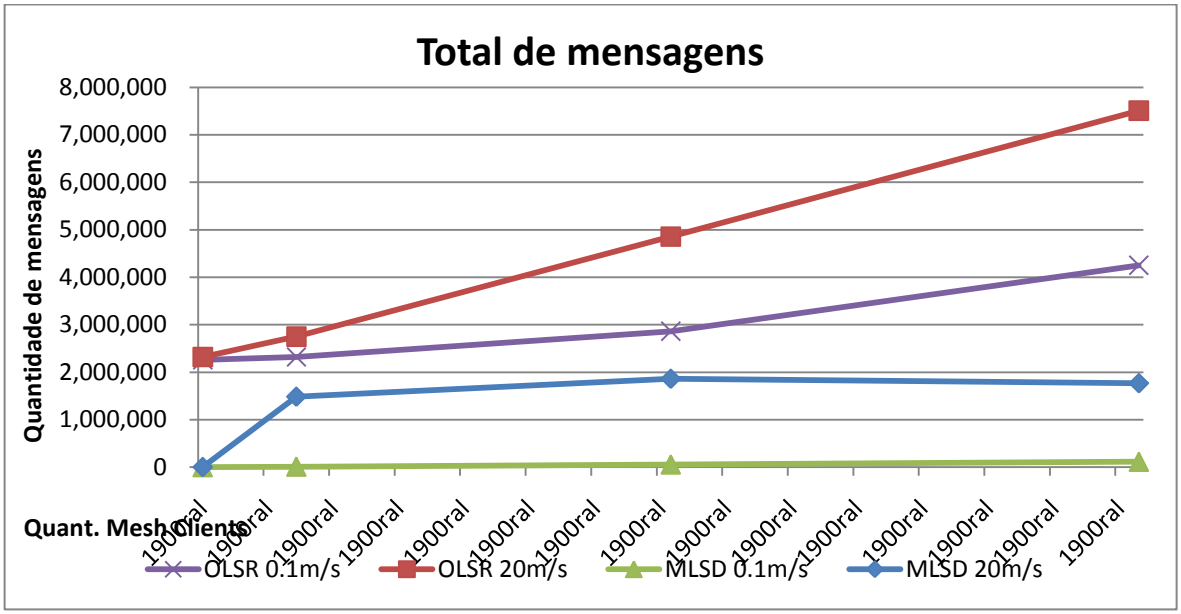


Figura 33 Total de mensagens para os nós em movimento

A razão do baixo desempenho do OLSR em relação ao total de mensagens quando se aumenta a velocidade é que, somado as razões já citadas, a movimentação dos *Mesh Clients* faz os nós estabelecerem outros enlaces elegendo outros MPRs e, consequentemente, enviando mais mensagens. Contudo, a perda do enlace no OLSR não é imediata e enquanto não for alertado pela vizinhança, que pode ocorrer em 6 segundos conforme a especificação, o nó eleito MPR permanece enviando e repassando TCs por algum tempo.

O MLSD apresenta um comportamento diferenciado quando a velocidade é de 20m/s e aumenta a quantidade de *Mesh Clients* em movimento. Conforme o gráfico, a curva reduz o crescimento. A razão disto é que, após enviar uma LSU, o tempo para outra mensagem do mesmo *Mesh Router* ser enviada é multiplicado pela quantidade de mensagens necessárias para entregar as atualizações do *buffer*. Quando aumenta a quantidade de clientes se movimentando rapidamente o número de atualizações de topologia se eleva. Se for necessária mais de uma LSU para enviar as atualizações, então o multiplicador do número de mensagens

retarda o envio acumulando mais atualizações na LSU adicional, e assim envia menos mensagens contendo mais atualizações.

Esta característica permite que o MLSD se adapte aos cenários com grande quantidade de eventos da rede, em especial cenários com grande mobilidade. Quando existem poucas atualizações, elas são rapidamente divulgadas, podendo elevar o total de mensagens. Quando existem muitas atualizações, um retardo maior entre as mensagens é aplicado através do multiplicador da quantidade de mensagens, permitindo enviar mais atualizações de topologia por LSU.

Podemos observar no gráfico da Figura 34 que no cenário de velocidade variável a curva do MLSD mostra um comportamento crescente, significando que o número de eventos não é suficiente para ativar o efeito do multiplicador, não ocorrendo saturação da rede.

