UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

APLICAÇÃO RADAR PARA VISUALIZAÇÃO DA IMPLANTAÇÃO DE UMA REDE MESH

HENRIQUE OECKSLER BERTOLDI

BLUMENAU
2016

HENRIQUE OECKSLER BERTOLDI

APLICAÇÃO RADAR PARA VISUALIZAÇÃO DA IMPLANTAÇÃO DE UMA REDE MESH

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Franciso Adell Péricas, Mestre - Orientador

BLUMENAU
2016

APLICAÇÃO RADAR PARA VISUALIZAÇÃO DA IMPLANTAÇÃO DE UMA REDE MESH

Por

HENRIQUE OECKSLER BERTOLDI

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Franciso Adell Péricas, Mestre – Orientador, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Prof. Paulo Fernando da Silva, Mestre – FURB

Blumenau, 07 de julho de 2016

Dedico este trabalho à minha família, que exaustivamente me mostrou o valor do conhecimento e como persegui-lo.

AGRADECIMENTOS

À minha família por me dar suporte nas horas que mais necessitei.

Aos amigos, que estavam presentes para me ouvir, incentivar, dar dicas e conselhos no decorrer do desenvolvimento deste trabalho.

Ao meu orientador Francisco Adell Péricas, que acreditou na ideia do meu trabalho mesmo sabendo dos grandes desafios a serem enfrentados. Por sempre ter dado força para continuar, mesmo em algumas situações de total desespero no decorrer deste trabalho. E também por passar sábios concelhos técnicos sobre o assunto.

O sucesso é ir de fracasso em fracasso sem perder o entusiasmo.

Winston Churchill

RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação radar Wi-Fi para visualização da implantação de uma rede *mesh*. Esta rede *mesh* foi implantada utilizando roteadores TL-WR841ND da TP-LINK, que são roteadores de baixo custo. Foi realizada a compilação do *firmware* OpenWrt juntamente com o protocolo de roteamento B.A.T.M.A.N para instalação nos roteadores. A configuração utilizada nestes roteadores permite a expansão da rede somente com a adição de mais roteadores configurados da mesma forma. O aplicativo Wi-Fi foi desenvolvido utilizando a linguagem C# na plataforma Windows. Utilizou-se a biblioteca NativeWiFi para obtenção de informações das conexões Wi-Fi. Para execução do radar, o usuário informa somente o nome da rede Wi-Fi a ser monitorada e clica no comando para pesquisar. É possível realizar novas buscas informando outros nomes de rede na mesma tela. Foi realizado um experimento provando o funcionamento da rede *mesh* com o uso do aplicativo radar Wi-Fi em uma situação de uso real.

Palavras-chave: Radar wi-fi. Redes mesh. OpenWrt.

ABSTRACT

This paper presents the development of a radar Wi-Fi application for the visualization of deploying a *mesh* network. This *mesh* network was deployed using TL-WR841ND TP-LINK routers, which are low cost routers. The compile of OpenWrt *firmware* was made with the B.A.T.M.A.N routing protocol for installation in these routers. The configuration used in these routers allows the expansion of the network by adding more routers configured in the same way. The Wi-Fi application was developed using C# language in the Windows platform. The NativeWiFi library was used to obtain information from Wi-Fi connections. For radar execution, the user just need to inform the Wi-Fi network name to be monitored and click in the command to search. It is possible to perform new searches by just inform other network names on the same screen. An experiment was made in order to prove the operation of the *mesh* network using Wi-Fi radar application in a real-use situation.

Key-words: Wi-fi radar. Mesh networks. OpenWrt.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AP - *Access Point*

BSS - *Basic Service Set*

ESS - *Extended Service Set*

LAN - *Local Area Network*

MAC – *Media Access Control*

SSID - *Service Set Identifier*

UCI - *Unified Configuration Interface*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, com o desenvolvimento do padrão IEEE 802.11, mais conhecido como Wi-Fi, tem havido um grande aumento no desenvolvimento de aplicações utilizando esta tecnologia, alcançando a maioria das atividades humanas (GÓMEZ, 2012, p. 1). O principal uso dessa tecnologia está na área de sistemas de telefonia móvel, onde a comunicação entre eles deve ser feita sem haver interrupções. Na área de redes de computadores também há a busca por essa mobilidade, que pode ser alcançada por redes em malha sem fio, também chamadas de redes *mesh* (ARRUDA, 2010, p. 13).

Segundo Filippi (2014), as redes *mesh* devem ser levadas mais a sério por possuírem benefícios desde arquiteturais até políticos. Em 2014, depois que um tufão percorreu o arquipélago das Filipinas, várias cidades não puderam receber ajuda comunitária pois perderam sua conexão com uma rede viável e confiável. Em um país onde 90% da população tem acesso a dispositivos móveis, a implementação de uma rede *mesh* emergencial poderia ter salvo vidas.

Em redes de arquitetura normal, sempre há uma autoridade central ou uma organização responsável por gerenciar a rede em geral. Redes *mesh* conectam *notebooks* e *smarthphones*, e de forma independente conectam-se entre si sem a intervenção de uma unidade central de controle. Dessa forma ocorre uma dinamicidade de ajuste de acordo com a largura de banda e a densidade das conexões da rede. Essas conexões dinâmicas entre os nós da rede permitem o uso de múltiplas rotas para comunicação dos nós através da rede, o que se torna assim uma rede robusta (FILIPPI, 2014).

Para Anthony (2014), a utilização de redes *mesh* com a adição de uma criptografia sendo utilizada em larga escala no mundo torna-se um dos negócios mais desruptivos que se pode imaginar. Porém, segundo Filippi (2014), essas redes ainda são vistas como uma rede emergencial ao invés de uma rede comum para comunicação. Outra barreira são os desafios tecnológicos envolvidos na utilização dessas redes, que vão em contrapartida dos benefícios humanitários trazidos. Na Alemanha o Freifunk e na Espanha o GuiFi.net são ferramentas construídas com o objetivo de trazer facilidade na implantação de uma rede *mesh* simples e com uma estrutura mínima para funcionamento.

Diante do exposto, propõe-se a implantação de uma rede *mesh* sem entrar no escopo de segurança dessa rede. Ainda, propõe-se o desenvolvimento de uma aplicação radar para essa rede em malha, com objetivo de visualizar a topologia dela em relação a conexão do cliente.

1.1 OBJETIVO

O objetivo desse trabalho é o desenvolvimento de um aplicativo radar para visualização da topologia dos roteadores de uma rede *mesh*, bem como a implantação dessa rede *mesh*.

Os objetivos específicos do trabalho são:

a) utilizar roteadores Wi-Fi que servirão como “nós” *mesh* e também como “nós” *access point* (AP);
b) utilizar recursos disponíveis no *Firmware* OpenWrt para tornar possível estas duas funcionalidades nestes “nós”;
c) desenvolver uma aplicação que permitirá ao usuário visualizar em forma de radar e em forma gráfica a topologia (ligações) dos roteadores implantados na rede.

1.2 ESTRUTURA

Este trabalho está divido em quatro capítulos, onde no primeiro capítulo é apresentada a introdução e os objetivos do trabalho. O segundo capítulo trata de apresentar a fundamentação teórica sobre as redes sem fio locais, redes *mesh*, algoritmos de roteamento, protocolos de roteamento para redes *ad hoc* e por último os *firmwares* DD-WRT e OpenWrt. No terceiro capítulo é demonstrada a especificação e requisitos da rede e do aplicativo, assim como a implantação e implementação dos mesmos, e posteriormente é apresentada a operacionalidade da aplicação. Finalmente, no quarto e último capítulo são relatadas as conclusões, os resultados obtidos e suas possíveis extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está organizado em sete seções. A seção 2.1 aborda as redes sem fio locais, sua história, seus conceitos e os tipos existentes. A seção 2.2 apresenta os conceitos sobre as redes em malha ou redes *mesh*. A seção 2.3 apresenta os algoritmos de roteamento, seus conceitos e os dois principais algoritmos utilizados. A seção 2.4 aborda os protocolos de roteamento para redes *ad hoc*, bem como os tipos disponíveis e ainda dois dos protocolos mais utilizados. As seções 2.5 e 2.6 apresentam respectivamente os *firmwares* DD-WRT e OpenWrt, juntamente com suas características e funcionalidades. E por fim, a seção 2.7 apresenta os trabalhos correlatos.

2.1 REDES SEM FIO LOCAIS PADRÃO IEEE 802.11

Logo que surgiram os primeiros *notebooks*, em meados da década de 1990, a mobilidade trazida por eles fez com que muitas pessoas não quisessem mais utilizar as conexões *Local Área Network* (LAN) com fio para se conectar à rede, e sim ansiassem por uma conexão sem fio. Para suprir essa necessidade, grupos começaram a trabalhar para tornar essa ideia realidade e de forma simples e eficaz instalaram transmissores e receptores de rádio nos ambientes e *notebooks* para que fosse possível fazer a troca de dados através de ondas de rádio (TANENBAUM, 2003, p. 73).

Entretanto estes aparelhos de rádio eram produzidos por diversas marcas, cada qual com características distintas, o que levou à incompatibilidade na comunicação sem fio entre os *notebooks* de maneira abrangente. Depois de muitas discussões sobre os problemas envolvidos nessa nova tecnologia, a indústria decidiu que a padronização geral de uma LAN sem fio seria o ideal para resolvê-los, e assim o comitê da IEEE (2015) ficou responsável pela padronização do padrão 802.11, popularmente conhecido como Wi-Fi (TANENBAUM, 2003, p. 73).

Já existem diversos padrões IEEE 802.11 especificados, entre eles os mais comuns são 802.11a, 802.11b, 802.11g e 802.11n. São apresentadas algumas de suas características técnicas no Quadro 1.

Quadro 1 - Padrões 802.11 com suas características técnicas

Padrões	Frequências	Taxa de Dados
802.11b	2,4 GHz - 2,485 GHz	até 11 Mbit/s
802.11g		até 54 Mbit/s
802.11a	5,1 GHz - 5,8 GHz	até 54 Mbit/s
802.11n	2,4 GHz e 5,0 GHz	até 600 Mbit/s

Fonte: Adaptado de Ribeiro (2007, p. 19).

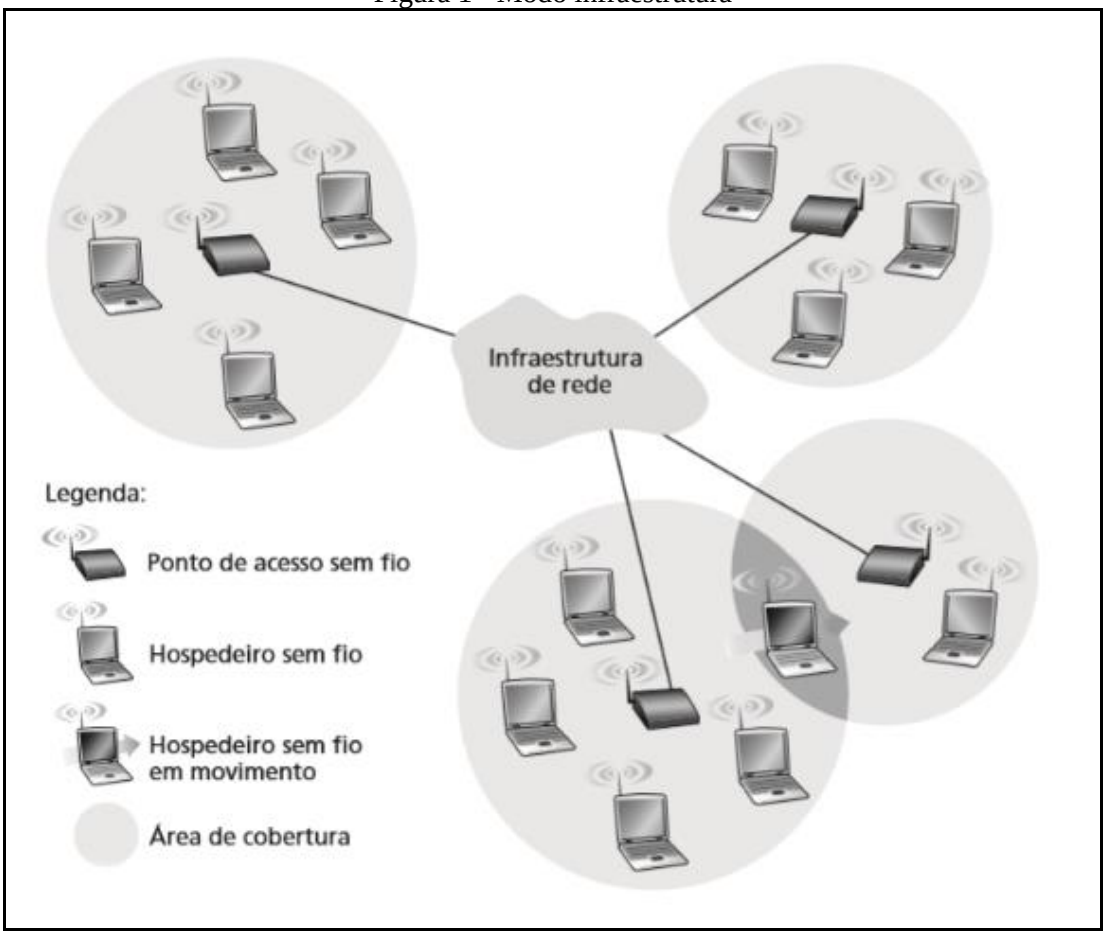
As LANs sem fio 802.11b/g operam nas frequências 2,4 GHz - 2,485 GHz não licenciadas e competem no aspectro de telefones e micro-ondas operados na frequência 2,4 GHz. Mesmo que o padrão 802.11a tenha uma taxa de transferência significativamente grande, por funcionar em uma frequência mais alta perde poder de transmissão em longas distâncias. O padrão LAN sem fio 802.11g é compatível com o padrão 802.11b e permite transição direta dos clientes 802.11b. Ainda possui a mesma taxa de transferência significativamente mais alta do padrão 802.11a, porém por operar na faixa de frequência menor como a do 802.11b, pode ser considerado o melhor dos dois mundos (KUROSE, 2010, p. 386-387). Segundo Tavares (2008, p. 37), o padrão 802.11n é relativamente novo em relação aos outros três padrões e foi projetado para atender serviços de *streaming*, transmissão de HDTV em alta velocidade e em longas distâncias.

Todos esses padrões citados podem funcionar em modo infraestrutura ou em modo *ad hoc* (KUROSE, 2010, p. 386-387).

2.1.1 Modo Infraestrutura

O modo infraestrutura, apresentado na Figura 1, é o modo mais comum para conectar clientes, como *notebooks* e *smartphones*. Em geral, redes residenciais e empresariais utilizam esse modo por ser o mais simples e prático para implantação (ANTUNES, 2012, p.24).

Figura 1 - Modo infraestrutura

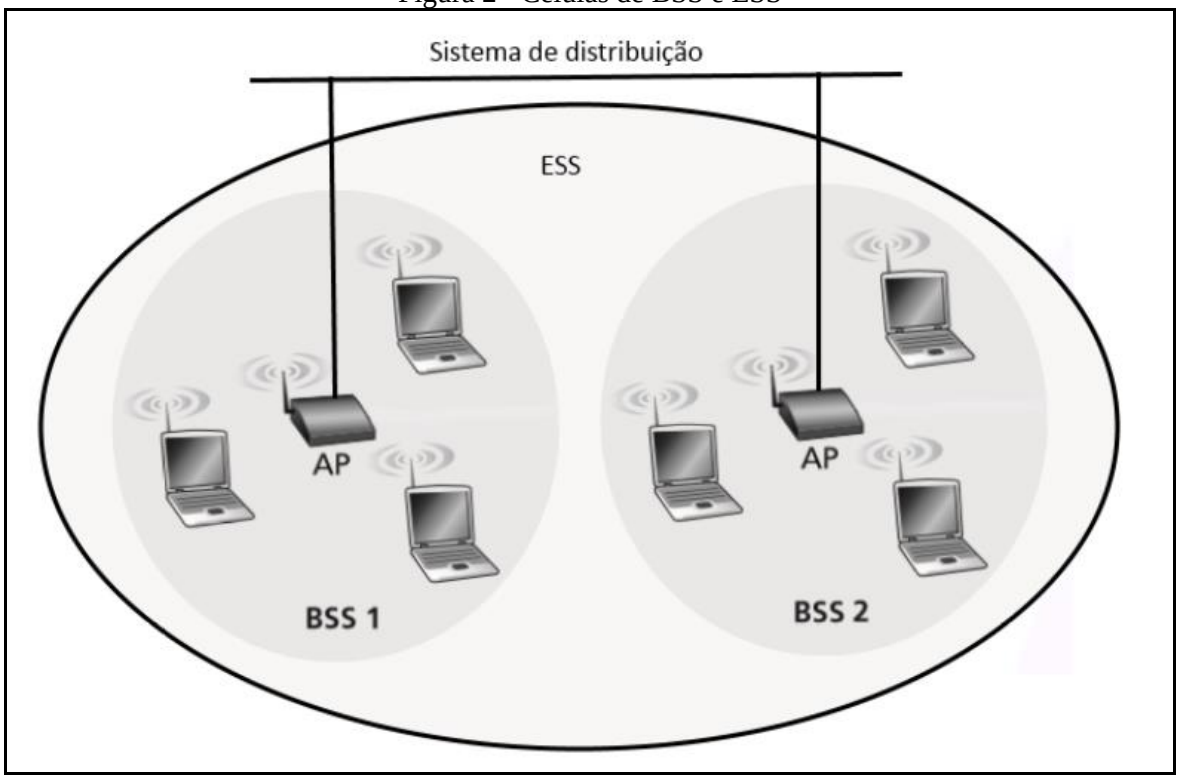


Fonte: Kurose (2010, p. 378).

Conforme Antunes (2012, p.24), nesse modo os pontos de acesso da rede se comunicam através da conexão física – cabeada – com o interconector (comutador ou roteador) responsável por gerenciar toda a rede e disponibilizar Internet aos pontos de acesso, como demonstrado na Figura 1. Ou seja, os pontos de acesso não conseguem se comunicar diretamente entre si, mesmo que estejam muito próximos uns dos outros.

Dois tipos de estruturas podem ser utilizados nesse modo, o *basic service set* (BSS) ou conjunto básico de serviço e o *extended service set* (ESS) ou conjunto estendido de serviço. Estes dois modos de estrutura podem ser visualizados na Figura 2.

Figura 2 - Células de BSS e ESS



Fonte: Adaptado de Kurose (2010, p. 378).

