INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

CRISTIANO MICHEL NUNES

REDES MESH COM PROTOCOLO OLSR – INFRAESTRUTURA SEM FIO ESCALÁVEL, RESILIENTE E AUTOCONFIGURÁVEL PARA CONEXÃO DE DISPOSITIVOS IoTs

Prof. Dr. João Mário Lopes Brezolin

Passo Fundo

2019

CRISTIANO MICHEL NUNES

REDES MESH COM PROTOCOLO OLSR – INFRAESTRUTURA SEM FIO ESCALÁVEL, RESILIENTE E AUTOCONFIGURÁVEL PARA CONEXÃO DE DISPOSITIVOS IoTs

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Prof. Dr. João Mário Lopes Brezolin

Passo Fundo

2019

CRISTIANO MICHEL NUNES

REDES MESH COM PROTOCOLO OLSR- INFRAESTRUTURA SEM FIO ESCALÁVEL, RESILIENTE E AUTOCONFIGURÁVEL PARA CONEXÃO DE DISPOSITIVOS IoTs

Trabalho de Conclusão de Curso aprovado em 25/06/2019 como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Prof. Dr. João Mário Lopes Brezolin

_____

Prof. Me. José Antônio Oliveira de Figueiredo

_____

Prof. Me. Lisandro lemos Machado

_____

Prof. Me. Rafael Marisco Bertei

PASSO FUNDO

2019

Aos meus pais,
pela compreensão e o estímulo
em todos os momentos.

AGRADECIMENTOS

Primeiramente gostaria de agradecer, muito especialmente aos meus pais, José Itassir Nunes e Salete Maria Possan Nunes, pela ajuda e incentivo incondicional, fator determinante para esta conquista, aos meus irmãos e irmãs, Carlos Eduardo, Fabrício e Natália pelo apoio nessa caminhada e a minha irmã Marina, revisora dos meus textos.

Também quero prestar meus agradecimentos aos servidores do IFSul, aos professores do curso superior de Tecnologia em Sistemas para Internet, e em especial ao meu orientador, Prof. Dr. João Mário Lopes Brezolin pelo auxílio e compartilhamento de conhecimento para o êxito deste trabalho.

Igualmente dar o meu obrigado a todos que de alguma forma contribuíram para este triunfo, colegas de curso, amigos e familiares, e a amiga Natália Bortholacci por me informar da lista de aprovados do vestibular

"Quando [a conexão] wireless *for perfeitamente aplicada,*
a Terra inteira será convertida em um enorme cérebro,
o que na verdade é, sendo que todas as coisas são
partículas de um conjunto real e rítmico."
(Nicola Tesla)

RESUMO

O presente trabalho valida o uso do protocolo OLSR (Optmized Link State Routing Protocol) para criar uma MANET (*Mobile Ad hoc Network*) e assim fornecer uma infraestrutura de segunda camada sem fio para conexão de dispositivos IoTs(Internet of Things). Essa infraestrutura, uma rede WMN (Wireless Mesh Network), autoconfigurável, escalável, resiliente, multiplataforma e autoconfigurável.

Palavras-chaves: OLSR; Mesh; IoT. MANET.

ABSTRACT

The present paper validates the use of the OLSR (Optmized Link State Routing Protocol) to create a Mobile Ad Hoc Network (MANET) and thus provide a second layer wireless infrastructure for connection of Internet of Things (IoTs) . This infrastructure, a WMN (Wireless Mesh Network), self-configuring, scalable, resilient, cross-platform and self-configuring network.

Key words: OLSR. Mesh. IoT. MANET.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AP	Access Point
ARM	Acorn RISC Machine
DHCP	Dynamic Host Control Protocol
EAP	Extensible Authentication Protocol
ESSID	Extend Service Set ID
GLP	Gás Liquefeito de Petróleo
HNA	Host and Network Association
IETF	Internet Engineering Task Force
IFSUL	Instituto Federal Sul-rio-grandense
INRIA	Institut National de Recherche en Informatique et en Automatique
IoT	*Internet of Things*
IP	Internet Protocol
IPv4	Inernet Protocolo versão 4
IPv6	Internet Procolo versão 6
ISM	Industrial Sientific and Medical
KB	Kilo Byte
LAN	Local Area Network
MANET	Mobile Ad hoc Network
MB	Mega Byte
Mbps	Mega bit por segundo
MID	*Multi Interface Declaration*
MPR	Multi Point Relay
NAT	Network Address Translation
SNAT	Source Network Address Translation
DNAT	Destination Network Address Translation
OLSR	Optmized Link State Routing Protocol