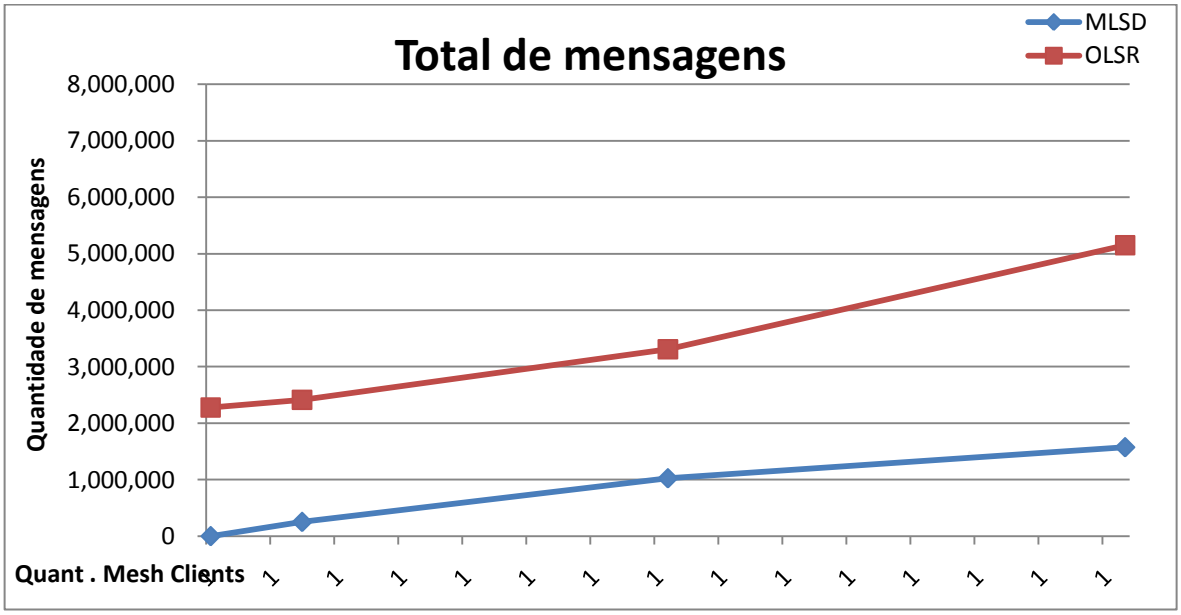


Figura 34 Total de mensagens em velocidade variável de 0 a 20m/s

Os gráficos mostram que o MLSD é mais eficiente que o OLSR no total de mensagens em todos os cenários avaliados, apresentando ganhos que vão de 46,0% até 100% de eficiência em relação ao OLSR como pode ser visto na Tabela 1.

Tabela 1 Ganho percentual do total de mensagens do MLSD sobre o OLSR

Mesh Clients	Velocidades em m/s					
	Parado	0.1	1	10	20	0-20
0	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
10	100,0%	99,5%	95,1%	62,9%	46,0%	89,5%
50	100,0%	98,0%	81,8%	57,8%	61,7%	69,0%
100	99,9%	97,3%	78,0%	75,0%	76,4%	69,4%

6.2. Carga de mensagens

A carga de mensagens dos protocolos indica a quantidade *bytes* utilizada pelas mensagens de controle que cada protocolo produz na rede durante a simulação. Este parâmetro foi analisado considerando a quantidade de bytes necessários para transmitir o quadro na camada MAC. Como já explicado, apenas as mensagens envolvidas no controle da topologia foram contabilizadas. Quanto menor a carga de mensagens para manter as informações de topologia consistentes, mais eficiente é o protocolo.

Para o OLSR, que pode enviar mensagens *TC* e *HELLO* no mesmo pacote, a carga relacionada às mensagens de cotrole da vizinhança (*HELLO*) foram removidos para permitir a comparação somente das informações da topologia.

Conforme observado na Figura 35, em que todos os *Mesh Clients* estão parados, é possível observar que a carga de mensagens do protocolo OLSR é influenciada pelo aumento da quantidade de *Mesh Clients*. A razão é a mesma do aumento do total de mensagens: a heurística do OLSR elege mais nós que o necessário para cobrir todo o *backbone* e, com isto, envia e repassa mais mensagens e isto eleva a carga de mensagens.