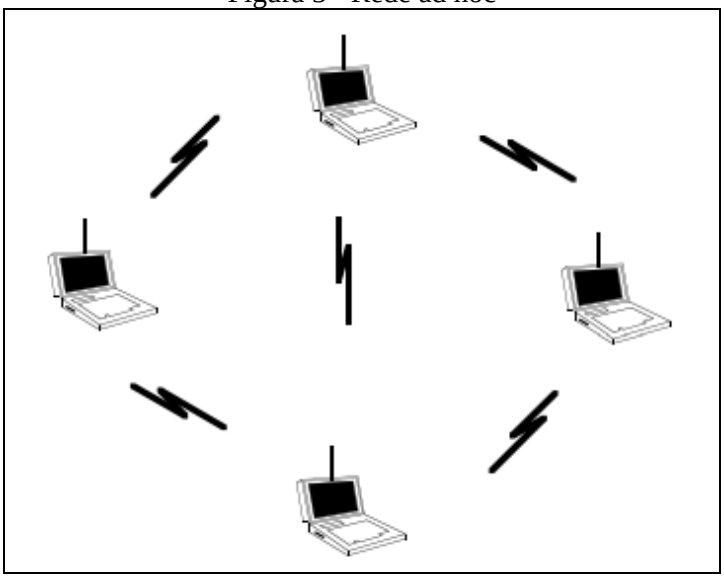
Segundo Vasseur (2010, p. 159), o BSS é formado por uma célula composta pelas estações clientes e por somente um AP, que possui um *service set identifier* (SSID), geralmente composto por caracteres textuais. Para que o cliente se conecte nessa rede, ele precisa se associar com este AP através do seu SSID.

O ESS é a junção de duas ou mais células de BSS interconectadas por um sistema de distribuição. Geralmente esse sistema consiste em uma conexão cabeada entre os pontos de acesso de cada BSS ou qualquer outro tipo de comunicação de rede (STALLINGS, 2007, p. 533). Vale ressaltar que o ESS tem um SSID associado a ele, ou seja, cada AP do ESS tem o mesmo SSID configurado. Desta forma o cliente será conectado no ESS através de um AP invisível para ele (VASSEUR, 2010, p. 159).

2.1.2 Modo ad-hoc

Redes *ad hoc* são formadas por estações que se comunicam diretamente entre si sem o intermédio de um interconector como nas redes de infraestrutura. Essas estações são aparelhos móveis como *notebooks* e *smartphones*. Conforme Figura 3, essa rede não possui comunicação com o mundo exterior. Isso acontece porque seu objetivo é estabelecer uma conexão rápida entre dispositivos numa situação imediata como uma reunião ou uma conferência empresarial (STALLINGS, 2007, p. 526).

Figura 3 - Rede ad hoc



Fonte: Stallings (2007, p. 527).

2.2 REDE EM MALHA IEEE 802.11

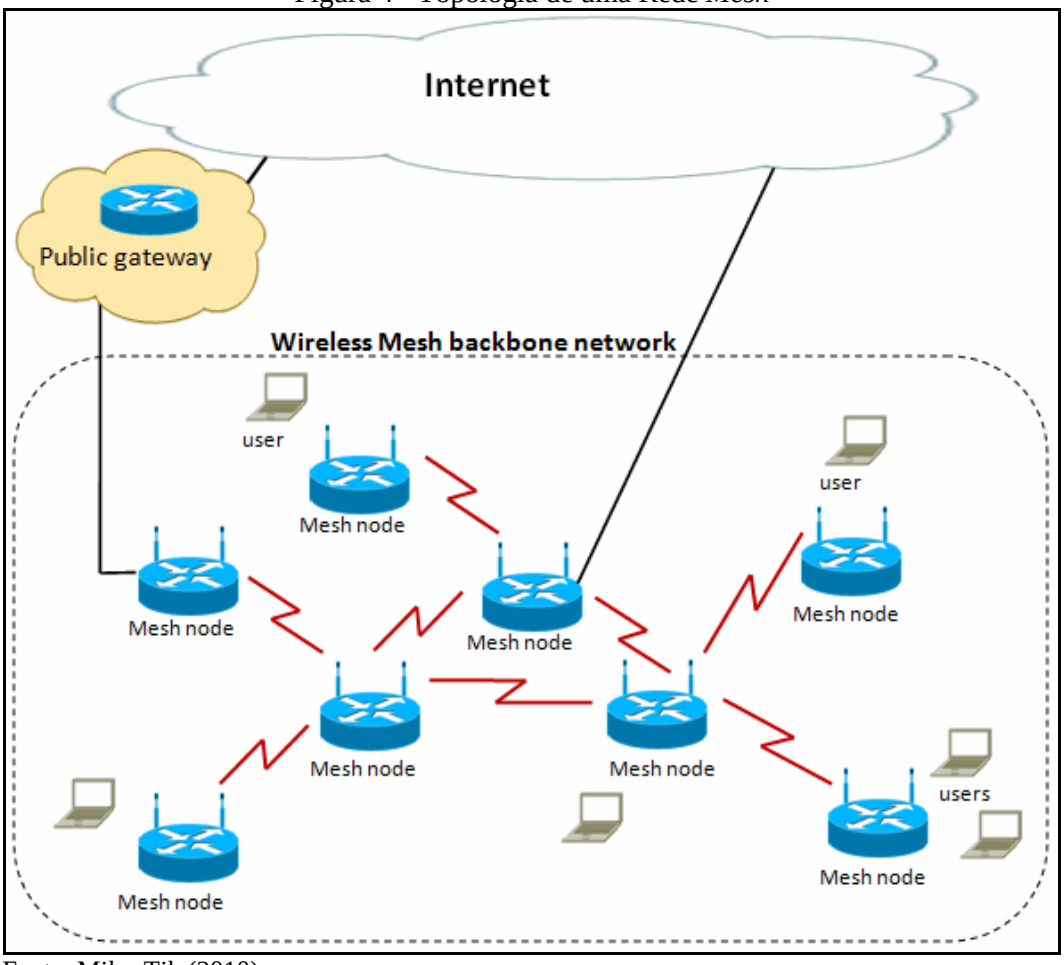
As Redes em Malha IEEE 802.11 ou também conhecidas como redes *mesh* são formadas por um conjunto de nós que se comunicam entre si sem a utilização de fios. Por isso, são amplamente utilizadas em áreas extensas onde somente um AP não daria cobertura ou ainda em locais onde a utilização de cabeamento de uma rede Wi-Fi normal não seja viável (ANTUNES, 2012, p.37-39).

Conforme Antunes (2012, p.37-39), soa muito como uma nova tecnologia no mundo *wireless*, porém apenas pode ser considerada como um novo conceito, que utiliza os padrões IEEE 802.11 e evolui na base das redes *ad hoc*. Os nós utilizados nesta rede divergem dos utilizados nas redes *ad hoc*, pois tendencialmente possuem maior poder de processamento, capacidade de armazenamento e gestão de energia.

Dispostos na Figura 4, os nós Mesh podem ser pequenos radiotransmissores exclusivos para redes *mesh* ou ainda roteadores *wireless* que suportam a instalação de um *firmware* com recursos disponíveis para configurar uma rede *mesh* (TAVARES, 2008, p.49). Esses nós têm a capacidade de trocar dados diretamente entre si sem a intervenção de uma autoridade central. Conforme necessário, cada nó se encarrega do tráfego um do outro.

Para disponibilizar Internet na rede é necessário, conforme Figura 4, que somente um nó Mesh esteja conectado à um aparelho chamado de *Public gateway*, geralmente um *modem* ou *switch*, através de um cabo *Ethernet*. A partir deste nó comumente chamado de nó Mesh *Gateway*, todos os outros nós da rede podem ter acesso à Internet.

Figura 4 - Topologia de uma Rede *Mesh*



Fonte: MikroTik (2010).

Visto que cada nó é uma peça fundamental para a troca de mensagens na rede, é necessário um mecanismo auto gerenciável que identifique possíveis problemas na rede. Caso um nó fique indisponível por um motivo qualquer como queda de energia ou por ser destruído, se bem planejada esta rede torna-se "auto curável", ou seja, mesmo que um nó caia ela é capaz de identificar e corrigir automaticamente esse problema em relação ao tráfego de dados na topologia da rede. Para aumentar a extensão da rede, basta adicionar mais nós nas periferias da rede ou ainda em pontos estratégicos onde está havendo problemas de congestionamento ou lentidão, e de forma automática a rede vai integrá-los (WNDW, 2007, p. 53-54).

Em essência, as redes *mesh* não se comunicam diretamente com clientes comuns como *notebooks* e *smartphones*, a não ser que estes estejam configurados para funcionar como nó Mesh, que é uma prática incomum. Para conectá-los em seu funcionamento normal, é necessário configurar pontos de acessos que irão disponibilizar rede Wi-Fi comum para conexão destes dispositivos. Para isso, pode-se utilizar aparelhos *access point* simples

conectados à um nó Mesh através do cabo *Ethernet*. Depois, basta configurar essa rede Wi-Fi normal com SSID e afins (AKYILDIZ, 2005, p. 448).

O seu funcionamento, como citado, difere das redes convencionais. Isso devido ao modo como ela é tratada e ao protocolo de roteamento usado.

2.3 ALGORITMO DE ROTEAMENTO

Dentro de um roteador, pode-se imaginar que há dois processos, roteamento e encaminhamento, e é muito importante fazer a distinção entre eles. Roteamento trata de decidir quais rotas utilizar, através do preenchimento e atualização das tabelas de roteamento. O encaminhamento trata do que acontece com um pacote que chega, verificando uma saída para ele na tabela de roteamento (TANENBAUM, 2003, p. 372-373).

Segundo Kurose (2006, p. 276), no processo de roteamento é que entram os algoritmos de roteamento. São responsáveis por descobrir um melhor caminho entre um roteador origem e um destino dentro de uma rede. Um bom caminho é considerado aquele que tem o menor custo ou menor número de saltos. Ainda, há políticas do mundo real que impactam nessa regra e acaba complicando ainda mais, como por exemplo a restrição de um roteador *X* de uma organização *Y* não poder repassar pacotes vindos de um roteador *Z* de uma organização *H*.

Uma das classificações dos algoritmos de roteamento é entre estáticos e dinâmicos. Nos estáticos as rotas mudam muito lentamente e geralmente com alguma interação humana alterando a tabela de roteamento. Já os dinâmicos mudam suas rotas conforme mudanças na topologia da rede ou na carga de tráfego da rede (KUROSE, 2006, p. 277). Este último é o mais utilizado, visto que as redes atuais tendem sempre a aumentar sua carga conforme o uso. Os dois algoritmos dinâmicos mais conhecidos atualmente são o roteamento com vetor de distância e o roteamento por estado de enlace (TANENBAUM, 2003, p. 379).

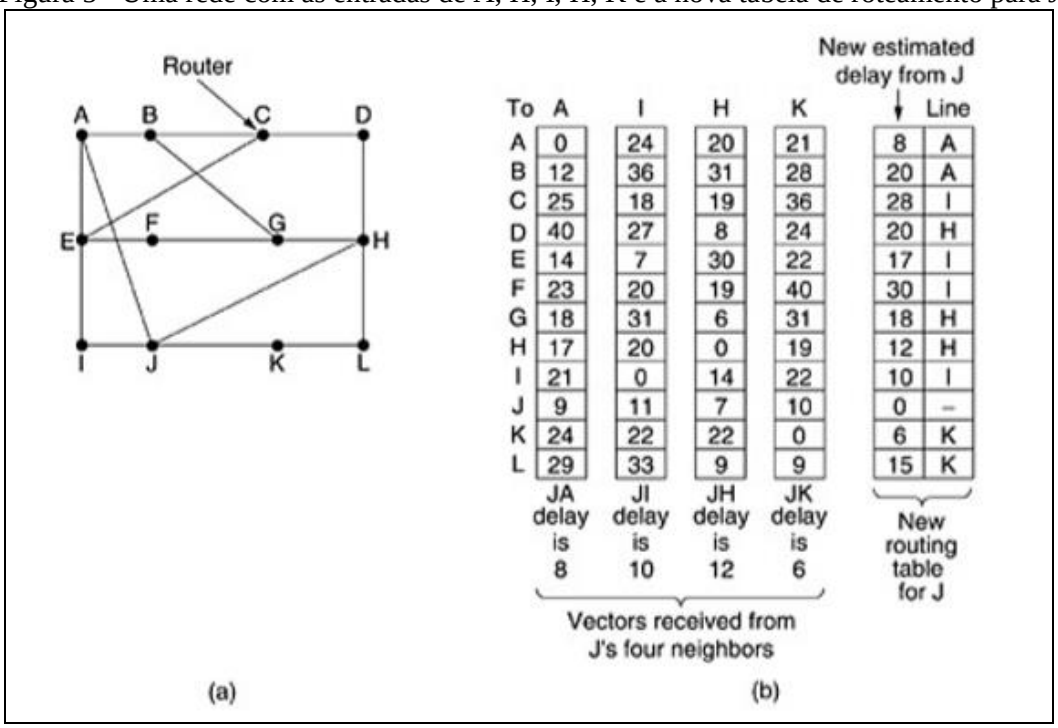
2.3.1 Algoritmo com vetor de distância

O algoritmo com vetor de distância define que cada nó recebe alguma informação dos vizinhos conectados a ele diretamente, e com estas informações monta o vetor de distância inicial. Quando feito isso, ele repassa esse vetor aos seus vizinhos. Ao receber esse vetor, cada vizinho recalcula sua tabela de roteamento e se houve alguma mudança, repassa essa tabela para todos os seus vizinhos. Essa interação continua até que não haja mais informação para ser trocada entre os nós. Nesse momento, o algoritmo entra em estado chamado de inatividade. Mesmo em inatividade, caso alguma tabela sofra alguma alteração, ela será

repassada novamente para todos seus vizinhos, que farão novamente seus cálculos e repassarão adiante sua tabela caso haja alguma alteração (KUROSE, 2003, p. 281-285).

Essa tabela consiste em uma entrada para cada um dos nós da rede. Como pode ser visto na Figura 5 (b), essa entrada possui duas partes: a linha de saída preferível para o nó em questão e uma estimativa de tempo ou distância para percorrer (TANENBAUM, 2003, p. 380).

Figura 5 - Uma rede com as entradas de A, H, I, H, K e a nova tabela de roteamento para J



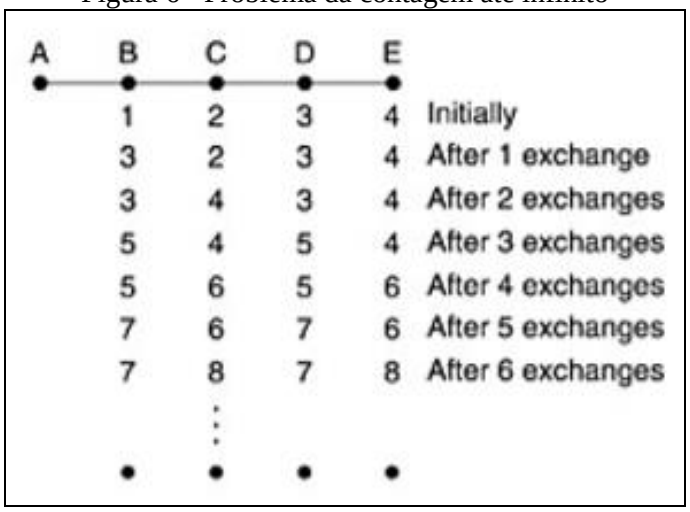
Fonte: Tanenbaum (2003, p. 381).

A Figura 5 parte (b) detalha o recebimento das informações que todos os nós vizinhos do nó *J* enviaram para ele e detalha também a tabela de roteamento de *J* após os cálculos. *H* e *K* informaram que o seu tempo para chegar em *L* é de 9ms, porém o tempo que *J* leva para chegar em *H* é 12ms e para *K* é 6ms. Ou seja, no cálculo feito por *J* para chegar no nó *L* resultou que passando por *H* levará 21ms (9ms + 12ms) e passando por *K* levará 15ms (9ms + 6ms). Com isso o nó *J* gravou em sua tabela que para chegar ao nó *L* precisa sair preferivelmente pelo nó *K* e levará em torno de 15ms para chegar. Esse processo acontecerá com todos os nós, não somente com *J* (TANENBAUM, 2003, p. 381).

Um problema grave do algoritmo é que ele não consegue lidar muito bem com más notícias, como a perda de um nó ou de um enlace na rede. Para explicar isso, a Figura 6 apresenta uma topologia de rede com os nós *A*, *B*, *C*, *D*, *E* e suas distâncias para o nó *A*. Inicialmente o protocolo já está em estado inativo. Imaginando que o nó *A* saia do ar ou aconteça algum problema no enlace com *B*, na primeira nova troca de dados *B* não receberá

retorno de *A*, porém *C* informará que possui uma rota para *A* com custo 2. Então *B* concluí que possuí uma rota para *A* de custo 3 com saída para *C*. Mesmo que *B* saiba, e ele sabe, que *C* precisa passar por ele mesmo para chegar em *A*, pode acontecer que existam outras rotas alternativas para *C* chegar em *A* também com peso 2. Entretanto em algum momento todos os nós perceberão que *A* está inativo, porém pode demorar bastante até que isso aconteça. Já foram propostos alguns meios para contornar esse empecilho, como a divisão horizontal de inversão envenenada, entretanto nenhum deles se mostrou eficaz em termos gerais (TANENBAUM, 2003, p. 382-383).

Figura 6 - Problema da contagem até infinito



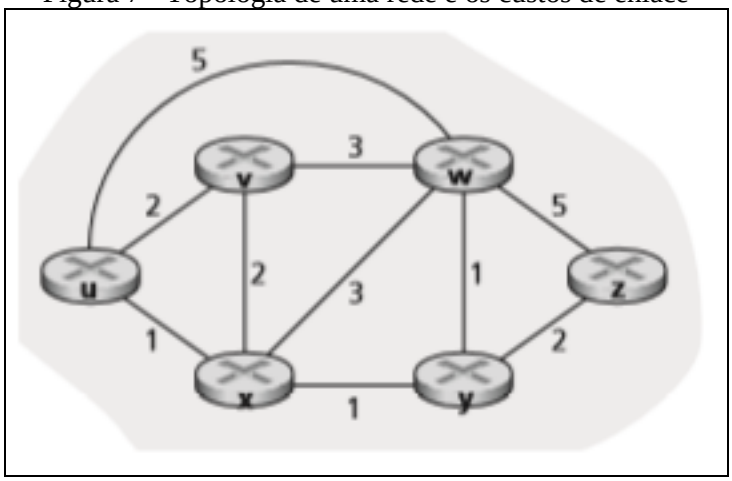
Fonte: Tanenbaum (2003, p. 382).

2.3.2 Algoritmo por estado de enlace

O algoritmo de roteamento por estado de enlace veio para substituir o algoritmo de roteamento com vetor de distância, em 1979. Isso devido alguns problemas que ele vinha apresentando, como a unidade métrica utilizada na tabela de roteamento que não leva em consideração a largura de banda ao selecionar as rotas. Outro problema impactante era o da contagem até o infinito (TANENBAUM, 2003, p. 383).