QoS	Quality of Service
RAM	Read Only Memory
RFC	*Request For Change*
SO	Sistema Operacional
SSH	Secure Shell
SSID	Service Set Idenfier
TCC	Trabalho de Conclusão de Curso
TCP	Transport Control Protocol
USB	Universal Serial Bus
WAN	Wide Area Network
WiFi	Wireless Fidelity
WPA	WiFi Protected Access
WLAM	*Wireless Local Area Network*
WMN	Wireless Mesh Network

Sumário

1 INTRODUÇÃO

O presente trabalho procura avaliar o uso do protocolo OLSR (*Optmized Link State Routing Protocol*) para qualificar a comunicação de dispositivos IoT (*Internet of Things*) em redes *mesh*, objetivando oferecer mais rotas de acesso a estes dispositivos em uma infraestrutura escalável. A utilização de tabelas dinâmicas de rotas, gerenciadas pelo protocolo OLSR permitem instaurar uma comunicação mais eficiente, e também proporcionar uma infraestrutura sem fio resiliente, escalável, autoconfigurável e multiplataforma. Dispositivos de comunicação sem fio estão sujeitos a interferências em suas faixas de frequência devido a outros equipamentos que operam na mesma faixa de frequência ou a mecanismos que geram ondas eletromagnéticas capazes de interferir no sinal transmitido ou recebido. Além das interferências eletromagnéticas há ainda obstáculos físicos que podem atenuar a onda propagada para envio de dados ou na recepção de dados. A distância entre o dispositivo e o *gateway* IoT também pode causar perda de comunicação, já que dispositivos IoT, geralmente, são projetados para baixo consumo de energia, consequentemente possuem menor potência de transmissão de dados, assim como, velocidades de transmissão.

Geralmente utiliza-se apenas um "*gateway*" IoT, um dispositivo que funciona semelhante a um roteador *wireless* que concentra toda comunicação entre os dispositivos conectados. Porém, ter apenas um ponto para conexão dificulta a ampliação da área de cobertura de atuação destes dispositivos, além disso os dispositivos mais distantes geograficamente são mais suscetíveis a interferências e consequentemente a perda de comunicação.

Conforme o site do Governo Federal (BRASIL, 2017), o Brasil possui aproximadamente 20 milhões de dispositivos da Internet das Coisas (*Internet of Things*) e a previsão para 2020 é que este número saltará para algo em torno de 42 milhões. No mundo, a quantidade de dispositivos deverá ficar entre 100 e 200 milhões até o ano de 2025. Estes dispositivos estão cada vez mais presentes e estão possibilitando ligar o mundo físico a Internet ou a uma outra rede, causando impactos na sociedade e na economia em praticamente todos os setores.

Particularmente por se tratar de uma tecnologia com um espectro de utilização amplo, devido ao baixo consumo de energia, é possível ter vários cenários para a aplicação da mesma. Hoje é possível rastrear bens que estão sendo transportados, monitorar batimentos cardíacos durante a prática de exercícios físicos, controlar o ambiente, etc. Em outras palavras, os dispositivos IoT estão alterando a forma como as pessoas interagem com o meio a sua volta. Na indústria, por exemplo, onde o aumento de processos automatizados requerem um maior controle e gerenciamento principalmente, em áreas críticas ou de perigo,

o acesso aos sensores de monitoramento deve ser eficaz e confiável. Sensores WiFi são fáceis de implantar devido a sua mobilidade e podem funcionar de forma autônoma. Além da indústria, as residencias também tendem a fazer uso de dispositivos IoT. Dispositivos como sensores de gás GLP (gás liquefeito de petróleo) e temperatura podem ser efetivos na prevenção de desastres.

Com o constante aumento do número de dispositivos IoT sendo inseridos no Brasil e no mundo anualmente, cabe trazer a discussão de como melhor implementar e utilizar este tipo de tecnologia. Esta nova tecnologia já é uma realidade e a produção acadêmica é um meio que possibilita compreender melhor sua implementação e utilização, e, sendo assim, também ser capaz de produzir conhecimento para o seu avanço ou melhorar a forma como é empregada. Dado este contexto, quanto maior a produção de pesquisas, mais a sociedade pode se beneficiar desta tecnologia. Nesse sentido, o presente trabalho busca avaliar as vantagens e implicações da implementação de uma rede de *mesh* que utiliza tabelas de roteamento baseadas no protocolo OLSR para prover uma infraestrutura sem fio para as conexões de dispositivos da Internet das Coisas, infraestrutura esta, escalável, resiliente e autoconfigurável.