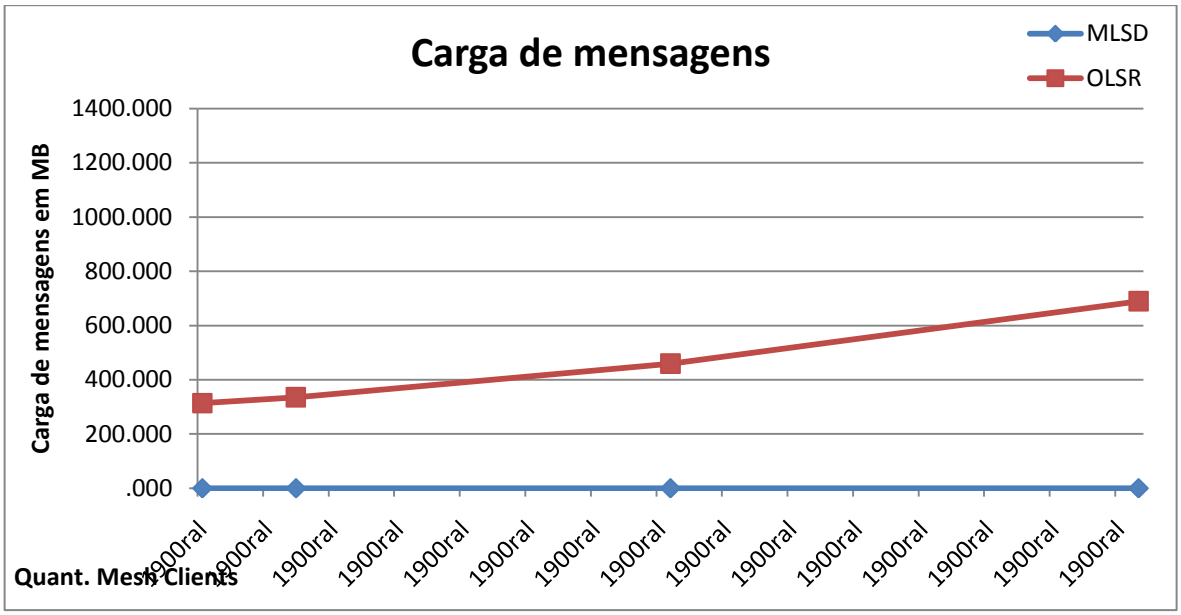


Figura 35 Carga de mensagens com os *Mesh Clients* parados

O comportamento da carga de mensagens no OLSR é semelhante ao comportamento do total de mensagens. Vale destacar na Figura 35 que mesmo quando existem apenas os *Mesh Routers*, a carga de mensagens enviadas pelo OLSR é bastante elevada, acima dos 300 MB. Isto ocorre devido à elevada quantidade de mensagens geradas e repassadas por todos os MPRs que, conforme mencionado, são a maioria dos nós do *backbone*.

Além do elevado total de mensagens, outras características contribuem para a elevada carga do OLSR. Primeiro o pacote do OLSR é encapsulado no datagrama UDP e IP. Segundo, o OLSR sempre divulga todos os enlaces do conjunto *MPR Selectors* em cada TC.

No MLSD, diferentemente do OLSR, a LSU é enviada diretamente no quadro da camada MAC. Apenas as atualizações incrementais e não todos os enlaces são divulgados nas mensagens. Além disto, as atualizações de topologia podem pertencer a um único ou mais *Mesh Routers*, limitado apenas à quantidade máxima de atualizações por pacotes. Com isto, o MLSD torna-se mais eficiente também para divulgar as informações.

O desempenho do MLSD é superior ao OLSR em cenários com *Mesh Clients* parados, não importando a quantidade de clientes, chegando a 100% de ganho nos cenários de 0, 10 e 50 *Mesh Clients*. Todavia, o MLSD apresenta uma pequena carga de mensagens quando a quantidade de *Mesh Clients* é igual a 100. Da mesma forma que no total de mensagens, esta carga de mensagens corresponde às mensagens de correção quando a camada de vizinhança, devido a colisões, perde temporariamente o enlace com algum *Mesh Client* fazendo o MLSD enviar LSU. Contudo, conforme o gráfico da Figura 35, a carga de mensagens dessas correções é extremamente pequena quando comparada a carga do OLSR.

Observando o gráfico da Figura 36 é possível verificar o comportamento da carga de mensagens dos protocolos MLSD e OLSR quando os *Mesh Clients* estão em movimento. Fica evidenciada, pelo posicionamento das curvas do mesmo protocolo em velocidades diferentes, que a mobilidade tem efeito na carga de mensagens de ambos os protocolos.

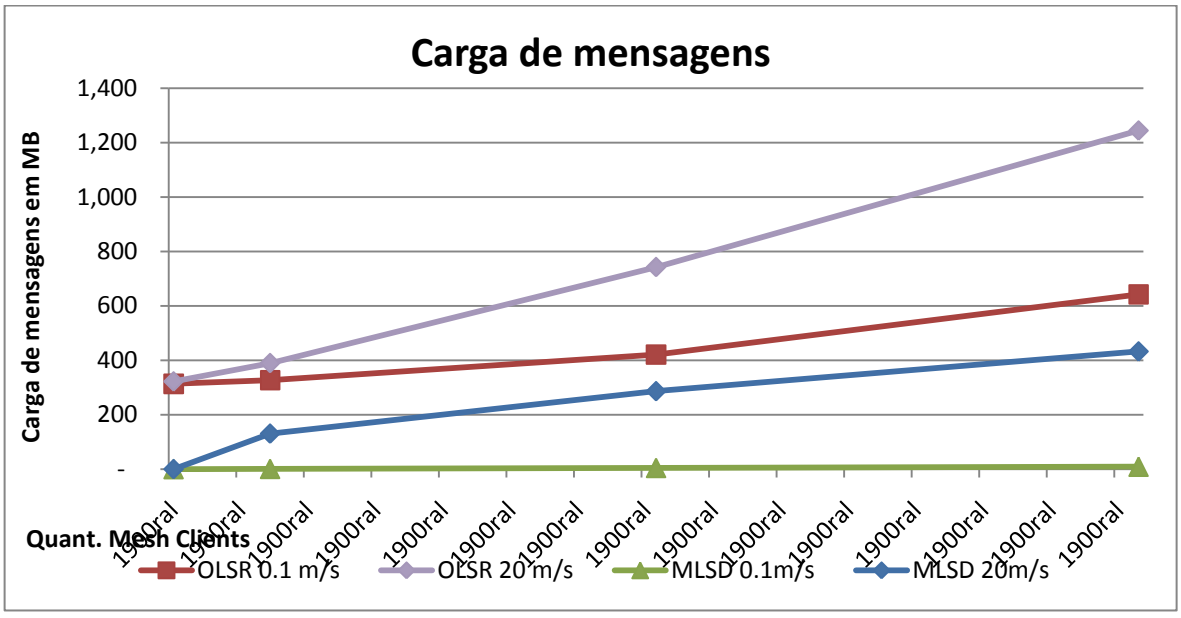


Figura 36 Carga de mensagens com os *Mesh Clients* em movimento

A razão do aumento da carga de mensagens provocada pelo aumento da velocidade é, no MLSD, derivada do aumento das mudanças nos enlaces fazendo atualizações de topologia ser geradas. Já no OLSR, o aumento se dá porque a movimentação faz os nós estabelecerem outros enlaces, elegendo outros MPRs e, consequentemente, enviando mais mensagens.