Para iniciar a execução deste algoritmo, cada nó da rede fica responsável por medir o custo de enlace para cada vizinho e repassar essas informações para todos os outros nós da rede. Ao final desta etapa, todos os nós têm a mesma visão da topologia da rede, com os custos e a identificação de cada enlace disponível na rede. Para elucidar sua execução, a Figura 7 apresenta a topologia de uma rede fictícia mostrando os valores dos enlaces disponíveis (KUROSE, 2006, p. 278).

Figura 7 - Topologia de uma rede e os custos de enlace



Fonte: Kurose (2007, p. 276).

Segundo Kurose (2007, p. 279), para calcular uma rota utiliza-se o algoritmo de *Dijkstra* sobre as informações já coletadas. Ele consiste em calcular o caminho de menor custo de um nó origem para todos os outros nós. Levando em consideração que o algoritmo de *Dijkstra* será executado tomando como nó origem o nó *u*, a Figura 8 apresenta o passo a passo da execução e o resultado obtido, com *N'* sendo o subconjunto de nós já validados, *D(x)* o custo do menor caminho entre o nó fonte e o nó destino, *p(x)* o nó anterior ao nó atual de *N'*.

Figura 8 - Execução do algoritmo de *Dijkstra*

Etapa	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Fonte: Kurose (2007, p. 279).

Cada etapa funciona da seguinte maneira:

a) etapa 1: inicia-se o subconjunto de nós *N* com o nó inicial *u*, e para cada nó que tem enlace com este preenche-se *D(valor do enlace)* e *p(nó anterior)* que é o próprio *u*. Vale ressaltar que os nós *y* e *z* não possuem enlace com ele, por isso recebem infinito (∞);

b) etapa 2: verifica-se o nó da etapa 1 com o menor custo que é *x*, então *x* vai para *N*. Para alcançar *v*, o menor custo continua sendo por *u* com custo 2, e para chegar em *w*, o menor custo não é mais só por *u* e sim passando por *x* ($u \rightarrow x \rightarrow w = 4$), logo essa informação de *w* foi alterada;

c) etapa 3: verifica-se o nó com o menor custo da etapa 2, *v* e *y* possuem custo 2, arbitrariamente *y* foi escolhido e vai para *N*. Novamente verificou-se que para

chegar em *w* o menor custo é por *y* e finamente foi possível alcançar *z* com valor 4.

Dessa forma o algoritmo é executado até que todos os nós estejam contemplados em *N*, e ao final é possível encontrar o menor caminho para chegar em qualquer nó a partir de *u*. Com essas informações é possível construir a tabela de repasse (KUROSE, 2007, p. 279).

2.4 PROTOCOLO DE ROTEAMENTO PARA REDE MÓVEIS *AD HOC*

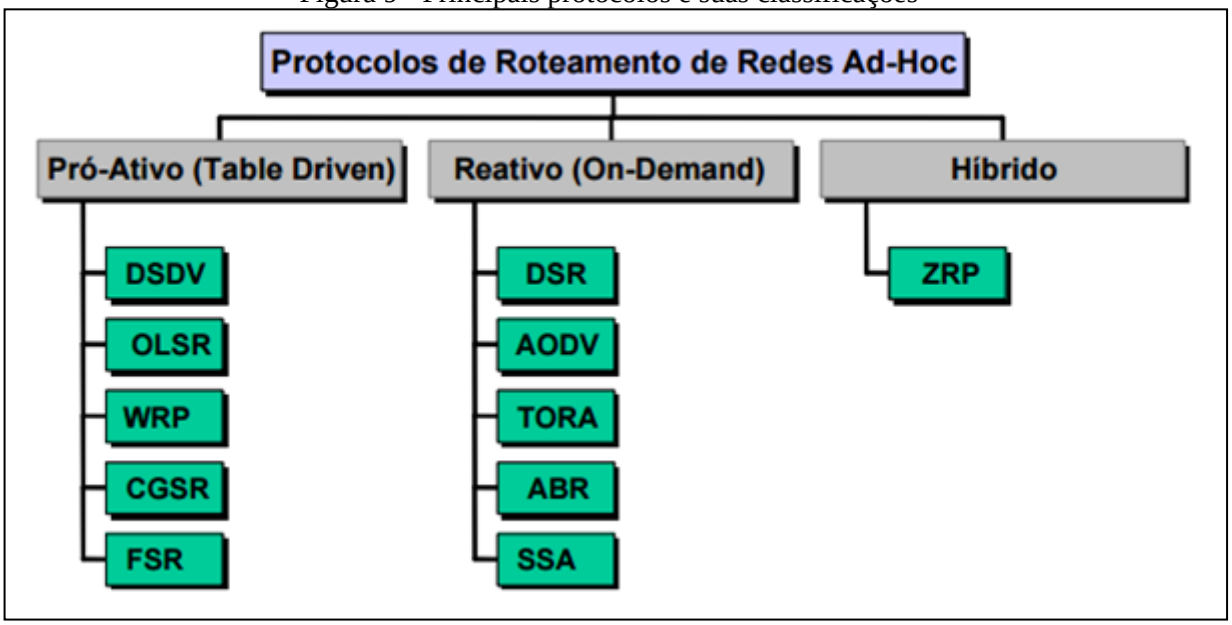
Em uma rede em malha, para que um pacote de dados trafegue de um nó origem até um nó destino podendo transitar entre nós intermediários, deve ser calculada uma rota. Os protocolos de roteamento são responsáveis por realizar essa tarefa (ARRUDA, 2010, p. 18).

Devido a dinamicidade das redes *mesh*, é necessário que o protocolo recalcule as rotas periodicamente com baixo custo de transmissão nas operações necessárias. Para encontrar uma rota ótima, esse cálculo deve ser feito baseado em métricas de qualidade de enlace, descobrimento de fronteiras, largura de banda, latência e controle na segurança de rede (MIGUEL, 2011, p. 42).

Protocolos de roteamento para redes LAN estruturadas ou mesmo para Internet, como o com vetor de distância e por estado de enlace não são indicados para redes *ad hoc*. Para suprir essa necessidade, vários algoritmos já foram e estão sendo propostos para as redes *ad hoc* (FERMINO, 2009, p. 35). Em geral, estes protocolos podem ser classificados em três grandes grupos: Proativos, Reativos e Híbridos (ROYER, 1999, p. 46). A Figura 9 lista alguns dos principais protocolos já padronizados e utilizados amplamente no mercado.

Outra classificação dos protocolos pode ser feita quanto à técnica de roteamento. Podem ser divididos em duas categorias: roteamento na origem (*source routing*) e roteamento salto-a-salto (*hop-by-hop*). No roteamento por origem um nó de origem carrega consigo o endereço e a sequência de todos os nós pelos quais deverá passar para chegar no destino, e assim os nós da rede ficam responsáveis somente por guiá-lo na rota. Já no roteamento salto-a-salto, o nó origem traz consigo o endereço do próximo nó e o endereço do nó destino, assim os outros nós da rede se encarregam de informar o endereço do nó a seguir com base na sua tabela de roteamento (FERMINO, 2009, p. 36).

Figura 9 - Principais protocolos e suas classificações



Fonte: Ribeiro (2007, p. 43).

2.4.1 Protocolos Proaativos

Os protocolos proativos visam manter uma tabela de informações consistentes e atualizadas da rota de todos nós para todos os outros nós da rede, e por isso também são conhecidos como dirigidos por tabela (*table-driven routing protocol*). Cada nó fica responsável por manter uma ou mais tabelas com informações de todas as rotas, e ainda devem estar sempre atentos a qualquer mudança na topologia de rede, pois caso ocorra alguma, novas informações devem ser propagadas por toda a rede com o intuito de manter as tabelas atualizadas (ROYER, 1999, p. 46).

Uma vantagem nesse protocolo é o descobrimento quase instantâneo de uma rota quando há a necessidade de comunicação com um destino específico, visto que todas a rotas já são conhecidas por todos os nós. Porém, essa mesma característica traz consigo uma desvantagem, o alto consumo de energia e de banda para manter todas as rotas atualizadas (RIBEIRO, 2007, p. 43).

2.4.2 Protocolos Reativos

Sendo praticamente o oposto dos protocolos proativos, nos protocolos reativos as rotas são criadas somente conforme demanda. Quando um nó origem deseja se comunicar com um nó destino, inicia-se o processo de descobrimento de rota. Esse processo só termina quando uma rota é encontrada ou quando todas as rotas possíveis já foram exploradas, mas nenhuma foi encontrada. Caso tenha encontrado alguma, ela ficará armazenada até que o nó destino fique inacessível ou ela não seja mais necessária (ROYER, 1999, p. 48).

A vantagem neste protocolo é a redução do consumo de energia e de banda, pois ele evita o desperdício de tráfego de pacotes na rede para descobrimento de rota caso não sejam necessários. Em contrapartida, a desvantagem significativa é o possível atraso na descoberta de rotas, principalmente em redes onde a topologia pode sofrer constante mudança (RIBEIRO, 2007, p. 54).

2.4.3 Protocolo Híbridos

Os protocolos híbridos reúnem características dos protocolos proativos com características dos protocolos reativos. Foram projetados para lidar com grupos, onde cada grupo é formado por um conjunto de nós que possuem alcance entre si. Para descobrimento de rotas dentro de um grupo são utilizados algoritmos de roteamento pró-ativo, e para descobrimento de rotas entre os grupos são utilizados algoritmos de roteamento reativo (FERMINO, 2007, p. 38)

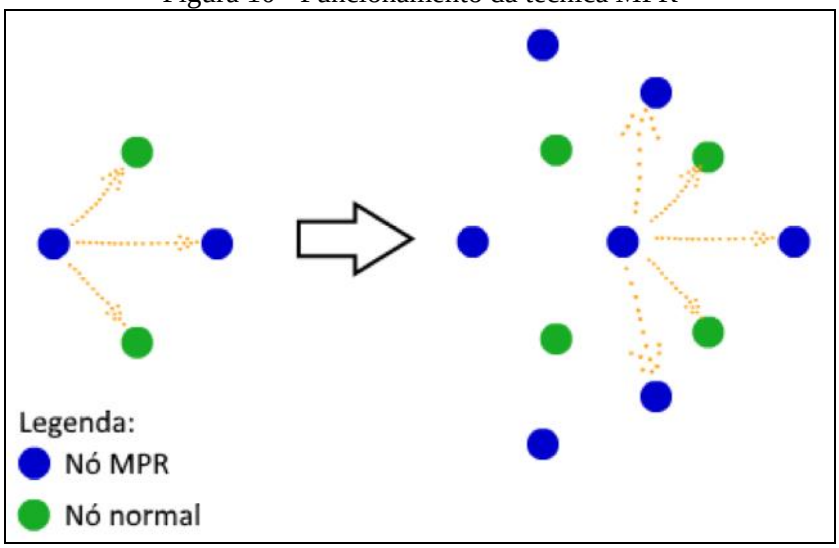
2.4.4 Optimized Link State Routing Protocol (OLSR)

O OLSR é um protocolo de natureza pró-ativa e ainda uma otimização dos algoritmos de roteamento por estado de enlace para redes *ad hoc*. Sua primeira otimização é a redução do tamanho dos pacotes de controle, ao invés de guardar todas as informações de todos os enlaces da rede como no algoritmo por estado de enlace, guarda somente dos enlaces em relação ao conjunto de nós MPR (*Multipoint Relay*). A segunda otimização é sobre inundação (*flooding*) da rede com os pacotes de controle, ao invés de todos os nós fazerem a transmissão destes pacotes, somente os nós MPR podem executar tal tarefa (CLAUSEN, 2003, p. 2).

Conforme Fermino (2009, p. 52 – 60), uma das duas mensagens de controle utilizadas pelo OLSR é a mensagem *HELLO*. Os nós da rede periodicamente enviam mensagens *HELLO* no modo *broadcast* a todos seus vizinhos a 1 salto de distância informando dados do estado de enlace. Todo vizinho receberá esta mensagem, processará esse pacote de informações, porém não repassará adiante. Com a troca destas informações, todo nó terá conhecimento de todos seus nós vizinhos que estão a 2 saltos de distância. Com isso, já possuem as informações necessárias para decidir quem serão seus nós MPR. A seleção dos nós MPR por parte de cada nó deve ser feita de modo que a informação deste nó principal seja capaz de atingir todos os nós que estejam a 2 saltos do nó em questão. Vale ressaltar que quanto mais reduzido for o conjunto de nós MPR necessários, mais eficiente essa técnica será. Depois que os conjuntos MPR foram estabelecidos, somente haverá mudanças caso alguns desses nós falhe ou haja alguma alteração detectada em um nó a 2 saltos de distância.

Somente os nós MPR serão utilizados para formar uma rota. Cada nó MPR mantem uma lista de nós vizinhos a 1 salto com informação do estado de enlace. Periodicamente espalham esta lista por toda a rede, para que sequencialmente cada nó MPR possa calcular o custo para cada nó vizinho seu. A Figura 10 apresenta o envio de mensagem de controle por difusão através dos nós MPR. Nota-se que os nós normais também recebem estas mensagens, porém somente os MPR passam adiante (FERMINO, 2009, p. 52 – 60).

Figura 10 - Funcionamento da técnica MPR

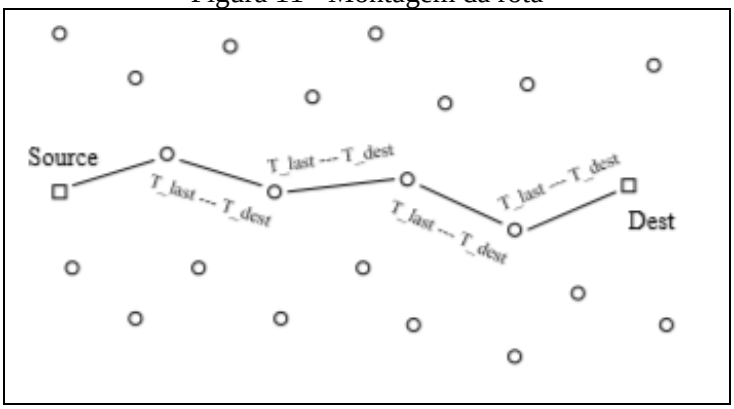


Fonte: Adaptade Moraes (2009).

A segunda mensagem de controle enviada pelo OLSR é a mensagem *Topology Control* (TC). Essa mensagem é enviada de forma *broadcast* periodicamente por todos os nós MPR apenas, processo semelhante à técnica utilizada no algoritmo por estado de enlace. As informações dessas mensagens contêm dados de todos os enlaces do seu conjunto MPR responsável. O intervalo configurado para troca destas mensagens pode ser reduzido na hora que houver, e se houver, alguma alteração na topologia da rede ou nos conjuntos de nós MPR. Cada nó que recebe a mensagem TC fica responsável por armazenar as informações da topologia da rede na tabela de topologia (FERMINO, 2009, p. 52 – 60).

O OLSR utiliza a técnica de roteamento salto-a-salto e a rota é sempre calculada através da sequência de nós MPR, conforme Figura 11. O procedimento utilizado para encontrar a rota é pelo caminho mais curto, para isso é utilizado o algoritmo de busca em largura (CLAUSEN, 2003, p. 5).

Figura 11 - Montagem da rota



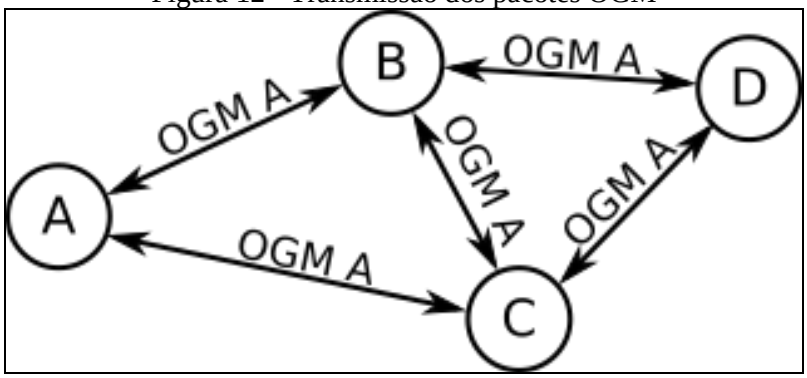
Fonte: Clausen (2003, p. 5).

2.4.5 Better Approach To Mobile *Ad hoc* Networking (B.A.T.M.A.N)

O protocolo OLSR passou por muitas melhorias para tentar resolver os problemas de roteamento impostos por grandes redes *mesh*, porém não foram suficientes para acompanhar o constante crescimento dessas redes cada vez mais comuns. Com isso tornou-se um desafio adentrar os limites desse algoritmo bastante pesado, e para resolver tal problema o protocolo roteamento B.A.T.M.A.N foi criado. Seu objetivo é dividir o conhecimento da melhor rota fim-a-fim de todos os nós com todos os nós da rede. Cada nó mantem somente a informação do melhor nó de saída para cada nó da rede, tornando seu uso ideal em roteadores com pouca memória e poder de processamento (BATMAN, 2006).

Cada nó da rede transmite uma mensagem em modo *broadcast*, chamada OGM, para todos os seus vizinhos. Esses vizinhos retransmitem essa mensagem para todos os seus vizinhos em modo *broadcast* também, e assim por diante até que todos os nós forem alcançados, como mostra a Figura 12. Essa mensagem é bem pequena, geralmente tem tamanho de 52 bytes. Ela contém o endereço do nó originário, o endereço do nó que está transmitindo, o Time-to-Live (TTL, tempo disponível para que o pacote percorra, caso estoure, o pacote é descartado) e ainda um número de sequência. Esse número de sequência é incrementado a cada salto na rede, ou seja, a cada nó percorrido. Por padrão é transmitida a cada 1 segundo (BATMAN, 2006).

Figura 12 - Transmissão dos pacotes OGM



Fonte: OgmV2 (2006).

Quando um nó recebe uma mensagem OGM, ele verifica na sua tabela de roteamento se já possui uma entrada gravada para o nó originário dessa mensagem. Se ainda não possui, ele criará uma entrada para esse nó, que guardará o nó de saída e o custo, representados respectivamente pelo nó atual de transmissão e pelo número de sequência. Se já possui, compara o custo gravado com o custo da nova mensagem e caso o custo da mensagem seja menor, o custo dessa entrada é alterado para o custo dessa mensagem e o nó de saída para o nó atual de transmissão dessa mensagem. Dessa forma cada nó possuirá uma tabela de roteamento com a melhor saída para todos os outros nós, assim sendo possível formar uma rota com menor custo entre quaisquer nós da rede (BATMAN, 2006).

O B.A.T.M.A.N advanced (B.A.T.M.A.N-adv) é a mais recente implementação do B.A.T.M.A.N. Geralmente os protocolos rodam na camada 3 da camada OSI, enquanto B.A.T.M.A.N-adv roda na camada 2. Isso significa que os nós da rede não precisam ter um IP atribuído e assim há mais liberdade na criação da rede, como é possível utilizar IPv4, IPv5, DHCP na camada acima do protocolo. Com isso, para identificação dos roteadores no processo do roteamento é feita pelo endereço *Media Access Control* (MAC) de cada roteador (BATMAN-adv, 2006).