1.1 MOTIVAÇÃO

A difusão das tecnologias IoT demandam a implementação de recursos que permitam a agregação destes dispositivos ao contexto do cenário atual, onde, cada vez mais surgem equipamentos da Internet das Coisas. Com o aumento de opções de dispositivos IoTs sendo lançados no mercado, surge a necessidade de prover uma infraestrutura de conexão adequada.

Um simples roteador *wireless* que antes supria as necessidades de conexões sem fio para diversos dispositivos em ambientes domésticos e profissionais, hoje pode não mais suprir todas as necessidades, pois tem sua atuação limitada ao alcance de cobertura da potência dos rádios transmissores/receptores propagada por sua antena. Além disso, as opções de ampliação de cobertura de sinal e, atualmente, as técnicas e tecnologias de ampliação da área de cobertura não apresentam bons resultado. Dessa forma, estabelece-se a necessidade de se buscar novas tecnologias para prover uma infraestrutura de comunicação sem fio para este novo cenário que se apresenta.

1.2 OBJETIVOS

Neste trabalho demostraremos que uma rede *mesh* gerenciada pelo protocolo OLSR, oferece uma infraestrutura de rede escalável, resiliente, autoconfigurável e multiplataforma para prover uma infraestrutura sem fio para conexões de dispositivos IoTs.

1.2.1 Objetivo Geral

Avaliar as vantagens e desvantagens da implementação de uma rede *mesh* utilizando o protocolo OLSR para estabelecimento de múltiplas e melhores rotas entre os nós de redes presentes.

1.2.2 Objetivos Específicos

- Conceituar redes *mesh* e o protocolo OLSR.
- Validar a solução proposta por meio de testes verificando o funcionamento do protocolo de roteamento OLSR.

1.3 ORGANIZAÇÃO DO TRABALHO

Para melhor compreensão, o presente trabalho está organizado em capítulos, onde o Capítulo 2 aborda os conceitos das principais tecnologias utilizadas para a construção da solução proposta neste trabalho. O capítulo inicia com uma breve revisão acerca da rede *wireless* e redes *mesh*, assim como, o protocolo OLSR, a principal tecnologia empregada para esta solução, e as ferramentas de *software* e *hardware*.

No Capítulo 3 segue a descrição da implementação da rede *mesh* e o protocolo OLSR para o gerenciamento de rotas, sua arquitetura e configuração, quais os *hardwares* utilizados para formarem os nós da rede e a configuração dos *softwares* para a criação da MANET.

Já o Capítulo 4 aborda todos os testes realizado para validar a proposta deste trabalho, assim como, as configurações e alterações necessárias para construir a infraestrutura sem fio, uma rede *mesh*, para prover uma rede em segunda camada escalável, resiliente e autoconfigurável para a conexão de dispositivos IoTs.

Por fim, o Capítulo 5 apresenta as considerações finais deste trabalho, que envolve as tecnologias de redes *mesh*, procolo OLSR e dispositivos da Internet das Coisas.

2 REFERENCIAL TEÓRICO

Para melhor compreensão das tecnologias abordadas neste trabalho, o referencial teórico será dividido em seções, onde inicialmente será apresentada uma breve revisão sobre redes Wireless e Redes Mesh. Após, uma exposição sobre o protocolo OLSR. Seguido por uma breve descrição dos dispositivos de hardware e software utilizados. Acompanhado de um levantamento sobre as ferramentas tecnológicas utilizadas para neste trabalho. E por fim, trabalhos acadêmicos relacionados com o tema deste trabalho.

2.1 REDE WIRELESS

A comunicação sem fio ocorre através de ondas de rádio nas quais rádios transmissores/receptores operam na mesma faixa de frequência e utilizam um mesmo regramento de comunicação, ou seja, utilizam um protocolo de comunicação em comum. Essas redes também são conhecidas como redes Redes WiFi (marca registrada pela Wi-Fi Aliance) que se tornou sinônimo para redes de computadores sem fio. As redes sem fio geralmente têm muita flexibilidade, o que pode se traduzir em implantação rápida (GAST, 2002).