O comportamento da curva do MLSD em relação à carga de mensagens é diferente do comportamento da curva do MLSD com relação ao total de mensagens na velocidade de 20m/s, em que ocorrem muitos eventos. Conforme já explicado, embora o multiplicador da quantidade de mensagens aumente o tempo entre o envio das mensagens, também permite acumular mais atualizações de topologia resultando em LSUs contendo mais atualizações.

Uma maior quantidade de atualizações por mensagem resulta em elevação da carga de mensagens. Contudo, a eficiência da estratégia de entrega de mensagens do MLSD, utilizando os *slots* diferenciados para cada *Mesh Router*, reduz a perda de mensagens provocadas por colisões, evitando que as atualizações se acumulem entre retransmissões e elevem demasiadamente a carga de mensagens.

Vale ressaltar que mesmo a 20 m/s o MLSD ainda envia uma carga de mensagens menor que o OLSR a 0.1 m/s mostrando um excelente desempenho. Outra informação obtida no gráfico é que os valores da carga do OLSR a 0.1m/s são ligeiramente inferiores ao OLSR parado, apresentado no gráfico da Figura 35. Esta diferença, no entanto, está dentro da margem de erro estabelecida.

É importante perceber que o valor da carga de mensagens do MLSD se mantém sempre abaixo do valor da carga de mensagens do OLSR. A Figura 37 apresenta o cenário de velocidade variável em que é possível perceber que quando a quantidade de *Mesh Clients* em movimento é igual a 100, o ganho do MLSD também aumenta. Neste cenário, conforme a Tabela 2, o ganho do MLSD é de 82% em relação ao OLSR.

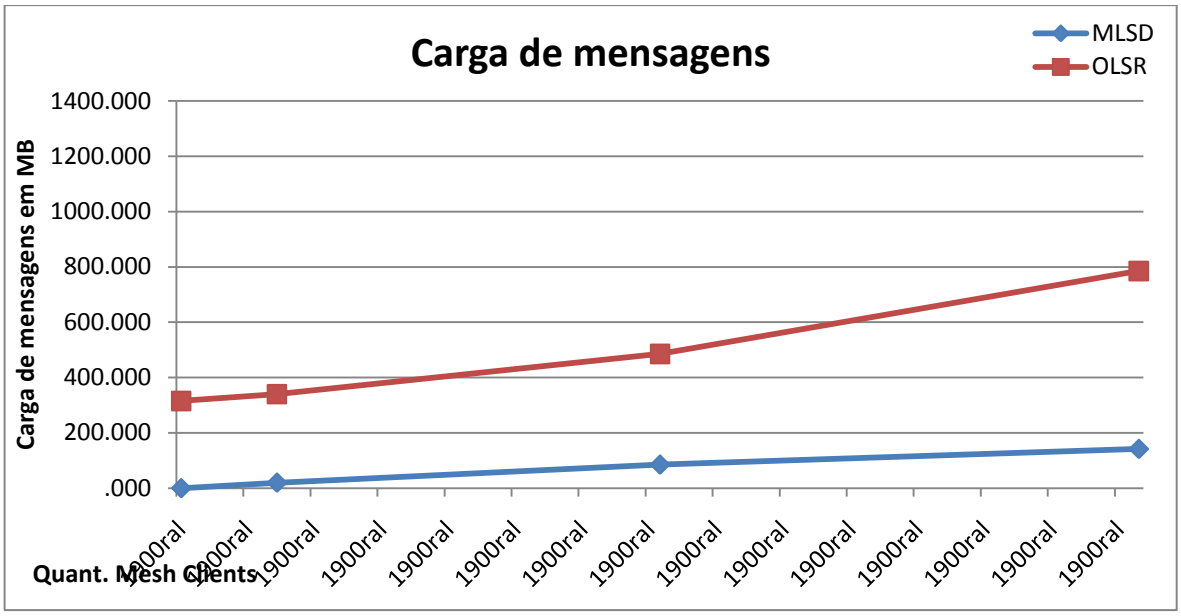


Figura 37 Carga de mensagens em velocidade variável de 0 a 20m/s

Tanto para o MLSD quanto para o OLSR, quanto maior a velocidade dos nós maior a carga de mensagens. Conforme pode ser visto na Tabela 2, em cenários de baixa mobilidade, isto é, de 0 até 1m/s, o desempenho do MLSD tem os maiores ganhos em relação ao OLSR, com valores superiores a 88,0%. Isto ocorre porque nestes cenários acontecem poucas modificações dos enlaces. No entanto, à medida que se aumenta a mobilidade, esta diferença é reduzida, contudo os resultados continuam favoráveis ao MLSD, que alcança um ganho de 65,3% quando os 100 *Mesh Clients* movimentam-se a 20m/s na área de cobertura.