2.5 DD-WRT

O DD-WRT é um *firmware* não oficial desenvolvido sob os termos da GPL (GNU GPL – *General Public Licence*) para roteadores IEEE 802.11a/b/g/d/h/ baseados nos chips *Broadcom* ou *Atheros*. Foi desenvolvido após a empresa Sveatsof Inc. começar a cobrar por seu firmware *AlchemyI*. Atualmente é disponibilizado gratuitamente (DD-WRT, 2007). A mostra a tela de configuração do DD-WRT, já na versão v24-sp2 que é um avanço da versão original (DD-WRT, 2007).

Figura 13 - Tela inicial do DD-WRT

Fonte: DD-WRT (2007).

Atualmente um pequeno grupo de pessoas é responsável por manter seu desenvolvimento ativo. A pessoa que mais se destaca é o próprio criador, BrainSlayer ou Sebastian Gottschall. Por ser código aberto, qualquer usuário pode baixar os fontes e compilar sua própria versão binária do *firmware*. Uma diferença entre a maioria dos projetos de código aberto e o projeto DD-WRT, é que em geral a compilação de um projeto de código aberto irá funcionar em grande parte dos dispositivos para qual foi projetado, já no DD-WRT, uma compilação feita para rodar em um roteador Linksys não irá rodar em outros roteadores como D-Link ou Belkin. Destaca-se ainda o fórum oficial que é bastante ativo e frequentado por vários usuários que utilizam o DD-WRT (JENKINS, 2013).

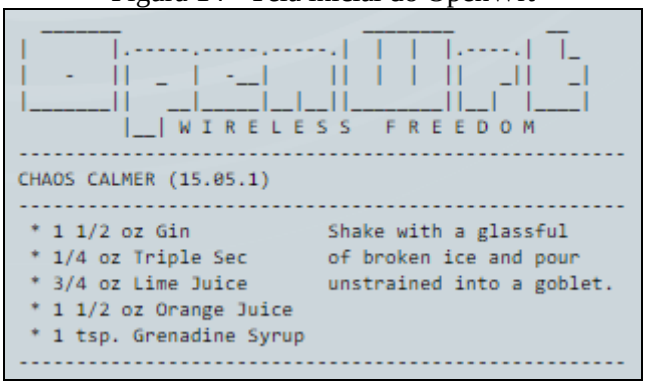
Algumas das funcionalidades mais características são: suporte a redes *ad hoc*; permite utilizar redes *mesh* por dar suporte à protocolos de roteamento proativos como o OLSR; implementa qualidade de serviço (QoS – *Qualite of Service*). Além de muitas outras características que já estão presentes na maioria dos *firmwares* das grandes marcas de roteadores (DD-WRT, 2007).

2.6 OPENWRT

OpenWrt é um *firmware* desenvolvido com base no núcleo do Linux para aparelhos de software embarcado, geralmente roteadores *wireless*. Diferentemente dos outros *firmwares* com esse propósito como o DD-WRT, foi construído para ser totalmente dinâmico através do gerenciador de pacotes, assim deixando livre ao usuário criar funcionalidades necessárias para seu aparelho. A versão mais recente disponibilizada é a OpenWrt Chaos Calmer 15.05.1. A

Figura 14 mostra a tela inicial dessa versão. As versões anteriores têm basicamente a mesma tela, porém com algumas modificações na descrição (OPENWRT, 2016).

Figura 14 - Tela inicial do OpenWrt



Fonte: OpenWrt (2016).

Juntamente com o OpenWrt, é disponibilizado o *toolchain*. O *toolachain* é um pacote de ferramentas que permitem fazer a compilação do OpenWrt. Com ele é possível escolher qual será o aparelho que receberá a instalação, qual será o tipo de instalação e entre outras configurações. O mais importante é que nesta seção os pacotes de funcionalidades são escolhidos para serem compilados. Esses pacotes podem ser compilados em arquivos separados para instalação posterior ou mesmo já embarcados no sistema (OPENWRT, 2016).

2.7 TRABALHOS CORRELATOS

A seção 2.7.1 descreve o Projeto de ReMesh (REMESH, 2005), trata-se de um projeto proposto e implantado para fornecer rede *mesh* comunitária. Já na seção 2.7.2, o trabalho de Gómez (2012) descreve a implantação de uma rede *mesh* de baixo custo que posteriormente foram feitos alguns testes em cima dessa rede. A seção 2.7.3 detalha a ferramenta Xirrus (2011) Wi-Fi Inspector, usada para monitorar e obter várias informações sobre a rede Wi-Fi.

2.7.1 Projeto ReMesh

Esse projeto foi desenvolvido por um grupo de pesquisa da Rede Nacional de Ensino e Pesquisa (RNP) e tem como objetivo prover uma alternativa de acesso à internet de banda larga a usuários universitários que residem próximo a Universidade Federal Fluminense e para outras diversas universidades brasileiras (REMESH, 2005). Em relação a malha de rede, nos nós Mesh foram utilizados roteadores WRT54G e WRT55G da Linksys, ambos servindo como pontos de acesso e roteadores Wi-Fi. Para aumentar a área de cobertura da rede, foram utilizadas antenas omnidirecionais que fornecem ganho mínimo de 8dB. Ainda, foram utilizados aparatos para proteção à chuva, ventos e vandalismo.

A execução do projeto se inicia na instalação do sistema OpenWRT nos roteadores Wi-Fi. O OpenWrt é um sistema operacional baseado no Linux que possui suporte aos protocolos de roteamento AODV e OLSR. Foram instalados compilando o protocolo OLSR e posteriormente o AODV. Depois foi feita a instalação preliminar no próprio campus da engenharia para realizar medições e testes em ambiente controlado. Após vários testes específicos para comprovação da eficácia da rede, finalmente foi formada a topologia adequada através da instalação dos equipamentos. E assim, a rede em malha foi formada disponibilizando acesso à Internet.

Os resultados obtidos foram satisfatórios, visto que foi a primeira experiência brasileira na construção de redes *mesh* comunitárias (REMESH, 2015). Em cima disso foram publicados vários artigos e relatórios técnicos. E para finalizar, foi comprovado que o modelo de projeto proposto funciona e pode ser utilizado em outros locais.

2.7.2 Implementação e avaliação do desempenho de Redes Wi-Mesh de baixo custo

O trabalho de Gómez (2012) tem como objetivo a criação de um projeto e implantação de uma rede *mesh*. Depois da implantação é realizada uma análise em cima da utilização da rede. Foram utilizados roteadores Linksys WRT54GL da Cisco Systems. Estes equipamentos formam dois grupos de nós necessários na rede, *mesh* e *ap* (Ponto de Acesso). Os nós “mesh” criam os caminhos e a redundância na malha de rede, e para estes foram utilizados roteados com o software Freifunk instalado, pois é o mais utilizado especificamente para redes *mesh*. Os nós “ap” servem de acesso Wi-Fi através de um IP válido aos dispositivos que querem entrar na rede. O software utilizado nos roteadores foi o DD-WRT pelo fato de permitir configurar rede *mesh* através do funcionamento do dispositivo *wireless* em modo *ad hoc*. Para que estes dois tipos de nós não se interfiram, foram configurados para funcionar em frequências distintas.

Logo após a implantação, foram feitos testes básicos para validação da rede e previsão de possíveis aplicações suportadas por ela. Os resultados da implementação foram satisfatórios, visto que a rede teve baixo custo e totalmente funcional, permitindo o aumento desta rede com a adição de mais roteadores se adaptando automaticamente e a expansão é feita facilmente.

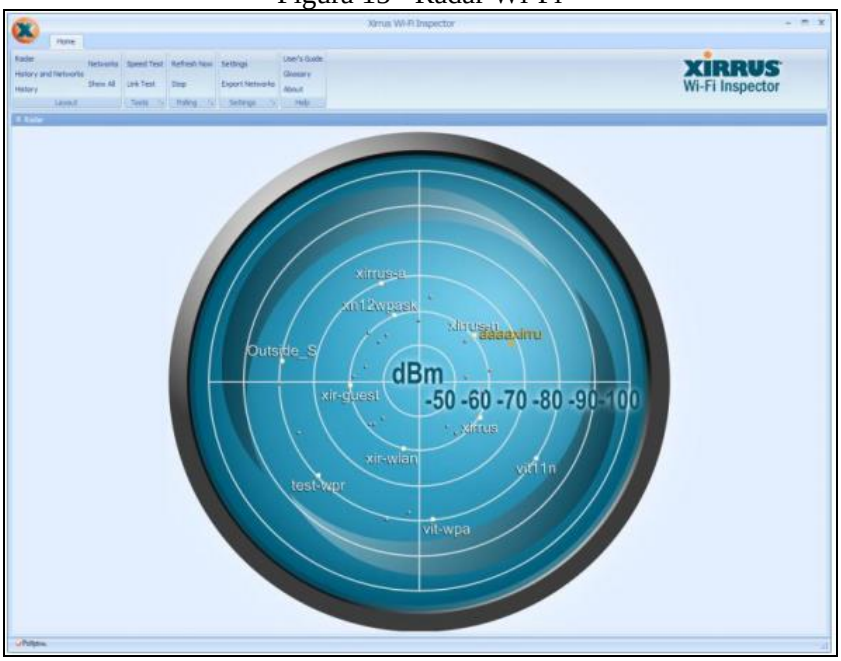
2.7.3 Xirrus Wi-Fi Inspector

Desenvolvido pela Xirrus (2011), é um aplicativo para monitoração e gerenciamento de redes Wi-Fi através de um *notebook*. Funciona em Windows 7, Vista ou XP. Ele permite

visualizar detalhes sobre a rede Wi-Fi, gerenciar a conexão e prover ferramentas para solucionar problemas de conectividade Wi-Fi. Ainda, possui funcionalidades como radar dinâmico da rede, disponibiliza detalhes sobre todas as redes Wi-Fi locais, medidor de força de sinal, exportação de redes Wi-Fi para um arquivo `.csv` e glossário completo de termos Wi-Fi.

Focando na funcionalidade de radar, ele mostra em forma de radar na tela uma visão dinâmica das conexões Wi-Fi na área local. Mostra aproximadamente a distância dos pontos de acesso juntamente com o SSID dos mesmos. Tudo isso é com relação ao local em que o usuário se encontra com o *notebook*. A Figura 15 ilustra tal radar disponível na aplicação.

Figura 15 - Radar Wi-Fi



Fonte: Xirrus (2011).

3 DESENVOLVIMENTO

As próximas seções descrevem os requisitos, a especificação, a implantação da rede e a implementação da aplicação. Ainda são apresentadas as ferramentas e bibliotecas utilizadas em cada parte da rede e da aplicação. Por último é mostrada a operacionalidade da aplicação.

3.1 REQUISITOS

A seguir são listados os Requisitos Funcionais (RF) e os Requisitos Não Funcionais (RNF):

a) a rede deve permitir acesso de um dispositivo externo (cliente/usuário) (Requisito Funcional - RF);
b) a aplicativo deve permitir ao usuário informar o nome da rede (RF);
c) a aplicativo deve permitir visualizar a topologia dos roteadores Wi-Fi na rede em tempo real (RF);
d) a aplicativo deve permitir visualizar a qualidade do sinal e o endereço MAC dos roteadores Wi-Fi (RF);
e) a aplicativo deve permitir visualizar o roteador Wi-Fi em que o usuário está conectado (RF);
f) a rede deve possuir um *gateway* já configurado com acesso à internet (RNF);
g) a rede deve utilizar 2 roteadores TP-LINK WR841-ND Hardware v9.4 (RNF);
h) a rede deve utilizar o *firmware* OpenWrt nos roteadores (RNF);
i) a rede deve utilizar o protocolo de roteamento B.A.T.M.A.N-adv (RNF);
j) a aplicativo deve utilizar a linguagem C# (RNF);
k) a aplicativo deve utilizar a biblioteca NativeWi-Fi (RNF).

O trabalho está dividido em duas partes para atender esses requisitos, são elas:

a) a rede *mesh* composta pelos dois roteadores WR841-ND;
b) a aplicação radar Wi-Fi do tipo Windows Forms, responsável por utilizar e visualizar a rede *mesh*.

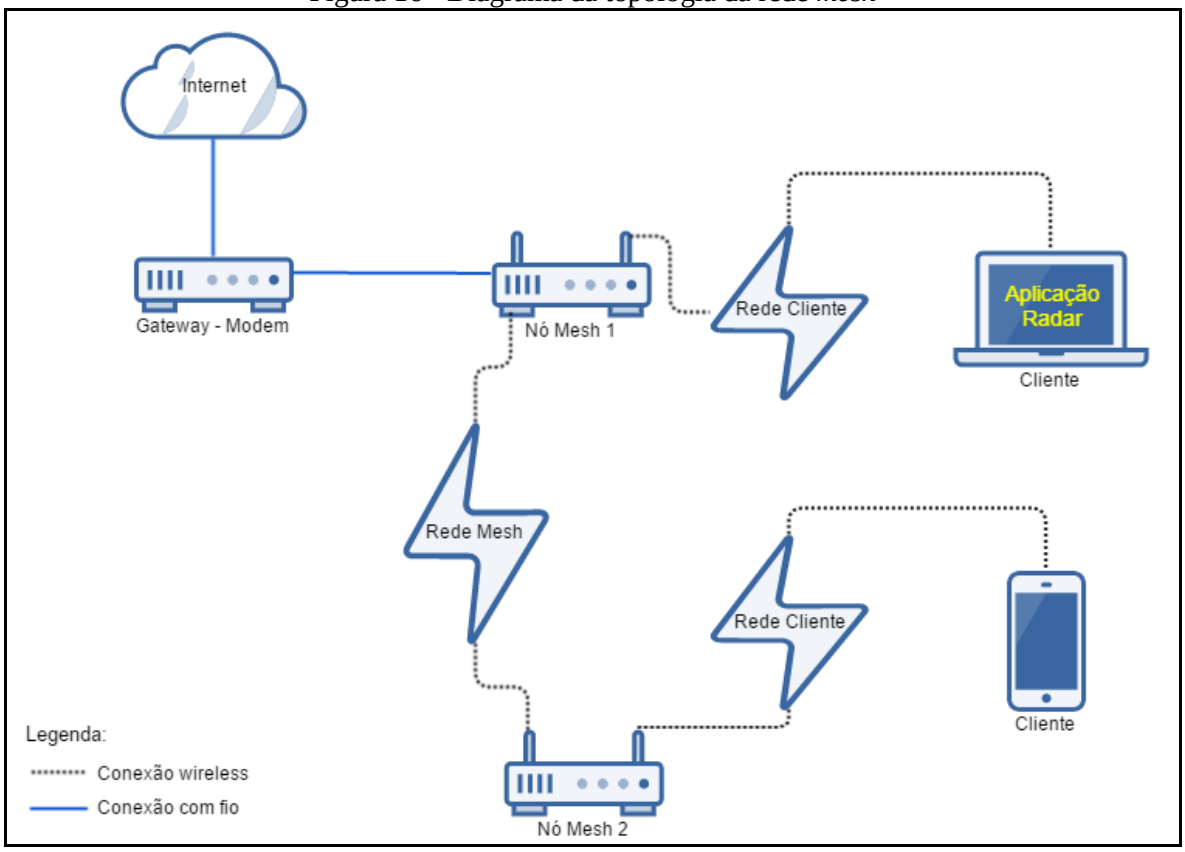
3.2 ESPECIFICAÇÃO

A especificação do trabalho foi criada usando a ferramenta Gliffy para elaboração dos diagramas de topologia da rede *mesh* e de casos de uso da aplicação radar Wi-Fi. Utilizando a ferramenta Draw.io, foi criado o diagrama de classes da aplicação.

3.2.1 Diagrama de topologia

Nesta seção é descrita a estrutura da rede implantada no trabalho. A Figura 16 demonstra a topologia da rede *mesh* assim como a comunicação dos componentes envolvidos.

Figura 16 - Diagrama da topologia da rede *mesh*

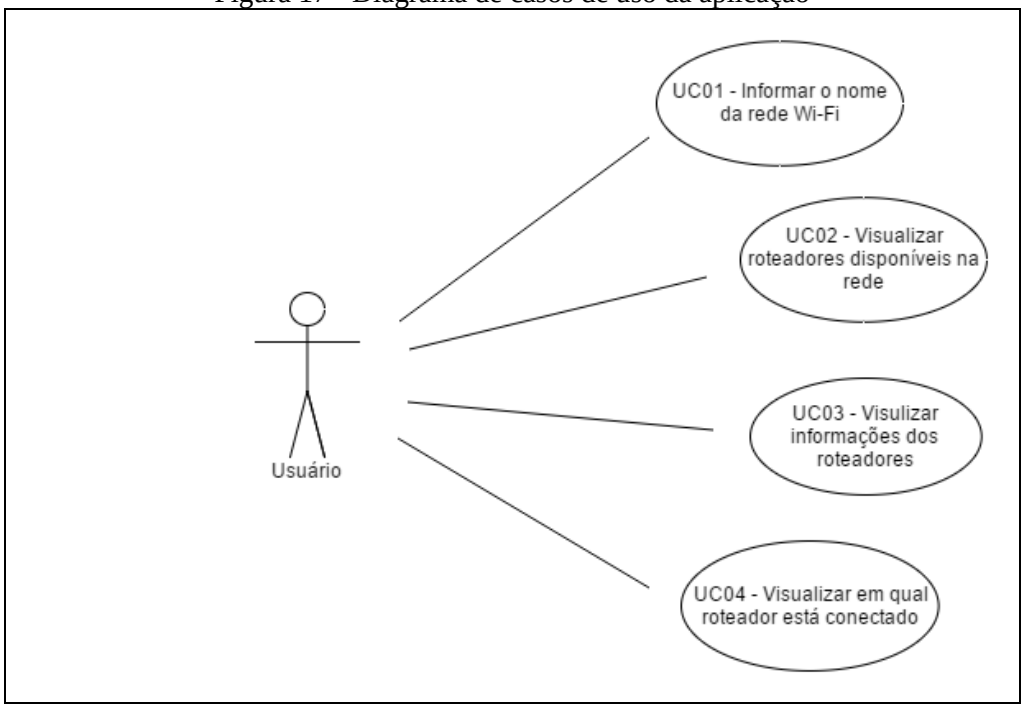


O `Nó Mesh 1` é conectado fisicamente ao `Gateway - Modem` responsável por dar acesso à `Internet`. Os dois nós Mesh formam duas redes *wireless*, a `Rede Mesh` responsável por criar a rede `mesh` que funciona em modo *ad hoc* e as `Redes Cliente` que dão acesso aos usuários (`Clientes`) e funciona no modo infraestrutura.