Tabela 2 Ganho percentual da carga de mensagens do MLSD sobre o OLSR

Mesh Clients	**Velocidades em m/s**					
	Parado	0.1	1	10	20	0-20
0	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
10	100,0%	99,7%	97,3%	78,2%	66,5%	94,1%
50	100,0%	98,9%	90,1%	68,6%	61,5%	82,5%
100	100,0%	98,6%	88,0%	75,8%	65,3%	82,0%

Pode-se observar que o MLSD é mais eficiente que o OLSR na carga de mensagens em qualquer um dos cenários avaliados, apresentando ganhos que vão de 61,5% até 100% de eficiência em relação ao OLSR como pode ser visto na Tabela 2.

6.3. Considerações Finais

A carga de mensagens é um parâmetro crítico na avaliação de desempenho de qualquer protocolo. Reduzir o consumo de recursos da rede, enviando menos mensagens, é sempre um dos objetivos mais importantes devido às limitações de banda do meio sem fio. Desta forma, o protocolo torna-se mais escalável permitindo sua aplicação em redes com mais dispositivos.

Foi verificado que a carga e o total de mensagens para ambos os protocolos sofre influência da mobilidade e da quantidade dos dispositivos. Entretanto, o MLSD apresentou os melhores resultados das simulações, com desempenho superior ao OLSR em todos os cenários.

Vale destacar que os melhores resultados foram com cenários de clientes parados ou em baixa velocidade. Nestes cenários o MLSD alcança ganhos, tanto na carga quanto no total de mensagens, de até 100% com relação ao OLSR.

Capítulo 7

Conclusão e Trabalhos Futuros

Neste trabalho foi apresentando o protocolo MLSD (*Mesh Network Link State Dissemination Protocol*), um protocolo para divulgação dos estados dos enlaces para RMSF infraestruturada, que através do uso efetivo do *backbone* criado pelos nós fixos da RMSF, busca reduzir a carga e o total de mensagens de topologia.

As funcionalidades de roteamento requeridas por protocolos para RMSF já estão disponíveis em protocolos de redes *Ad Hoc*, por exemplo, o protocolo OLSR, que embora tenha sido desenvolvido originalmente para redes *Ad Hoc*, também é amplamente usado em RMSF e suas estratégias são base para outros protocolos projetados para RMSF. Contudo, as características específicas das RMSF permitem construir protocolos especializados e mais eficientes, por exemplo, a arquitetura de protocolos de roteamento para redes em malha infraestruturada (IWMRA), possibilita construir o MLSD, um protocolo especializado para a camada de topologia, concentrando-se em desenvolver estratégias para tratar eficientemente os problemas desta camada.

Conforme estudos encontrados em [Ros 2010], [Ge 2005] e [Nguyen 2007], a carga de mensagens geradas pelos protocolos para manter as informações da topologia tem um grande impacto no desempenho do protocolo de roteamento. Isto ocorre porque as mensagens devem ser divulgadas por toda a rede, levando a problemas de escalabilidade quando a quantidade de nós aumenta. Ao explorar as características das RMSF infraestruturadas, o MLSD consegue reduzir expressivamente esta carga quando comparado ao OLSR.

A eficiência do protocolo desenvolvido pôde ser constatada através de diversas simulações em diversas configurações de cenários, variando a quantidade de dispositivos e a velocidade. Os mesmos cenários foram simulados tanto para o OLSR quanto para o MLSD. Através destes cenários foi possível explorar o máximo do comportamento dos protocolos, quando a topologia se mantém estável e quando ocorrem frequentes mudanças dos enlaces.

Os parâmetros quantitativos utilizados para comparar a eficiência dos protocolos foram o comportamento da carga das mensagens de topologia e o total de mensagens.

A análise dos resultados revelou que o MLSD alcançou seus objetivos reduzindo o total de mensagens e a carga e das mensagens de topologia, obtendo excelentes resultados, comparado ao OLSR, em todos os cenários avaliados. Vale ressaltar que quanto mais lentos estão os *Mesh Clients* melhor foi o desempenho do protocolo MLSD em relação ao protocolo OLSR. Quando todos os *Mesh Clients* estão parados o ganho do MLSD em relação ao OLSR é 100% tanto para a carga quanto para o total de mensagens.

É importante destacar que, conforme [Zhang 2006], a mobilidade dos dispositivos da RMSF geralmente é baixa. Além disto, mesmo quando cresce a quantidade de *Mesh Clients*, no MLSD, a carga só se eleva se os *Mesh Clients* estiverem em movimento.

Quando os *Mesh Clients* se movimentam o ganho do MLSD em relação ao OLSR tanto para o parâmetro carga quanto o total de mensagens diminuem, embora o MLSD continue apresentando um excelente desempenho.

Um comportamento interessante do MLSD emerge quando muitos *Mesh Clients* movimentam-se rapidamente. Nesta configuração, o total de mensagens reduz o crescimento mostrando que as estratégias de controle de envio são eficientes. Contudo, as mensagens enviadas são maiores.

É importante destacar que em cenários de velocidade variável, em que alguns *Mesh Clients* movimentam-se rapidamente e outros mais lentamente, a redução da carga de mensagens é de pelo menos 82% em relação ao OLSR.

Pode-se afirmar que a grande contribuição deste trabalho foi o desenvolvimento do protocolo MLSD. Este protocolo se mostrou mais eficiente para divulgar as informações de topologia do que protocolo OLSR, conseguindo obter reduções expressivas na carga de mensagens e no total de mensagens, mostrando-se mais escalável.

7.1. Trabalhos Futuros

Outros parâmetros de análise podem contribuir para tornar o estudo do protocolo MLSD desenvolvido nesta dissertação mais completo. Por exemplo, um parâmetro importante a ser investigado por protocolos de divulgação de topologia é a convergência das informações da rede. Embora tenham sido realizados diversos testes para garantir a convergência das informações nos diversos cenários, ainda seria interessante uma prova de corretude com

verificação formal. Além disto, outro parâmetro importante é o tempo de convergência, que revela o tempo em que a informação leva para ser conhecida por todos os nós do *backbone*. Para realizar este estudo um novo conjunto de simulações precisa ser executado.

Contudo, através de análises iniciais, acreditamos que o MLSD também apresente um bom desempenho em relação ao tempo de convergência, pois além de implementar um mecanismo de entrega garantida, o tempo de reenvio das informações é baixo, em no máximo 0,9375 segundos que é o maior tempo de retransmissão. No OLSR, que não fornece nenhuma garantia de entrega, a informação do mesmo enlace só é retransmitida pela origem em 5 segundos. Além disto, o OLSR apresenta sérios problemas de convergência, conforme estudos apresentados em [Gowrishankar 2007], chegando a nunca convergir em cenários de grande mobilidade.

Incluir cenários de redes com mais *Mesh Clients* e outros protocolos também pode contribuir para o estudo do protocolo. Contudo, o simulador utilizado mostrou problemas de estabilidade e consumo de recursos quando a simulação ocorre em cenários com grandes quantidades de nós. Por isto, outros simuladores devem ser utilizados, implicando em uma implementação específica do protocolo.

Como percebido pelos resultados da avaliação, mesmo apresentando um bom desempenho em cenários com grande velocidade, ainda há espaço para refinamentos e complementação da estratégia. Por exemplo, incluir no protocolo a capacidade de detectar quando um *Mesh Client* está em movimento. Os nós em movimento provocam a geração das mensagens de atualização nos *Mesh Routers*, mesmo que não estejam usando a rede para se comunicar. Ao detectar um nó em movimento, o protocolo pode reduzir a carga de mensagens gerada por ele se ele não estiver se comunicando, evitando emitir mensagens de atualização.

Por fim, é necessário completar especificação da camada de roteamento da IWMRA, e com isto finalizar o protocolo de roteamento.

Referências

[802.11i-Tg 2004] 802.11I-TG. *ISO/IEC International Standard - Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Medium Access Control (MAC) Security Enhancements*. ISO/IEC 8802-11, Second edition: 2005/Amendment 6 2006: IEEE STD 802.11i-2004 (Amendment to IEEE Std 802.11-1999), p.c1-178. 2004.

[802.11s-Tg 2008] *Summary Reports of IEEE 802.11 WLAN Working Group Sessions - Task Group "S"*. Disponível em: http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm. Acesso em 03/02/2010.

[Akyildiz 2005] AKYILDIZ, I. F. e XUDONG, W. *A survey on wireless mesh networks*. Communications Magazine, IEEE, v.43, n.9, p.S23-S30. 2005.

[Albuquerque 2006] *GT-Mesh RT1 - Termo de referencia e estado da arte*. Rede Nacional de Ensino e Pesquisa. 2006

[Altman 2003] *NS Simulator for beginners*. Disponível em: http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/n3.pdf. Acesso em 2007.

[Andel 2006] ANDEL, T. R. e YASINSAC, A. *On the credibility of manet simulations*. Computer, v.39, n.7, p.48-54. 2006.

[Andrews 2007] ANDREWS, J., GHOSH, A. e MUHAMED, R. *Fundamentals of WiMAX*. Understanding Broadband wireless Networking PRENTICE HALL. 2007.

[Aoki 2006] AOKI, H. *et al. IEEE 802.11 s Wireless LAN Mesh Network Technology*. NTT DoCoMo Tech J, v.8, n.2, p.13-21. 2006.

[Bahr 2006] BAHR, M. *Proposed Routing for IEEE 802.11s WLAN Mesh Networks*. 2nd Annual International Workshop on Wireless Internet. Boston, USA.p 10, 2006.

[Bahr 2007] _____. *Update on the Hybrid Wireless Mesh Protocol of IEEE 802.11s*. Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on.p 1-6, 2007.

[Basagni 2004] BASAGNI, S. *Mobile Ad Hoc Networking*: Wiley-IEEE Press. 2004.

[Bicket 2005] BICKET, J. *et al. Architecture and evaluation of an unplanned 802.11b mesh network*. Proceedings of the 11th annual international conference on Mobile computing and networking. Cologne, Germany: ACM.p, 2005.

[Bruno 2005] BRUNO, R., CONTI, M. e GREGORI, E. *Mesh networks: commodity multihop ad hoc networks*. Communications Magazine, IEEE, v.43, n.3, p.123-131. 2005.

[Campista 2008a] CAMPISTA, M. *et al. Routing Metrics and Protocols for Wireless Mesh Networks*. Network, IEEE, v.22, n.1, p.6-12. 2008a.