3.2.2 Diagrama de casos de uso

Nesta seção são detalhados os casos de uso da aplicação conforme Figura 17. A aplicação possui um cenário e um ator, sendo ele o `Usuário`.

Figura 17 - Diagrama de casos de uso da aplicação

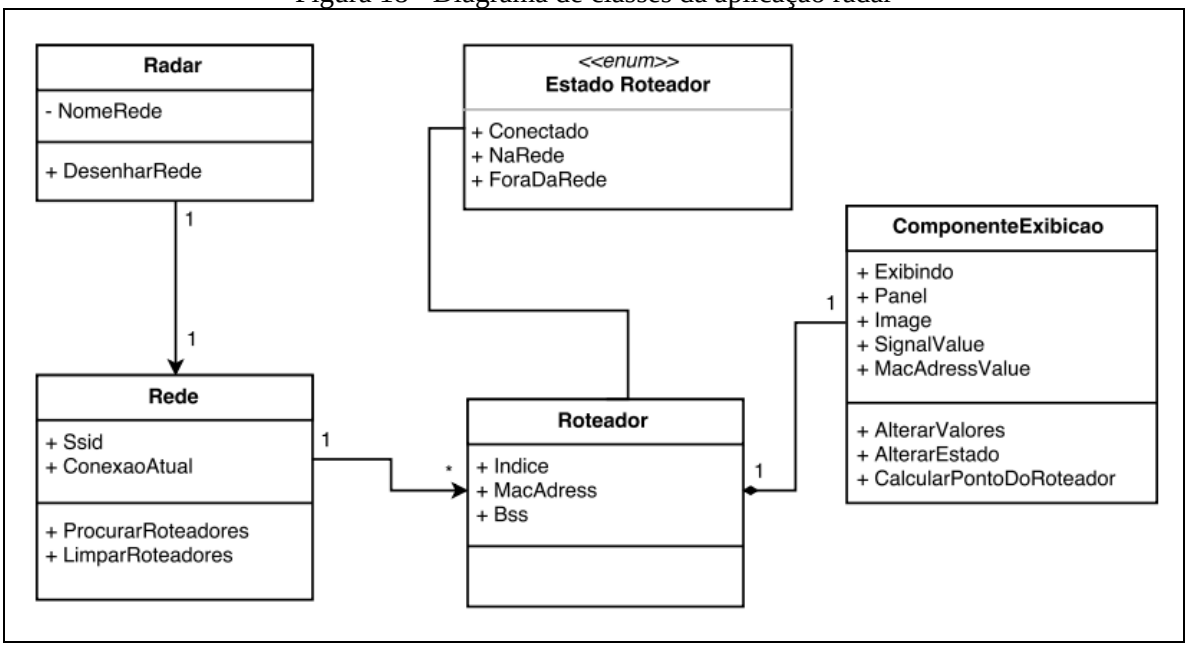


No caso de uso `UC01 – Informe o nome da rede Wi-Fi`, o `Usuário` informa o nome da rede Wi-Fi a ser exibida no radar. O caso de uso `UC02 – Visualizar roteadores disponíveis na rede` permite ao `Usuário` visualizar os roteadores disponíveis na rede. O caso de uso `UC03 – Visualizar informações dos roteadores` permite ao `Usuario` visualizar informações dos roteadores sendo exibidos na aplicação radar. O caso de uso `UC04 – Visualizar em qual roteador está conectado` permite ao `Usuário` verificar em qual dos roteadores do radar ele está conectado.

3.2.3 Diagrama de classes

Nesta seção está descrita a estrutura de classes da aplicação radar desenvolvida. A Figura 18 mostra o diagrama de classes assim como seus relacionamentos.

Figura 18 - Diagrama de classes da aplicação radar



A classe `Rede` é a principal classe da aplicação. Através dela que ocorre a busca de roteadores em uma rede Wi-Fi e a exibição destes no radar. A classe `Radar` é responsável por executar em intervalos intermitentes as funcionalidades da classe `Rede` com o propósito de manter o radar em constante atualização. A classe `Roteador` manterá informações do roteador como `Indice`, `MacAdress`, `Bss` e `Estado`. Por sua vez, a classe `ComponenteExibicao` é responsável por manter um componente de auxílio para exibição deste roteador e suas informações no radar.

3.3 IMPLEMENTAÇÃO

Para o melhor entendimento, a seção de implementação foi dividida em duas partes. Na primeira parte são descritos os processos para implantação da rede *mesh* e na segunda parte é descrita a implementação da aplicação radar Wi-Fi.

3.3.1 Rede *mesh*

Para criação da rede *mesh* foram utilizados roteadores que tiveram seu *firmware* original alterado para o *firmware* OpenWrt. Ainda, o OpenWrt utilizado foi customizado para trazer consigo o pacote de roteamento B.A.T.M.A.N-adv embarcado. Nesta seção são apresentados os processos envolvidos na compilação do OpenWrt e na configuração do protocolo B.A.T.M.A.N-adv.

3.3.1.1 Compilação do OpenWrt

Roteadores Wi-Fi comuns possuem *firmwares* originais de fábrica que não permitem a utilização de protocolos necessários em redes *mesh*, por esse motivo foi utilizado o OpenWrt. Os dois roteadores empregados neste trabalho são da TP-LINK, modelo TL-WR841ND v9.4. Possuem 4MB de memória *flash* e 32MB de memória *RAM*, configuração mínima necessária para instalação do *firmware* OpenWrt. Conforme a Figura 19 apresenta, o TL-WR841ND é um roteador Wi-Fi simples e seu preço varia entre R$ 125,00 à R$ 175,00.

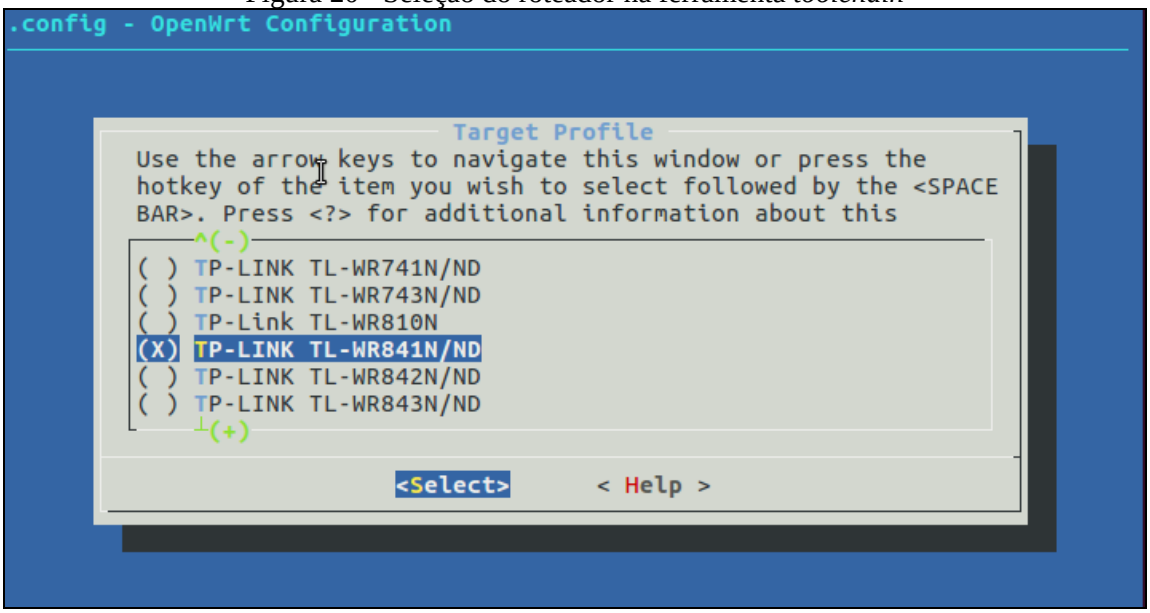
Figura 19 - Roteador TL-WR841ND

Fonte: TP-Link (2016).

Encontram-se várias versões do OpenWrt compiladas e prontas para uso no site deste roteador Wi-Fi. Juntamente podem ser compilados vários pacotes de customização, inclusive o pacote do protocolo de roteamento B.A.T.M.A.N-adv. Porém na instalação de alguns desses pacotes no *firmware* podem ocorrer problemas de compatibilidade da versão de *kernel* entre o *firmware* e o pacote. Devido a isso, a ferramenta *toolchain* é disponibilizada ao usuário. A ferramenta permite fazer a compilação customizada do OpenWrt, incluindo pacotes de forma embarcada na instalação.

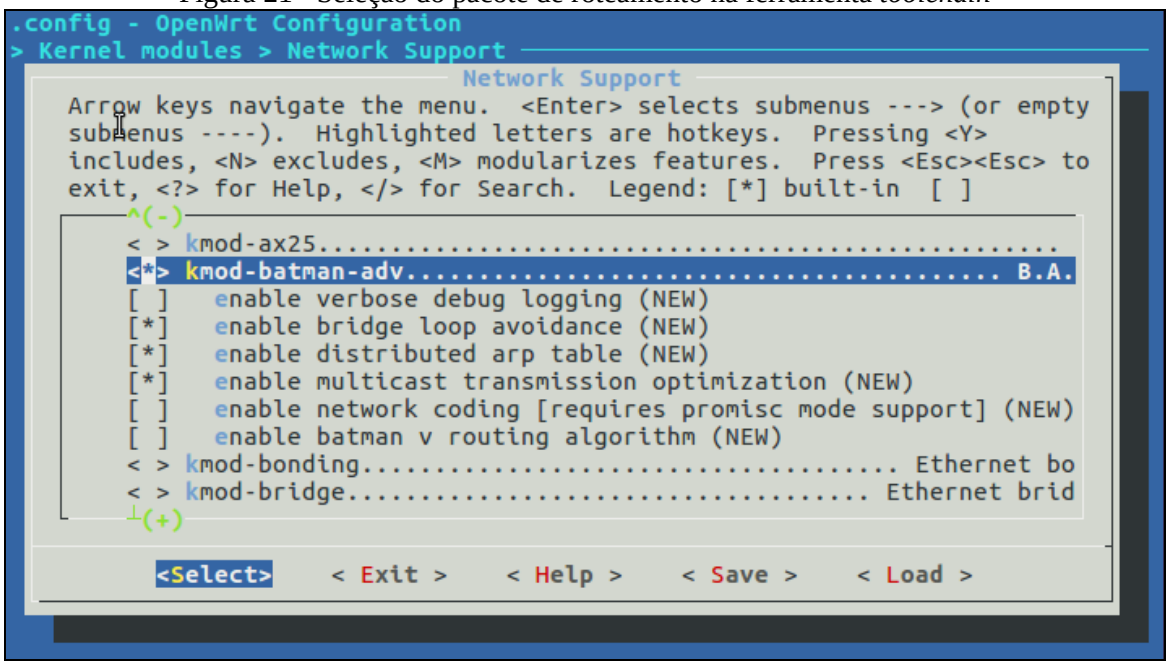
A ferramenta *tootlchain* foi instalada em um sistema operacional Ubuntu para fazer a compilação do *firmware* OpenWrt. Conforme mostra a Figura 20, no item *Target Profile* da configuração da ferramenta foi selecionado o modelo TP-LINK TL-WR841N/ND.

Figura 20 - Seleção do roteador na ferramenta *toolchain*



Com a escolha do roteador já seria possível fazer a compilação do OpenWrt. Porém ainda é necessário incluir o pacote de roteamento B.A.T.M.A.N-adv. No item *Kernel Modules* e dentro dele no item *Network Suport* encontra-se o pacote *kmod-batman-adv*, conforme demonstrado na Figura 21. Para que este pacote já venha embarcado na compilação do OpenWrt, sua seleção foi feita pressionando a tecla Y, caso contrário esse pacote seria compilado separadamente.

Figura 21 - Seleção do pacote de roteamento na ferramenta *toolchain*



Somente essas duas configurações foram suficientes para compilação do *firmware* com suporte ao B.A.T.M.AN-adv. Dentre os vários arquivos gerados na compilação, somente o

arquivo *openwrt-ar71xx-generic-tl-wr841-v9-squashfs-factory.bin* é necessário para a instalação. A instalação foi feita através interface WEB disponibilizada pelo roteador.

3.3.1.2 Configuração do B.A.T.M.A.N-adv

Após a instalação do OpenWrt nos 2 roteadores, é preciso fazer a configuração dos mesmos. Como mostra o diagrama da Figura 16, o roteador `Nó Mesh 1` foi conectado fisicamente ao `Gateway – Modem`. O roteador `Nó Mesh 2`, depois de configurado, só se conecta através do sinal *wireless* com o `Nó Mesh 1` para ter acesso à internet, formando assim a rede *mesh*.

O `Gateway – Modem` possui uma configuração padrão com acesso à internet. Além disso tem configurado um servidor *DHCP*. Esse servidor é único e responsável por disponibilizar um IP para cada cliente que irá se conectar nas redes cliente, pois se houver mais de 1 servidor *DHCP* podem ocorrer conflitos de IPs sendo disponibilizados.

O OpenWrt disponibiliza um sistema de configuração chamado *Unified Configuration Interface* (UCI). O UCI é um sistema que busca centralizar este processo mantendo todos os arquivos de configurações disponíveis em um mesmo local. Outro recurso disponibilizado pelo OpenWrt é o vi. O vi é o editor de texto padrão do *firmware* e permite que sejam feitas configurações nos arquivos do UCI. Apesar de ser funcional, não é intuitivo nos primeiros usos (OPENWRT, 2016).

A configuração do `Nó Mesh 1` implicou em modificações nos arquivos `/etc/config/network` e `/etc/config/wireless` do UCI. O arquivo `/etc/config/network` mantém as configurações de Switch, Interfaces e de Rotas do roteador. No Quadro 2 é apresentada a configuração utilizada neste arquivo para o `Nó Mesh 1`.

Quadro 2 - Arquivo `network` do `Nó Mesh 1`

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config interface 'lan'
        option type bridge
        option ifname 'eth0 bat0'
        option proto static
        option ipaddr '192.168.99.25'
        option mask '255.255.255.0'

config interface 'mesh'
        option fname 'adhoc0'
        option mtu '1528'
        option proto 'batadv'
        option mesh 'bat0'
```

Por padrão o *firmware* cria o bloco `interface 'loopback'`. Nada foi modificado neste bloco, pois não é aconselhável fazer alterações caso não for realmente necessário. Esse bloco é utilizado para configuração internas de delimitações de IPs e afins.

O segundo bloco cria uma `interface 'lan'` com o objetivo de habilitar as portas LANs do roteador. `eth0` é a interface física correspondente as portas físicas do roteador e `bat0` é a interface virtual disponibilizada pelo pacote B.A.T.M.A.N-adv. Dessa forma, além de habilitar as portas LANs, a combinação das opções `ifname 'eth0 bat0'` e `type bridge` permitem fazer a ponte de dados entre estas duas interfaces. Assim, ao engatar um cabo físico do `Modem – Gateway` em um roteador, a rede *mesh* passa a ter acesso à internet. As opções `proto static`, `ipaddr '192.168.99.25'` e `mask '255.255.255.0'` são configurações padrões para que as portas LANs tenham uma identificação na rede e possam se comunicar normalmente.

Para executar o protocolo de roteamento do pacote B.A.T.M.A.N-adv criou-se a `interface 'mesh'`. A interface física `fname 'adhoc0'` corresponde a interface do roteador que executa em modo *ad hoc*. A opção `proto 'batadv'` é responsável por informar aos scripts de execução que a interface `'adhoc0'` será parte da rede *mesh* e que o protocolo de roteamento deverá ser executado. O pacote B.A.T.M.A.N-adv disponibiliza uma interface virtual `mesh 'bat0'` que serve como um vínculo de entrada/saída com o processo de execução do protocolo. Ou seja, com estas configurações a interface `'adhoc0'` torna-se uma

interface direta para execução do roteamento através do seu vínculo com a interface `mesh 'bat0'`.

Essas configurações fazem com que tudo funcione dentro do roteador. A passagem de internet de uma porta LAN para a rede *mesh* e a execução do protocolo de roteamento já estão em funcionamento. Porém a comunicação com outros Nós Mesh e o acesso de clientes ainda não estão disponíveis. Para isso foram feitas modificações no arquivo /etc/config/wireless, como mostra o Quadro 3.

Quadro 3 - Arquivo wireless do `Nó Mesh 1`

```
config wifi-device radio0
        option type     mac80211
        option channel  11
        option hwmode   11g
        option path     'platform/qca953x_wmac'
        option htmode   HT20

config wifi-iface
        option device   radio0
        option network  lan
        option mode     ap
        option ssid     RedeCliente
        option channel  1
        optoin encryption none

config wifi-iface 'wmesh'
        option device 'radio0'
        option ifname 'adhoc0'
        option network 'mesh'
        option mode 'adhoc'
        option ssid 'RedeMesh'
        option bssid '02:CA:FE:CA:CA:40'
```

O primeiro bloco `wifi-device radio0` é criado automaticamente na primeira execução do sistema. Neste processo o OpenWrt identifica qual adaptador Wi-Fi está disponível no roteador e o configura nesta seção. As configurações `type`, `channel`, `hwmode`, `path` e `htmode` foram criadas e preenchidas automaticamente, porém caso seja necessário podem ser alteradas e até criadas novas configurações.

O segundo bloco de configuração `wifi-iface` criará uma rede Wi-Fi para acesso de clientes à rede *mesh*. Designou-se o `device radio0` como adaptador Wi-Fi a ser utilizado e feito o vínculo dessa rede com a `network lan`, configurada para liberar acesso à internet. Na criação da conexão Wi-Fi propriamente dita utilizou-se o `mode ap` para definir que este sinal

será do tipo ponto de acesso, foi dado o nome de `ssid RedeCliente` e irá rodar no canal 1. Nenhuma configuração de segurança e de encriptação foram utilizadas.

Para criação da rede Wi-Fi *mesh* foi definido o `wifi-iface 'wmesh'`, responsável por criar a comunicações entre os Nós Mesh da rede. Novamente, foi utilizado o `device 'radio0'` por ser o único adaptador Wi-Fi disponível no roteador. Definiu-se que esta conexão será acoplada à `network 'mesh'` pois ela está configurada para executar o protocolo de roteamento e ainda foi - e deve ser - utilizada a mesma interface física `ifname 'adhoc0'` dessa `network`. O modo de funcionamento utilizado foi o `mode 'adhoc'` e para criação do sinal Wi-Fi foi dado `ssid 'RedeMesh'` e `bssid '02:CA:FE:CA:CA:40'`. Por ter esse modo de funcionamento e este ssid e bssid específicos, somente aparelhos que tenham estas mesmas configurações conseguirão se comunicar.

Com essas configurações o `Nó Mesh 1` está pronto para uso. Ao conectá-lo ao `Gateway – Modem` ele já começa a servir como um nó de saída da rede *mesh* e ainda disponibiliza uma rede de acesso aos clientes. Para estender a rede será configurado o `Nó Mesh 2`, que no contexto deste trabalho irá se comunicar através da `RedeMesh` com o `Nó Mesh 1`.

Igualmente a configuração do `Nó Mesh 1`, a configuração do `Nó Mesh 2` será feita nos arquivos `/etc/config/network` e `/etc/config/wireless`. Em comparação a configuração do `Nó Mesh 1` pouco coisa será modificada. O Quadro 4 apresenta o arquivo `/etc/config/network`.

Quadro 4 - Arquivo `network` do `Nó Mesh 2`

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config interface lan
        option type bridge
        option ifname eth0
        option proto static
        option ipaddr 192.168.1.1
        option mask 255.255.255.0

config interface 'mesh'
        option fname 'adhoc0'
        option mtu '1528'
        option proto 'batadv'
        option mesh 'bat0'

config interface mb
        option ifname   bat0
        option type     bridge
        option proto    static
        option ipaddr   192.168.99.26
        option netmask  255.255.255.0
```

Enquanto no Nó Mesh 1 a `interface lan` tinha a função de servir como uma saída de dados para a internet, visto que estava ligado diretamente ao `Gateway – Modem`, aqui ela serve apenas para configuração. Essa configuração pode ser feita por exemplo através de um *notebook* conectado diretamente ao roteador.