[Campista 2008b] CAMPISTA, M. E., COSTA, L. e DUARTE, O. *WPR: Um Protocolo de Roteamento Pro-ativo Adaptado as Redes em Malha Sem Fio*. XXVI Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro, Brasil.p 14. Maio, 2008b.

[Campista 2008c] CAMPISTA, M. E., COSTA, L. H. M. K. e DUARTE, O. C. M. B. *WPR: A Proactive Routing Protocol Tailored to Wireless Mesh Networks*. IEEE Globecom Ad Hoc, Sensor and Mesh Networking Symposium. New Orleans, LA, USA.p 1-5. December, 2008c.

[Campos 2005] CAMPOS, G. M. M. *Avaliação de Desempenho de Protocolos de Roteamento para Redes Móveis Ad Hoc Sob Condições de Tráfego de Aplicações de VideoFone*. Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, p 110. 2005.

[Carrano 2009] CARRANO, R. C. *et al. Multihop MAC: IEEE 802.11s Wireless Mesh Networks*. Encyclopedia on Ad Hoc and Ubiquitous Computing Dharma Agrawal.p, 2009.

[Chakeres 2006] CHAKERES, I. e PERKINS, C. *Dynamic MANET On-demand (DYMO) Routing*. draft-ietf-nanet-dymo-05, June. 2006.

[Chen 2006] CHEN, J. *et al. Performance Comparison of AODV and OFLSR in Wireless Mesh Networks*. IFIP Mediterranean Ad Hoc Networking Workshop.p 271-278, 2006.

[Cisco 2004] *Cisco Aironet 350 Series Client Adapters*. Disponível em: http://www.cisco.com/en/US/prod/collateral/wireless/ps6442/ps4555/ps448/product_data_sheet09186a0080088828.pdf. Acesso em 15/01/2010.

[Clausen 2003] CLAUSEN, T. e JACQUET, P. *RFC3626 - Optimized Link State Routing Protocol (OLSR)*: IETF - Network Working Group.p 75, 2003.

[Cordeiro 2002] CORDEIRO, C. e AGRAWAL, D. *Mobile Ad Hoc Networking*. C. M. Cordeiro and D. P. Agrawal, Mobile Ad Hoc Networking, Tutorial/Short Course in 20 th Brazilian Symposium on Computer Networks, pp. 125-186, May 2002, http://www.ececs.uc.edu/-cordeicm/. 2002.

[Couto 2005] COUTO, D. S. J. D. *et al. A high-throughput path metric for multi-hop wireless routing*. Proceedings of the 9th annual international conference on Mobile computing and networking, v.11, n.4, p.419-434. 2005.

[Dijkstra 1959] DIJKSTRA, E. *A note on two problems in connexion with graphs. S. 269–271*. Numerische Mathematik, v.1. 1959.

[Draves 2004] DRAVES, R., PADHYE, J. e ZILL, B. *Routing in multi-radio, multi-hop wireless mesh networks*. Proceedings of the 10th annual international conference on Mobile computing and networking. Philadelphia, PA, USA.p 114-128, 2004.

[Faccin 2006] FACCIN, S. M. *et al. Mesh WLAN networks: concept and system design*. IEEE Wireless Communications, v.13, n.2, p.10-17. 2006.

[Fall 1997] FALL, K. e VARADHAN, K. *ns Notes and Documentation*. The VINT Project, UC Berkely, LBL, USC/ISI, and Xerox PARC. 1997.

[Ge 2005] GE, Y., LAMONT, L. e VILLASENOR, L. *Hierarchical OLSR-a scalable proactive routing protocol for heterogeneous ad hoc networks*. IEEE International Conference on Wireless And Mobile Computing, Networking And Communications.p 17-23, 2005.

[Gowrishankar 2007] GOWRISHANKAR, S. *et al. Scenario based Performance Analysis of AODV and OLSR in Mobile Ad hoc Networks*. Proceedings of the 24th South East Asia Regional Computer Conference.p 6, 2007.

[Haas 1997] HAAS, Z. e PEARLMAN, M. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. TERNET DRAFT-Mobile Ad hoc Networking (MANET) Working Group of the bternet Engineering Task Force (ETF), November. 1997.

[Hedrick 1988] HEDRICK, C. *Routing Information Protocol*: RFC 1058, Rutgers University, June 1988 1988.

[Held 2005] HELD, G. *Wireless Mesh Networks*: Auerbach Publications.p 149. 2005.

[Hossain 2008] HOSSAIN, E. *Kin K. leung,"Wireless Mesh Networks, Architectures and Protocols"*: Springer 2008.

[Huang 2006] HUANG, Y., BHATTI, S. e PARKER, D. *Tuning olsr*. The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications: Citeseer.p 1-5, 2006.

[Ieee 2007] International Standard - Information Technology Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks-Specific Requirements - Part 11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, p.C1-1184. 2007.

[Jacquet 2001] JACQUET, P. *et al. Optimized link state routing protocol for ad hoc networks*. IEEE International Conference - Technology for the 21st Century.p 62-68, 2001.

[Johnson 2004] JOHNSON, D., MALTZ, D. e HU, Y. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*. Ietf Manet Working Group (Draft 10). 2004.

[Johnson 2007] JOHNSON, D. L. *Performance Analysis of Mesh Networks in Indoor and Outdoor Wireless Testbeds* Departement: Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, p 113. 2007.

[Jun 2003] JUN, J. e SICHITIU, M. *The nominal capacity of wireless mesh networks*. IEEE Wireless Communications, v.10, n.5, p.8-14. 2003.

[Law 2007] LAW, A. M. *Statistical analysis of simulation output data: the practical state of the art*. Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come. Washington D.C.: IEEE Press.p, 2007.

[Lee 1999] LEE, S. J., GERLA, M. e TOH, C. K. *A simulation study of table-driven and on-demand routing protocolsfor mobile ad hoc networks*. Network, IEEE, v.13, n.4, p.48-54. 1999.

[Macker 1998] MACKER, J. P. e CORSON, M. S. *Mobile ad hoc networking and the IETF*. SIGMOBILE Mob. Comput. Commun. Rev., v.2, n.1, p.9-14. 1998.

[Mangold 2002] MANGOLD, S. *et al. IEEE 802.11e Wireless LAN for Quality of Service*: Citeseer.p 32-39, 2002.

[Miguel Elias 2007] MIGUEL ELIAS, M. C. *et al. The ad hoc return channel: a low-cost solution for Brazilian interactive digital TV*. Communications Magazine, IEEE, v.45, n.1, p.136-143. 2007.

[Moy 1998] MOY, J. *IETF RFC 2328: OSPF Version 2*: April 1998.

[Myung Jong 2006] MYUNG JONG, L. *et al. A new taxonomy of routing algorithms for wireless mobile ad hoc networks: the component approach*. Communications Magazine, IEEE, v.44, n.11, p.116-123. 2006.

[Nezhad 2008] NEZHAD, A., MIRI, A. e MAKRAKIS, D. *Efficient topology discovery for multihop wireless sensor networks*: IEEE Computer Society.p 358-365, 2008.

[Nguyen 2007] NGUYEN, D. e MINET, P. *Scalability of the OLSR protocol with the Fish Eye extension*.p 88, 2007.

[Pei 2000] PEI, G., GERLA, M. e CHEN, T.-W. *Fisheye State Routing: A Routing Scheme for Ad-hoc Wireless Networks*. IEEE International Conference on Communications. New Orleans.p 70-74, 2000.

[Perkins 1994] PERKINS, C. E. e BHAGWAT, P. *Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers*. Proceedings of the conference on Communications architectures, protocols and applications. London, United Kingdom: ACM.p, 1994.

[Perkins 1999] PERKINS, C. E. e ROYER, E. M. *Ad-hoc on-demand distance vector routing*. Second IEEE Workshop on Mobile Computing Systems and Applications. New Orleans.p 90-100, 1999.

[Php 2010] Disponível em: http://www.php.net. Acesso em 04/03/2010.

[Pore 2006] PORE, G. L. *A Performance Analysis of Routing Protocols for ad-hoc Networks*. Dept of Electrical And Computer Engineering, Naval Postgraduate School, Monterey Ca, p 113. 2006.

[Porto 2009] PORTO, D. C. F., CAVALCANTI, G. e ELIAS, G. *A Layered Routing Architecture for Infrastructure Wireless Mesh Networks*. ICNS '09 Fifth International Conference on Networking and Services.p 366-369, 2009.

[Ros 2010] Disponível em: http://masimum.dif.um.es/?Software:UM-OLSR. Acesso em 03/03/2010.

[Ros 2004] ROS, F. J. e RUIZ, P. M. *Implementing a New Manet Unicast Routing Protocol in NS2*: Dept. of Information and Communications Engineering University of Murcia.p 35. 2004.

[Royer 1999] ROYER, E. M. e CHAI-KEONG, T. *A review of current routing protocols for ad hoc mobile wireless networks*. IEEE Wireless Communications, v.6, n.2, p.46-55. 1999.

[Santos 2008] SANTOS, M. N. *SNDP: Um protocolo escalável para descoberta de vizinhança para redes em malha sem fio infra-estruturadas*. Departamento de Informática, Universidade Federal da Paraíba-UFPB, João Pessoa, p 101. 2008.

[Sichitiu 2005] SICHITIU, M. *Wireless Mesh Networks: Opportunities and Challenges*. Proceedings of the Wireless World Congress, Palo Alto, CA, May. 2005.

[Tanenbaum 2003] TANENBAUM, A. S. *Redes de Computadores*: Elsevier.p 945. 2003.

[Tsarmpopoulos 2005] TSARMPOPOULOS, N., KALAVROS, I. e LALIS, S. *A low-cost and simple-to-deploy peer-to-peer wireless network based on open source Linux routers*. Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on.p 92-97, 2005.

[Zhang 2006] ZHANG, Y., LUO, J. e HU, H. *Wireless Mesh Networking: Architectures, Protocols And Standards*: Auerbach Publications. 2006.

[Zigbee 2008] *ZigBee Alliance*. Disponível em: http://www.zigbee.org. Acesso em 21/07/2008.

[Zou 2002] ZOU, X., RAMAMURTHY, B. e MAGLIVERAS, S. *Routing Techniques in Wireless Ad Hoc Networks Classification and Comparison*. Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics, SCI: Citeseer.p 1-6, 2002.