A configuração da interface `'mesh'` é exatamente igual à do `Nó Mesh 1`, pois seu objetivo é o mesmo, apenas executar o protocolo de roteamento. No entanto foi criada uma nova interface chamada `mb`, que em partes desempenha o mesmo papel da interface `lan` do `Nó Mesh 1`. A configuração `ifname bat0` juntamente com `type bridge` resulta em uma ponte de conexão com a interface de saída liberada pelo protocolo B.A.T.M.A.N-adv. Ou seja, ao chegar dados nessa interface com um cliente acessando uma página da internet por exemplo, o protocolo identificará e será processado, passando ou não adiante (`Nó Mesh 1`). Ao retornar essa informação, essa mesma interface `bat0` irá disponibilizar esse dado que será repassado, ao cliente neste exemplo, através da conexão Wi-Fi. As configurações `ipaddr` e `netmask` servem para identificação e comunicação dessa interface.

Novamente, depois das configurações de `/etc/config/network` é feita a configuração de `/etc/config/wireless`. Como pode ser observado no Quadro 5, quase nada mudou em relação à configuração do Nó Mesh 1. As únicas mudanças foram a criação do `network mb` para ser a interface com o objetivo de servir como ponte com a interface `bat0` e a utilização do canal 6 para evitar conflitos entre as redes cliente.

Quadro 5 - Arquivo wireless do `Nó Mesh 2`

```
config wifi-device radio0
        option type     mac80211
        option channel  11
        option hwmode   11g
        option path     'platform/qca953x_wmac'
        option htmode   HT20

config wifi-iface
        option device   radio0
        option network  mb
        option mode     ap
        option ssid     RedeCliente
        option channel  6
        optoin encryption none

config wifi-iface 'wmesh'
        option device 'radio0'
        option ifname 'adhoc0'
        option network 'mesh'
        option mode 'adhoc'
        option ssid 'RedeMesh'
        option bssid '02:CA:FE:CA:CA:40'
```

Com estas configurações, a rede *mesh* e a rede cliente funcionam conforme desejado. Caso for necessário, é possível adicionar mais nós utilizando exatamente a mesma configuração do `Nó Mesh 2`, com atenção ao canal em que a rede cliente irá rodar, para que não haja conflitos.

3.3.2 Radar Wi-Fi

Nesta seção são descritos detalhadamente as técnicas e ferramentas utilizadas e a operacionalidade da implementação da aplicação radar Wi-Fi.

3.3.2.1 Técnicas e ferramentas utilizadas

Para desenvolvimento da aplicação radar Wi-Fi utilizou-se a linguagem C# da plataforma Microsoft .NET. Na busca de informações dos roteadores de uma rede Wi-Fi foi

utilizada a biblioteca NativeWi-Fi disponibilizada pela Microsoft. Para permitir a exibição do radar foi utilizada uma aplicação do tipo Windows Forms.

3.3.2.2 Implementação da aplicação

Nesta seção é apresentada de forma detalhada como é feita a busca dos roteadores na rede e depois como é realizada a apresentação destes roteadores no radar com suas devidas informações.

3.3.2.2.1 Busca dos roteadores

A classe `Rede` contém os processos de busca dos roteadores de uma rede. Para isso, em seu construtor é criado um objeto do tipo `WlanClient` que permite a busca de informações da rede Wi-Fi. Ainda, conforme mostra o Quadro 6, no construtor é definida a interface Wi-Fi utilizada no computador e criada uma lista de `Roteadores`.

Quadro 6 - Construtor da classe `Rede`

```
1. public Rede()
2. {
3.        wLanClient = new WlanClient();

4.        ConexaoAtual = wLanClient.Interfaces[0];

5.        Roteadores = new List<Roteador>();
6. }
```

O Quadro 7 apresenta o método `BuscarBssExato` responsável por buscar e retornar uma lista das entradas bss disponíveis na rede informada pelo usuário. Essas entradas bss são as redes criadas por cada roteador dentro da rede geral.

Quadro 7 - Método `BuscarBssExato` da classe `Rede`

```
1. public List<Wlan.WlanBssEntry> BuscarBssExato()
2. {
3.       Wlan.WlanBssEntry[] wlanBssEntries =
               ConexaoAtual.GetNetworkBssList();
4.
5.       return (from wlanBssEntry in wlanBssEntries
6.             where BuscarStringDoSsid(wlanBssEntry.dot11Ssid) ==
                                                                SSID
7.             select wlanBssEntry).ToList<Wlan.WlanBssEntry>();
8. }
```

Além de só buscar os roteadores disponíveis na rede, é preciso buscar algumas informações de cada um necessárias na exibição do radar. Por isso o método `ProcurarRoteadores` foi criado. Conforme mostra o Quadro 8, sua finalidade é trazer essa lista de roteadores, guardá-los em objetos do tipo `Roteador` e depois fazer a coleta de algumas informações. A classe `Roteador` possui as seguintes propriedades:

a) `Indice`: com a finalidade de alinhar os roteadores no radar;

b) `MacAdress`: endereço único de cada aparelho;

c) `Bss`: a rede criada pelo roteador;

d) `Componente`: objeto responsável por exibir o roteador no radar;

e) `Estado`: estado da conexão do roteador em relação a rede – usuário.

O método inicia com a busca da lista de bss disponíveis. Logo após é criada uma cópia da lista de roteadores já conhecidos da rede, ou seja, em alguma execução anterior já foram descobertos e armazenados. Então a lista de bss é percorrida e caso o roteador já seja conhecido, a informação `Bss` é atualizada e ele é removido da lista cópia por ainda estar disponível na rede. Caso ainda não for conhecido, é criado um novo objeto `Roteador` para ele e adicionado na lista de `Roteadores`. Ainda nesse laço é feita a verificação do `Estado` do `Roteador` na rede, onde ele pode ser o roteador em que o usuário está conectado no momento ou ser apenas um roteador ativo na rede. Após percorrer esse laço por todos os roteadores ativos, sobraram apenas os roteadores que não estão mais visíveis na rede, ou seja, todos os roteadores da lista `roteadoresParaRemover` tem seu `Estado` alterado para fora da rede.

Quadro 8 - Método `ProcurarRoteadores`

```
1. public void ProcurarRoteadores()
2. {
3.       var listaBss = this.BuscarBssExato();

4.       List<Roteador> roteadoresParaRemover =
               Roteadores.ToList<Roteador>();

5.       foreach (Wlan.WlanBssEntry bss in listaBss)
6.       {
7.             var roteador =
               roteadoresParaRemover.FirstOrDefault<Roteador>(x =>
               x.Bss.dot11Bssid.SequenceEqual(bss.dot11Bssid));

8.             if (roteador != null)
9.             {
10.                  roteador.Bss = bss;
11.                  roteadoresParaRemover.Remove(roteador);
12.            }
13.            else
14.            {
15.                  roteador =
                           new Roteador(Roteadores.Count, bss);
16.                  Roteadores.Add(roteador);
17.            }

18.            roteador.Estado =
         VerificarSeEhRoteadorConectado(roteador.Bss.dot11Bssid)
         ? EstadoRoteador.CONECTADO : EstadoRoteador.NAREDE;
19.      }

20.      foreach (Roteador roteadorParaRemover in
               roteadoresParaRemover)
21.      {
22.            roteadorParaRemover.Estado =
                     EstadoRoteador.FORADAREDE;
23.      }
24.      }
```

O `Estado` do `Roteador` é representado pelo `enum EstadoRoteador`. São 3 estados possíveis:

a) `CONECTADO`: é o roteador atual em que o usuário está conectado à rede;

b) `NAREDE`: o roteador está na rede ao alcance do usuário;

c) `FORADAREDE`: o roteador já foi reconhecido anteriormente na rede, porém no momento está fora do alcance do usuário.

3.3.2.2.2 Exibição dos roteadores no radar

Para a exibição do radar utilizou-se um `Form` em uma aplicação do tipo `Windows Forms`. A ideia geral do radar é mostrar no `Form` todos os roteadores da rede um abaixo do outro e com uma distância horizontal relativa ao ponto em que o usuário está localizado. Com

intuito de facilitar essa exibição no `Form` foi criada a classe `ComponenteExibicao`. Esse componente possui 5 propriedades relacionadas a exibição do roteador, são elas:

a) `Exibindo`: valor booleano indicando se roteador está sendo exibido no radar no momento;
b) `Panel`: painel responsável por organizar as informações exibidas;
c) `Image`: imagem indicando o estado do roteador;
d) `SignalValue`: rótulo indicando a potência do sinal;
e) `MacAdressVale`: rótulo indicando o endereço MAC do roteador.

Como mostra o trecho de código do Quadro 9, na criação do componente essas propriedades são preenchidas e o ponto em que o `Panel` deve ser exibido em tela é calculado. Esse cálculo é feito através do método `CalcularPontoDoRoteador` utilizando as informações de potência do sinal e o índice do roteador. Com a valor do sinal é definida a distância horizontal em que o roteador será posicionado em relação à posição do usuário, ou seja, quanto maior o sinal mais perto do usuário será posicionado. Com o índice é calculada a posição vertical em relação aos outros roteadores, para que cada roteador fique sempre um abaixo do outro.

Quadro 9 - Trechos de código do construtor da classe `ComponenteExibicao`

```
1. public ComponenteExibicao(int indice, int qualidadeDeSinal,
                                                    string macAdress)
2. {
3.       Exibindo = false;

4.       Panel = new FlowLayoutPanel();
5.       Panel.Location =
               CalcularPontoDoRoteador(qualidadeDeSinal, indice);
6.       ...
7.       SignalValue.Text =
               string.Format("Sinal: {0} %", qualidadeDeSinal);
8.       ...
9.       MacAdressValue = new Label();
10.      MacAdressValue.Text = macAdress;
11.      ...
12.      Image = new PictureBox();
13.      Image.BackgroundImage =
               global::RadarWifi.Properties.Resources.router;
14.      ...
15.      }
```

Depois de criado, ainda é possível alterar essas propriedades através dos métodos `AlteraEstado` e `AlterarValores`. O método `AlterarEstado` recebe um parâmetro do tipo `EstadoRoteador` e com base nele altera a `Image` para um dos 3 estados possíveis. O método `AlterarValores` é responsável por alterar o rótulo `SignalValue` com o novo valor de sinal e alterar o local do `Panel` também de acordo com o novo valor sinal do roteador, utilizando o

mesmo cálculo do construtor. No Quadro 10 são dispostas as assinaturas com os respectivos parâmetros destes 2 métodos.

Quadro 10 - Assinatura dos métodos `AlterarEstado` e `AlterarValores`

```
1. public void AlterarEstado(EstadoRoteador estado)

2. public void AlterarValores(int qualidadeSinal, int indice)
```

O radar necessita, em intervalos constantes, realizar a busca dos roteadores e atualizar as informações na tela. Para isso, como mostra o Quadro 11, criou-se um objeto do tipo `Radar` e um objeto do tipo `System.Windows.Forms.Timer`, responsável por executar o radar a cada 0.5 segundos.

Quadro 11 - Parte responsável por criar o `Radar` no `Form`

```
1. radar = new Radar(this);

2. timer.Tick += new EventHandler(ExecutarTimer);
3. // 0.5 segundos
4. timer.Interval = 500;
```

No método `ExcutarTimer` é feita a buscar dos roteadores e logo após a impressão deles na tela. Essas duas execuções são feitas através dos métodos `VerificarRede` e `DesenharRede` da classe `Radar`, conforme demonstrado no trecho de código do Quadro 12.

Quadro 12 - Método responsável pela execução do radar

```
1. private static void ExecutarTimer(Object myObject,
                                     EventArgs myEventArgs)
2. {
3.       radar.VerificarRede();
4.       radar.DesenharRede();
5. }
```

O Quadro 14 apresenta o método `VerificarRede`, responsável por fazer a busca dos roteadores disponíveis na rede e logo após retornar um valor binário verificando se existe algum roteador disponível. Essa busca é feita através do método `ProcurarRotedores` da classe `Rede`, deixando-os disponíveis no objeto `rede`. Nesta busca, caso o roteador já tenha sido descoberto anteriormente, a informação `bss` do objeto `Roteador` será alterada conforme mostra o Quadro 8. Assim que essa informação é alterada, o método `AlterarValores` do `ComponenteExibicao` é disparado para que sejam feitas as devidas alterações no roteador, de acordo com o trecho de código do Quadro 13.

Quadro 13 - Trecho de código que altera informações do roteador ao mudar de `bss`

```
1. private Wlan.WlanBssEntry bss;
2. public Wlan.WlanBssEntry Bss
3. {
4.       get { return bss; }
5.       set
6.       {
7.       bss = value;
8.       ComponenteExibicao.AlterarValores(
                    (int)bss.linkQuality, Indice);
9.       }
10.      }
```

O método `DesenharRede` irá percorrer os roteadores disponíveis na rede e para cada um que ainda não está sendo exibido no radar, criará um `ComponenteExibicao` para ele. Após sua a criação, o componente é adicionado no `Form` conforme mostra o Quadro 14.

Quadro 14 - Métodos `VerificarRede` e `DesenharRede` da classe `Radar`

```
1. public bool VerificarRede()
2. {
3.       rede.ProcurarRoteadores();

4.       return rede.Roteadores.Count > 0;
5. }

6. public void DesenharRede()
7. {
8.       foreach (Roteador roteador in rede.Roteadores)
9.       {
10.            if (!roteador.Componente.Exibindo)
11.            {
12.                  Form.Controls.Add(roteador.Componente.Panel);
13.                  roteador.Componente.Exibindo = true;
14.            }
15.      }
16.      }
```

3.3.2.3 Operacionalidade da implementação

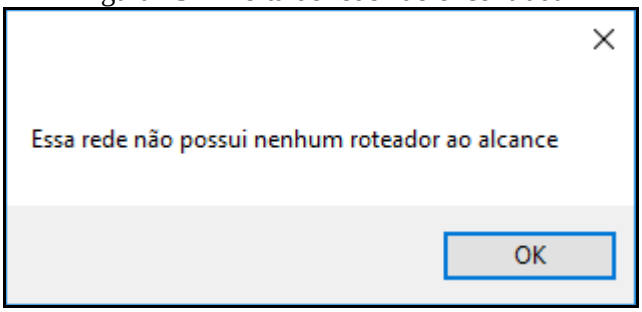
Nesta seção serão apresentadas as funcionalidades estipuladas nos casos de usos da seção 3.2.2. A aplicação possui somente duas etapas simples, resumidas em informar o nome da rede e posteriormente visualizar o radar mapeando a rede informada.

Conforme mostra a Figura 22, inicialmente o usuário deve informar o nome da rede e pressionar o botão "*Procurar"* para que o radar comece a executar. O nome deve ser informado exatamente igual ao da rede configurada, pois caso trocar uma letra maiúscula para minúscula ou vice-versa, o radar não consiguiria encontrar a rede.

Figura 22 - Tela inicial do radar

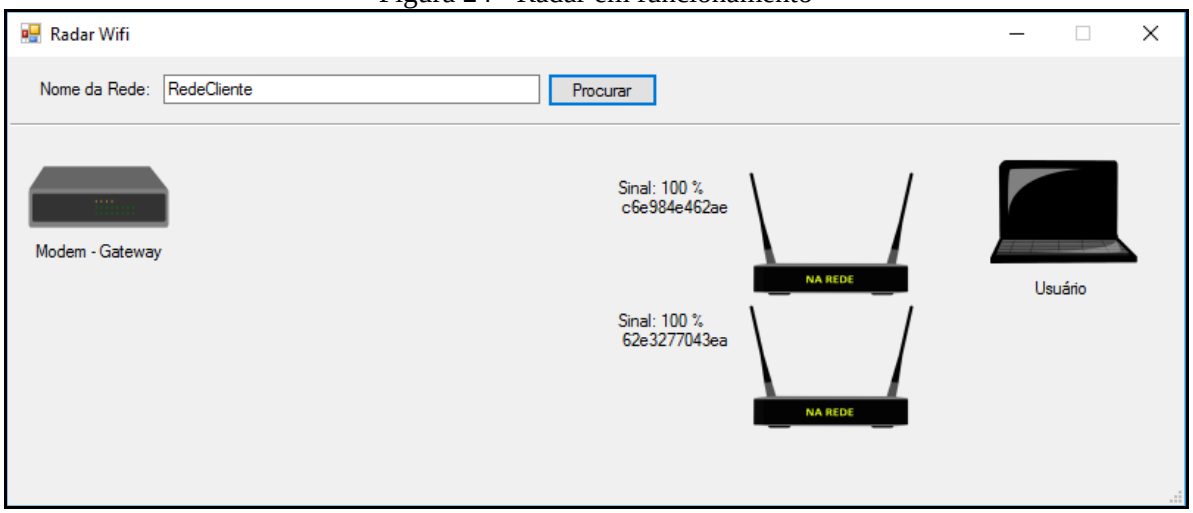
Caso não seja encontrado roteador na rede, é exibida uma mensagem conforme mostra a Figura 23. O usuário poderá informar novamente um novo nome de rede para buscar novamente ou sair do radar.

Figura 23 - Alerta de rede não encontrada



Se algum roteador for encontrado, o radar exibirá todos os roteadores da rede que estiverem ao seu alcance do usuário, conforme mostra a Figura 24.

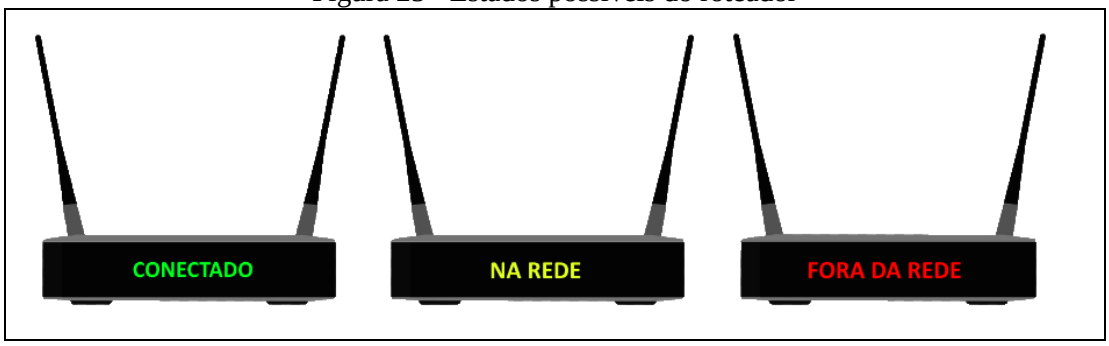
Figura 24 - Radar em funcionamento



O radar mostra um roteador abaixo do outro no eixo vertical. Em relação ao eixo horizontal o roteador é posicionado de acordo com potência de sinal que o usuário consegue

captar. Ou seja, quanto maior o sinal atribuído ao roteador, mais perto do usuário ele será posicionado. A informação de sinal e o endereço MAC do roteador são mostradas ao lado esquerdo da sua imagem. O estado do roteador é apresentado no *display* da imagem, podendo ser um dos três disponíveis conforme mostra a Figura 25.

Figura 25 - Estados possíveis do roteador



Com o radar executando, conforme o usuário se locomove com seu *notebook*, a potência de sinal dos roteadores poderá ser alterada. Além disso, o estado do roteador também pode ser alterado se o usuário se locomover a uma distância suficiente para perder contato com o roteador. Neste caso o roteador tem estado alterado para Fora da Rede. O inverso também pode acontecer, se o usuário alcançou um roteador que estava Fora da Rede, este roteador passa para o estado Na Rede novamente. Além disso o roteador pode ter seu estado alterado caso aconteça alguma queda de energia ou avaria em geral no aparelho que o faça desligar.

O outro estado possível para o roteador é de CONECTADO Se a rede informada no radar for a mesma rede em que o usuário está conectado, então este roteador terá o estado CONECTADO. Haverá somente um roteador com este estado, caso contrário não haverá nenhum. Para que este estado seja alterado, o usuário deve se locomover a uma distância onde a potência de sinal do roteador conectado seja aproximadamente abaixo de 50% e ainda encontre outro roteador que tenha uma potência acima deste valor. Nestas condições o roteador CONECTADO passa para Na Rede e este novo roteador assume o estado de CONECTADO.

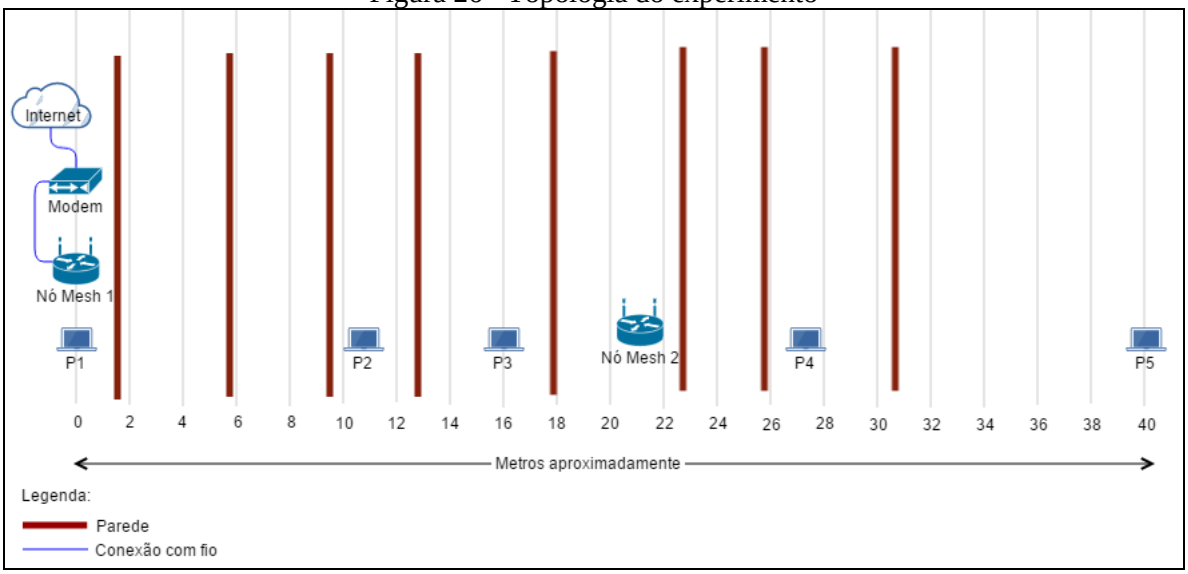
3.4 RESULTADOS E DISCUSSÕES

Nesta seção é apresentado o experimento feito para demonstrar o funcionamento da rede *mesh* utilizando o radar Wi-Fi. Logo após é feita a comparação dos trabalhos correlatos com o trabalho desenvolvido.

3.4.1 Experimento do trabalho

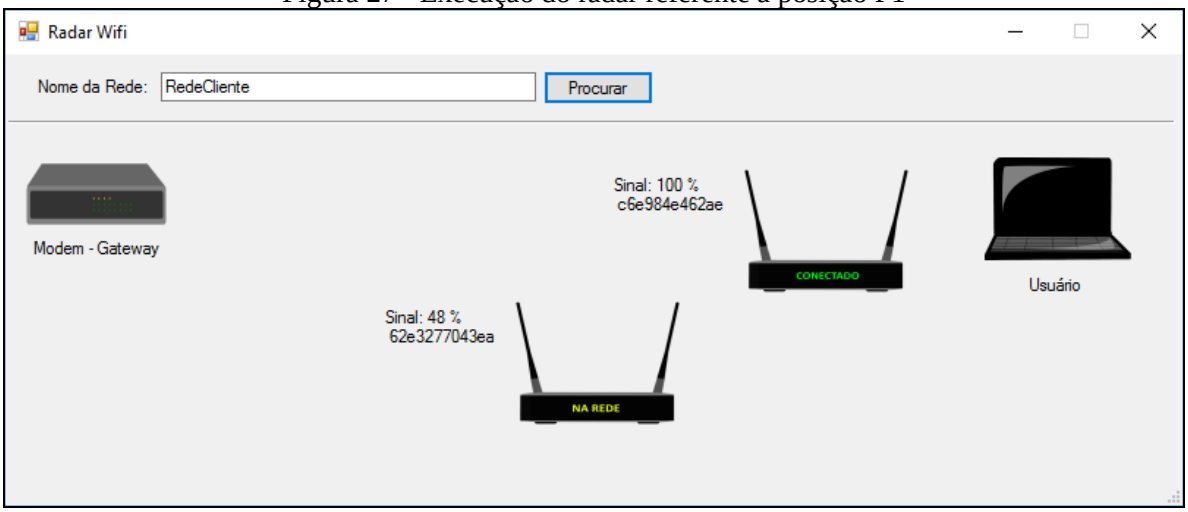
O experimento feito com a rede *mesh* e o radar Wi-Fi é baseado em uma situação de uso real onde o usuário se locomove nas extremidades da rede *mesh*. O cenário montado é composto pelos 2 roteadores Mesh configurados no trabalho. Estes roteadores estão a uma distância aproximada de 21 metros e são responsáveis por formar a rede *mesh* e a rede cliente. O `Nó Mesh 1` está conectado diretamente a um modem configurado com internet e como servidor *DHCP*. A Figura 26 mostra em forma de topologia como este cenário foi disposto no ambiente. Nessa topologia foram adicionadas as medidas aproximadas em metros e as paredes que formam uma barreira ao sinal *wireless*. Além disso, foram adicionadas posições (P1 à P5) percorridas pelo usuário na execução do experimento.

Figura 26 - Topologia do experimento



No experimento foi utilizado um *notebook* conectado à rede "RedeCliente" e logo após feita a execução da aplicação radar Wi-Fi. No radar o nome da rede foi informado e então foi dado o comando de "*Pesquisar*". Neste momento o experimento de locomover-se através dos roteadores da rede *mesh* foi iniciado. Inicialmente o usuário estava localizado na posição P1. Nesta posição o radar estava mostrando a rede conforme apresenta a Figura 27.

Figura 27 - Execução do radar referente a posição P1



Vale ressaltar que o `Nó Mesh 1` é representado pelo primeiro roteador exibido com endereço MAC c6e984e462ae e o `Nó Mesh 2` pelo segundo com endereço MAC 62e32777043ea. Devido ao usuário estar praticamente junto ao `Nó Mesh 1`, este é o roteador que ele está conectado na rede, com sinal 100%. A Figura 28 mostra a execução do comando *ping* na linha de comando provando o acesso à Internet, ou seja, o `Nó Mesh 1` está funcionando perfeitamente.

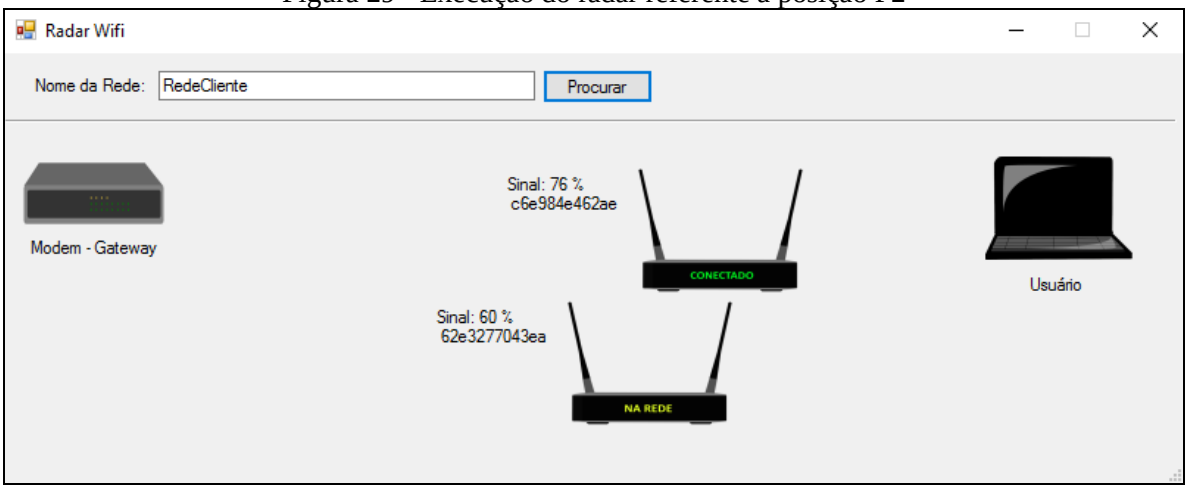
Figura 28 - Execução do comando ping na posição P1

```
Disparando www.google.com [74.125.196.103] com 32 bytes de dados:
Resposta de 74.125.196.103: bytes=32 tempo=246ms TTL=45
Resposta de 74.125.196.103: bytes=32 tempo=195ms TTL=45
Resposta de 74.125.196.103: bytes=32 tempo=160ms TTL=45
Resposta de 74.125.196.103: bytes=32 tempo=235ms TTL=45

Estatísticas do Ping para 74.125.196.103:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de
             perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 160ms, Máximo = 246ms, Média = 209ms
```

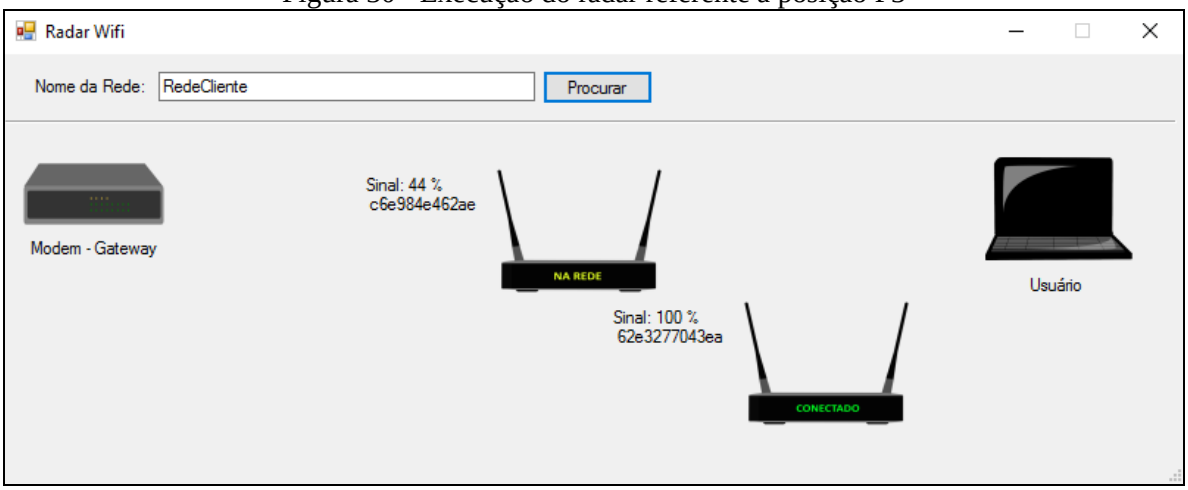
O usuário se locomoveu até a posição P2 com o radar em execução. A Figura 29 apresenta a leitura do radar nesta posição.

Figura 29 - Execução do radar referente a posição P2



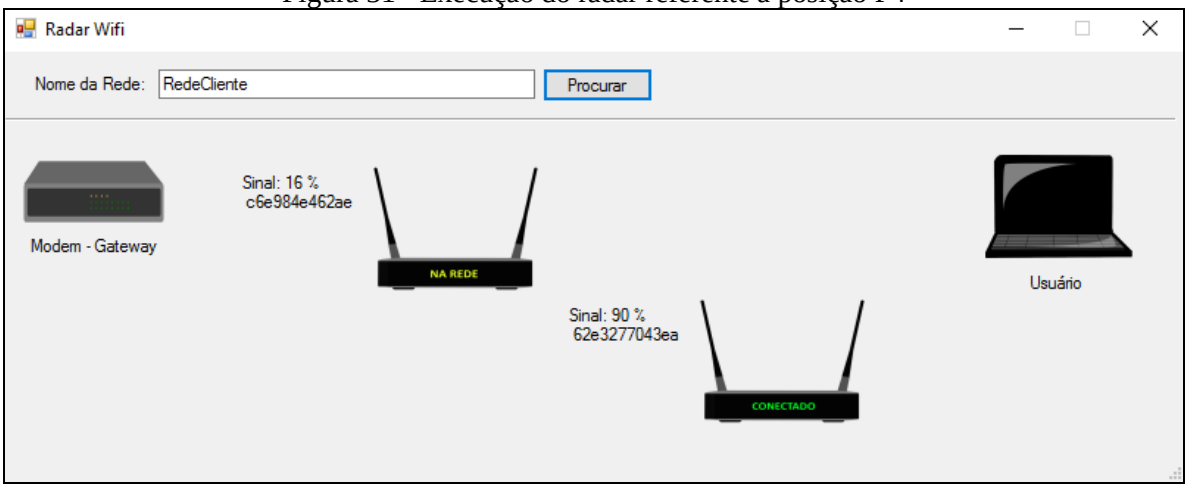
As únicas mudanças em relação a P1 foram as potências de sinal de cada roteador, pois o usuário afastou-se do `Nó Mesh 1` e aproximou-se do `Nó Mesh 2`. Com o usuário localizado na posição P3, o radar mostrou as conexões conforme apresenta a Figura 30.

Figura 30 - Execução do radar referente a posição P3



Na posição P3 o usuário foi automaticamente conectado no roteador `Nó Mesh 1`. Esse processo é feito pelo Windows e aconteceu pois a conexão com o `Nó Mesh 1` estava muito fraca em relação a potência de sinal do `Nó Mesh 2`. Na posição P4, o radar apresentou a rede apresentada na Figura 31. Novamente, somente os valores de potência de sinal sofreram alterações em relação à P3.

Figura 31 - Execução do radar referente a posição P4



Com o usuário localizado na posição P5 aconteceram várias mudanças significativas. Conforme pode ser visto na Figura 32, o `Nó Mesh 1` não está mais ao alcance da aplicação radar, pois está à uma distância de 40 metros em média e com 8 paredes dificultando o sinal *wireless*. Porém, mesmo nessa distância ainda há acesso à internet, como demonstra a execução do comando *ping* na Figura 33. Novamente, esse acesso à Internet prova que a rede *mesh* está funcionado perfeitamente.

Figura 32 - Execução do radar referente a posição P5

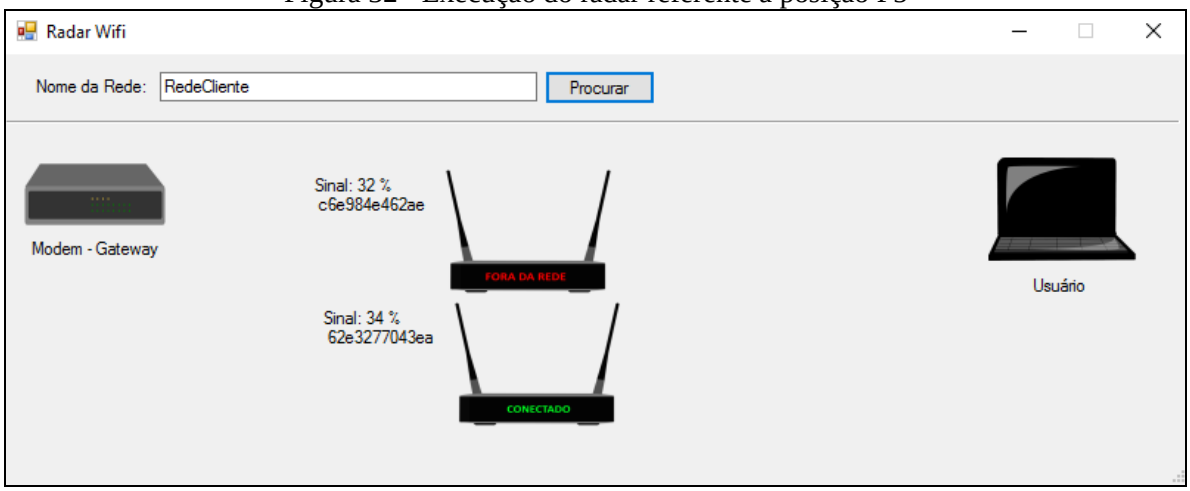


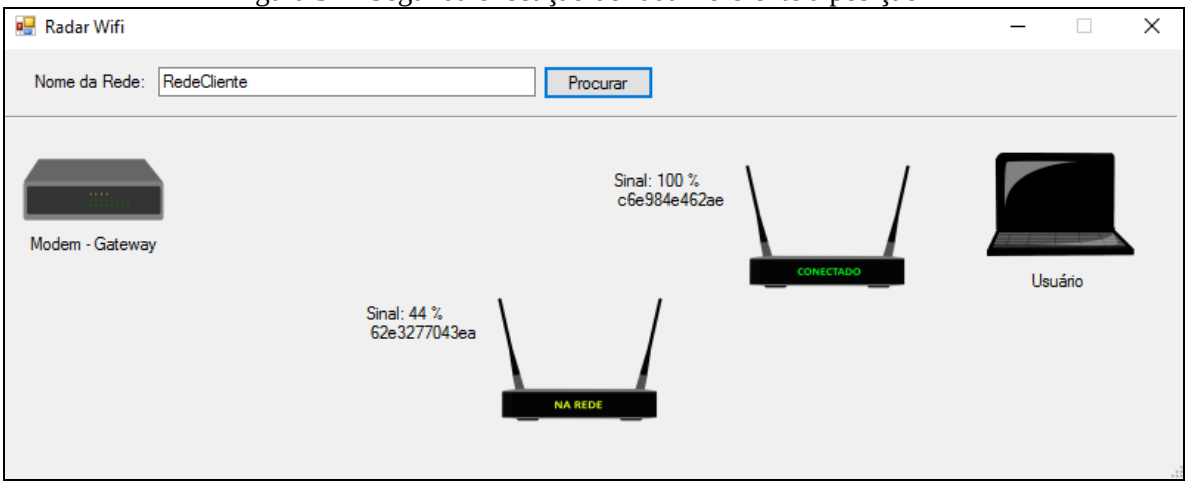
Figura 33 - Execução do comando ping na posição P5

```
Disparando www.google.com [216.58.208.100] com 32 bytes de dados:
Resposta de 216.58.208.100: bytes=32 tempo=306ms TTL=54
Resposta de 216.58.208.100: bytes=32 tempo=304ms TTL=54
Resposta de 216.58.208.100: bytes=32 tempo=293ms TTL=54
Resposta de 216.58.208.100: bytes=32 tempo=297ms TTL=54

Estatísticas do Ping para 216.58.208.100:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de
             perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 293ms, Máximo = 306ms, Média = 300ms
```

Para finalizar o experimento, o usuário moveu-se até a posição P1. A Figura 34 mostra que o `Nó Mesh 1` voltou a ser ativo pois tem potência de sinal bem maior em relação à potência do `Nós Mesh 2`.

Figura 34 - Segunda execução do radar referente a posição P1



A execução deste experimento mostrou que a rede *mesh* configurada funcionou sem interrupções. Porém no radar Wi-Fi constatou-se um pouco de atraso, em média 10 segundos, para refletir as alterações da rede quando o usuário se locomovia. Investigando mais a fundo, notou-se que esse atraso ocorre na biblioteca NativeWiFi utilizada pela aplicação radar.

3.4.2 Comparação com os trabalhos correlatos

A comparação dos trabalhos correlatos com o trabalho desenvolvido é esquematicamente separada entre comparação com a rede *mesh* e com o radar Wi-Fi. O projeto Remesh (2005) e o trabalho de Gómez (2012) fazem a criação de uma rede *mesh*, logo foram comparados à rede *mesh* do trabalho. O Quadro 15 mostra de forma comparativa algumas características entre os trabalhos correlatos que implantam uma rede *mesh* e o trabalho presente. O aplicativo Xirrus (2011) é um radar Wi-Fi, então foi comparado ao aplicativo radar Wi-Fi do trabalho.

Quadro 15 – Características dos trabalhos correlatos relacionados à rede *mesh* e o presente

Característica / trabalho	Remesh (2005)	Gómez (2012)	Trabalho Presente
roteador utilizado	WRT54G e WRT55AG	WRT54GL	TL-WR841ND
Protocolo utilizado	AODV e OLSR	OLSR	B.A.T.M.A.N
firmware utilizado	OpenWrt	Freifunk e DD-WRT	OpenWrt

O projeto Remesh (2005) utiliza roteadores Linksys modelos WRT54G e WRT55AG que possuem maior poder de processamento e de memória em relação aos utilizados neste trabalho. Entretanto utilizaram o mesmo *firmware*, o OpenWrt, que na época era possível somente ser instalado em roteadores mais potentes. Além disso utilizaram os protocolos AODV e OLSR, que exigem mais processamento e memória do roteador por não terem otimização se comparados ao protocolo B.A.T.M.A.N. O objetivo do projeto Remesh é construir uma rede *mesh* completa com otimização de banda e com sistema de segurança e permissão bem definidos. Neste trabalho esses conceitos não estavam projetados, pois o objetivo foi mostrar como criar a rede *mesh* de forma simples e ainda criar o aplicativo radar Wi-Fi.

Em relação ao trabalho de Gómez (2012), o roteador utilizado foi o Linksys WRT54GL, igualmente utilizado no projeto Remesh. Porém a concepção deles foi feita de forma diferenciada. Foram criados 2 tipos de roteadores. Um tipo tem o objetivo de formar o *backbone* da rede *mesh*, então foi instalado o *firmware* Freifunk e nele configurado o protocolo OLSR. O outro foi instalado o *firmware* DD-WRT pois seu objetivo é servir somente como ponto de acesso ao cliente. Essa é a grande diferença entre o trabalho desenvolvido, que centraliza essas duas funcionalidades no mesmo roteador. O trabalho de Gómez (2012) tem foco também em construir uma rede *mesh* com otimização de velocidade e alto desempenho.

O aplicativo Xirrus (2011) oferece um radar Wi-Fi com muitas funcionalidades. Diferente do objetivo deste trabalho, o Xirrus tem ferramentas de análise avançada de redes Wi-Fi e de fornecer correção de erros mais comuns encontrados na rede. Além disso, o radar do Xirrus mostra todos os roteadores de todas as redes disponíveis na área com mais eficiência na precisão de local dos roteadores, mais precisão na distância dos mesmos. Enquanto o aplicativo desenvolvido neste trabalho apresenta somente os roteadores de uma rede específica e a distância em que se encontram.

4 CONCLUSÕES

O principal objetivo deste trabalho que era desenvolver um radar Wi-Fi para análise da implantação de uma rede *mesh* foi alcançado. Porém, para chegar no resultado alcançado ocorreram várias mudanças em relação aos meios utilizados para alcançar tais objetivos.

Inicialmente iria-se utilizar o *firmware* DD-WRT nos roteadores da rede. Com isso comprou-se dois roteadores D-Link através da Internet, visto que há mais variedades tanto em preço quanto em aparelhos. Entretanto a maioria das lojas online não informam qual a versão do hardware do roteador, somente o modelo e a marca. Vale ressaltar que um modelo de roteador pode ter várias versões de hardware. Notou-se então a importância dessa informação, pois não foi possível a instalação do DD-WRT para a versão do hardware do primeiro roteador comprado. Essa é uma característica deste tipo de *firmware*, para cada modelo e cada versão de hardware de um aparelho existe uma instalação específica para ele. Sendo assim, pode acontecer que para uma versão de hardware de um modelo não haja instalação, mesmo que para todas as outras versões de hardware tenham.

Sabendo desse empecilho, foram comprados outros 2 roteadores TP-LINK WR840ND versão de hardware 9.4 em loja física na cidade, pois desta forma foi possível verificar a versão do hardware com exatidão. Com eles em mãos, foi instalado o *firmware* DD-WRT e funcionou corretamente. Inicialmente se iria utilizar o protocolo de roteamento OLSR, pois nas pesquisas feitas mostrava que o DD-WRT teria suporte a este protocolo. Porém, depois de instalado não foi possível achar o protocolo OLSR para configuração no *firmware*.Com base em informações mais detalhadas, descobriu-se que para ter suporte ao protocolo OLSR o roteador deve possuir mais poder de processamento e mais memória RAM do que o roteador escolhido. E tudo isso porque o OLSR é um protocolo pesado e robusto. Devido a isso, foram feitas novas pesquisas e encontrou-se o protocolo de roteamento B.A.T.M.A.N, mais novo, enxuto e simples.

Como o *firmware* OpenWrt é quem dá o melhor suporte ao protocolo B.A.T.M.A.N-adv, foi decidido utilizar ele no lugar do DD-WRT. Dessa forma foi feita a instalação do OpenWrt nos roteadores. A instalação padrão disponibilizada não trazia consigo o pacote de roteamento B.A.T.M.A.N-adv, porém esse pacote era disponibilizado para instalar separadamente. Na tentativa de instalar tal pacote separadamente, sempre ocorria erro de incompatibilidade de *kernel* entre a versão de *build* do *firmware* e a versão de *build* do pacote. Tentou-se com várias outras compilações do pacote, porém todas sem sucesso.

Devido a facilidade de compilação do OpenWrt, decidiu-se fazer sua compilação. Neste processo vários problemas ocorreram em várias tentativas de compilação. Uma lição aprendida é que para compilação do OpenWrt é preciso bastante espaço de disco na pasta do usuário no Linux, em torno de 10 GBs. Caso faltar espaço, ocorrem vários erros no meio da compilação, porém erros aleatórios que não apontam para a falha real que é a falta de espaço em disco. Com bastante espaço disponível, a execução da compilação ocorreu com sucesso. Essa compilação já foi feita com o pacote B.A.T.M.A.N-adv embarcado. Com o OpenWrt trazendo embarcado o pacote B.A.T.M.A.N-adv não ocorreu erro de *kernel* e sua instalação nos roteadores aconteceu com sucesso.

Contornadas todas estas casualidades, o que se percebeu que mais dificultou na implantação da rede *mesh* foi a falta de informações destes detalhes. Com eles resolvidos, configurar a rede *mesh* foi uma tarefa relativamente fácil de ser compreendida e executada.

Quanto à aplicação radar Wi-Fi, tudo ocorreu conforme previsto. A biblioteca NativeWiFi é intuitiva de ser utilizada e mostrou-se bem adequada para o propósito da aplicação. Porém a única limitação encontrada é que ocorrem pequenos atrasos em refletir as mudanças ocorridas na rede, ou seja, não é possível verificar as mudanças das conexões da rede Wi-Fi instantaneamente com essa biblioteca.

O resultado final é de grande valia para comunidade que procura conhecer sobre as redes *mesh*. O trabalho apresenta de forma prática o que é preciso e como montar uma rede *mesh* sem avançar em campos mais complexos de redes como otimização, segurança e afins.

4.1 EXTENSÕES

Algumas das possíveis extensões para este trabalho são:

a) permitir visualizar no aplicativo radar o percurso do pacote de dados através dos nós Mesh: atualmente só é possível visualizar os roteadores da rede. Para execução de operações avançadas no protocolo B.A.T.M.A.N é disponibilizado o pacote batctl. Uma destas operações é a visualização das rotas disponíveis na rede, ou seja, assim é possível visualizar o caminho que um pacote irá percorrer de um nó Mesh até o nó Mesh Gateway;
b) disponibilizar as informações da rede em tempo real no aplicativo radar: atualmente alterações nas conexões da rede Wi-Fi sofrem atrasos para serem refletidas no radar, devido ao uso da biblioteca NativeWiFi. Utilizando outra biblioteca mais avançada ou alguma outra linguagem mais robusta talvez seja possível fazer essa busca de informações de forma mais instantânea;

c) implementar sistema de segurança na rede mesh: no momento a rede mesh não possui senha e permite o acesso livre a qualquer aparelho, deixando-a vulnerável. Utilizando os recursos de configuração do OpenWrt é possível fazer várias configurações de segurança como senha, limitação de IP entre outros;
d) realizar testes e melhorias de performance na rede mesh: atualmente não foi medida a performance desempenhada e nem mesmo implementado algum tipo de aumento na performance da rede mesh. Tem disponível configurações mais avançadas para o protocolo B.A.T.M.A.N que permitem melhorar os índices de performances na rede mesh.

REFERÊNCIAS

AKYILDIZ, Ian F.; WANG, Xudong; WANG, Weilin. Wireless mesh networks: a survey. **Science Direct**, Usa, v. 47, n. 7, p. 445-487. Acesso em: 01 jul. 2016.

ARRUDA, F. W. **Estudo da tecnologia, do desenvolvimento e da utilização das redes mesh**. 2010. 50 f. Trabalho de Conclusão de Curso (Curso de Engenharia de Telecomunicações) – Centro de Ciências Tecnológicas, Universidade Regional de Blumenau, Blumenau.

ANTHONY, Sebastian. **What is mesh networking, and why Apple's adoption in iOS 7 could change the world**. [S.I.], 2014. Disponível em: <http://www.extremetech.com/computing/179066-what-is-mesh-networking-and-why-apples-adoption-in-ios-7-could-change-the-world>. Acesso em: 01 jul. 2016.

ANTUNES, Rothschild Alencastro. **Instalação de uma rede mesh metropolitana utilizando o padrão IEEE 802.11a e implementação do serviço VoIP (wman-VoIP)**. 2012. 93 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Automação, Faculdade de Engenharia do Campus de Ilha Solteira, Ilha Solteira, 2012.

BATMAN. **Protocol Concept.** [S.I.], [2006]. Disponível em: <https://www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept>. Acesso em: 01 jul. 2016.

BATMAN-adv. **B.A.T.M.A.N. advanced.** [S.I.], [2006]. Disponível em: < https://www.open-mesh.org/projects/batman-adv/wiki/Wiki>. Acesso em: 01 jul. 2016.

CLAUSEN, Thomas Heide; JACQUET, Philippe. **Optimized link state routing protocol (OLSR)**. Le Chesnay: The Internet Society, 2003. 75 p. Disponível em: <http://www.rfc-base.org/rfc-3626.html>. Acesso em: 01 jul. 2016.

DD-WRT. **Unleash you router.** [S.I.], [2007?]. Disponível em: < http://www.dd-wrt.com/>. Acesso em: 01 jul. 2016.

FERMINO, Gunnar Ramos. **Roteamento baseado em caminhos de maior grau de conectividade aplicado ao OLSR**. 2009. 141 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2009.

FILIPPI, Primavera de. **It's time to take mesh networks seriously (and not just for the reasons you think)**. [S.I.], 2014. Disponível em: <http://www.wired.com/2014/01/its-time-to-take-mesh-networks-seriously-and-not-just-for-the-reasons-you-think/>. Acesso em: 01 jul. 2016.

GÓMEZ, Carles Blanco. **Implementação e avaliação do desempenho de redes wi-mesh de baixo custo**. 2012. 48 f. Monografia (Especialização) - Curso de Engenharia de Telecomunicações, Universidade de Fortaleza, Fortaleza, 2012.

IEEE. **Advancing technology for humanity.** [S.I.], [2015?]. Disponível em: <http://www.ieee.org/index.html>. Acesso em: 01 jul. 2016.

JENKINS, Steve. **Difference between DD-WRT BrainSlayer, Eko, Fractal, and Kong Builds**. 2013. Disponível em: <http://www.stevejenkins.com/blog/2013/08/difference-between-dd-wrt-brainslayer-eko-fractal-and-kong-builds/>. Acesso em: 01 jul. 2016.

KUROSE, James F.; ROSS, Keith W.. **Rede de computadores e a internet:** Uma abordagem Top-Down. 5. ed. São Paulo: Pearson, 2010. 614 p.

MIGUEL, Alvaro Garcia de. **Estudo das redes mesh e sua aplicação na telemedicina**. 2011. 102 f. TCC (Graduação) - Curso de Engenharia de Telecomunicações, Universidade de Fortaleza, Fortaleza, 2011.

MIKROTIK: **Testwiki/advanced MikroTik wireless networks**. [S.l.], 2010. Disponível em: < http://wiki.mikrotik.com/wiki/Testwiki/Advanced_MikroTik_Wireless_networks>. Acesso em: 01 jul. 2016.

MORAES, Ana Luiza D.; XAUD, Arthur F. dos Santos; XAUD, Marco F. dos Santos. **Redes ad hoc**: Protocolos DSR, AODV, OLSR, DSDV. 2009. Disponível em: <http://www.gta.ufrj.br/grad/09_1/versao-final/adhoc/index.html>. Acesso em: 01 jul. 2016.

OGMV2. **Originator message version.** [S.I.], [2006]. Disponível em: <https://www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept>. Acesso em: 01 jul. 2016.

OPENWRT. **Wireless freedom.** [S.I.], [2016]. Disponível em: <https://wiki.openwrt.org/doc/start>. Acesso em: 01 jul. 2016.

REMESH. **Gt rede mesh**. Niterói, 2005. Disponível em: <https://iframe-memoria.rnp.br/pd/gts2005-2006/mesh.html>. Acesso em: 01 jul. 2016.

RIBEIRO, Anna Verônica Fernandes. **Uso de redes mesh como solução para o canal de retorno da tv digital interativa**. 2007. 117 f. Dissertação (Mestrado) - Curso de Pós-graduação “stricto Sensu” em Engenharia de Telecomunicações, Universidade Federal Fluminense, Rio de Janeiro, 2007.

ROYER, E.m.; TOH, Chai-keong. A review of current routing protocols for ad hoc mobile wireless networks. **Ieee Pers. Commun.**, [s.l.], v. 6, n. 2, p.46-55, abr. 1999. Institute of Electrical & Electronics Engineers (IEEE). http://dx.doi.org/10.1109/98.760423. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=760423>. Acesso em: 01 jul. 2016.

STALLINGS, William. **Data and computer communications.** 8. ed. Saddle Rive: Pearson Prentice Hall, 2007. 878 p.

TANENBAUM, Andrew S. **Rede de computadores**. Tradução Vandenberg D. de Souza. 4. ed. Rio de Janeiro: Elsevier, 2003.

TAVARES, Everardo de Lucena. **Sistema de comunicações operacionais multimídia e comunicações móveis (rede mesh 802.11s).** 2008. 126 f. Monografia (Especialização) - Curso de Gestão da Segurança da Informação e Comunicações, Ciência da Computação, Universidade de Brasília, Brasília, 2008.

TP-LINK. **The reliable choice.** [S.I.], [2016]. Disponível em: <http://www.tp-link.com.br/>. Acesso em: 01 jul. 2016.

VASSEUR, Jean-philippe; DUNKELS, Adam. **Interconnecting smart ibjects with IP:** The Next Internet. Massachusetts: Elsevier, 2010. 432 p.

WNDW (Org.). **Redes sem fio no mundo em desenvolvimento:** Um guia prático para o planejamento e a construção de uma infra-estrutura de telecomunicações. 2. ed. [S.I.]: Hacker Friendly, 2007. 397 p. Disponível em: <http://wndw.net/pdf/wndw-pt/wndw-pt-ebook.pdf>. Acesso em: 01 jul. 2016.

XIRRUS. **Xirrus wi-fi inspector**. Thousand Oaks, 2011. Disponível em: <http://wvw.xirrus.com/wifi-inspector>. Acesso em: 01 jul. 2016.