A vantagem da comunicação *wireless* é a mobilidade e elimina a necessidade de cabeamento. Rede cabeadas são mais seguras e rápidas, no entanto sua estrutura pode ser mais cara, além disso se a rede atravessa rodovias públicas está sujeita a regulamentação. O espectro de frequência de rádio para comunicação também é fortemente regulado, no entanto, o espectro de frequência ISM (Industrial Sientific and Medical) facilitou o crescimento comunicação *wireless* para criar redes locais (HOLT; HUANG, 2010)

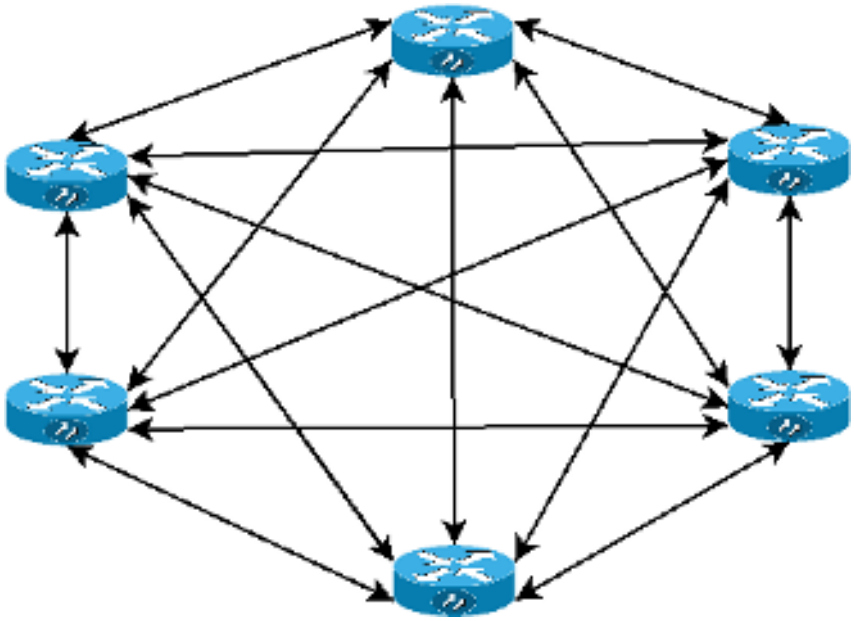
A rede WiFi para computadores mais comumente utilizada, é estruturada na topologia estrela, na qual a comunicação entre todos os componentes da rede devem necessariamente passar por um centralizador/controlador que identifica os atores da comunicação, emissor e receptor, e estabelece um canal de comunicação entre eles. Uma rede *wireless* tem seus padrões definidos pela IEEE (*Institution of Electrical and Electronic Engineers*). Todos os fabricantes de *hardware* devem seguir estes padrões para que a comunicação possa acontecer, independente do fabricante da placa de rede WiFi.

Estes padrões, depois de aprovados pela IEEE, são disponibilizados na forma de RFCs (*Request For Change*), nominadas por uma numeração que representa o tipo de tecnologia que aborda. No caso da rede *wireless*, é regrada pela RFC 802.11. As tecnologias que operam em uma rede *wireless* ou os avanços que a tecnologia sofre possuem uma subdivisão dentro da RFC 802.11 acrescentando caracteres alfanuméricos como por exemplo RFC 802.11b, RFC 802.11g, RFC 802.11n, RFC802.11a.

2.2 REDE MESH

Uma rede *mesh* se caracteriza por ser autoconfigurável, auto gerenciável, orgânica, resiliente e escalável (ZHANG; LUO; HU, 2007). As redes sem fios na topologia malha são chamadas de WMNs (*Wireless Mesh Networks*) e funcionam como uma rede *Ad hoc*, onde todos os *hosts* comunicam entre si, como exemplificado na Figura 1.

Figura 1 – Topologia Mesh



Fonte: Do Autor.

No entanto, uma rede *Ad hoc* somente consegue se comunicar com os nós ao alcance de sua placa *wireless*. Os nós não fazem o roteamento das informações, por isso recebem o nome de *Ad hoc*, expressão do latim que em uma tradução nossa quer dizer "para isto" ou "para esta finalidade". Quando implementamos algoritmos de roteamento a nós *Ad hoc* criamos uma rede *mesh*. Isso proporciona uma ampliação do alcance de comunicação da rede, já que agora os nós podem rotear informações fazendo com que todos eles, mesmo os fora de alcance da placa *wireless*, possam se comunicar. Nesse caso, a informação salta(*hop*) de nó em nó que está ao alcance da placa WiFi, até o nó destino da informação. Este tipo de comunicação recebe o nome de *multihop*.

Redes *mesh* não necessitam de um nó central para estabelecer *links* de comunicação entre os *hosts* devido aos vários protocolos que podem ser implementados. Os nós da rede *mesh* são *hosts* e também são roteadores, assim, as informações são trafegadas ponto a ponto, ou seja, cada nó entrega a informação para seu nó vizinho até que a informação chegue a seu destino.

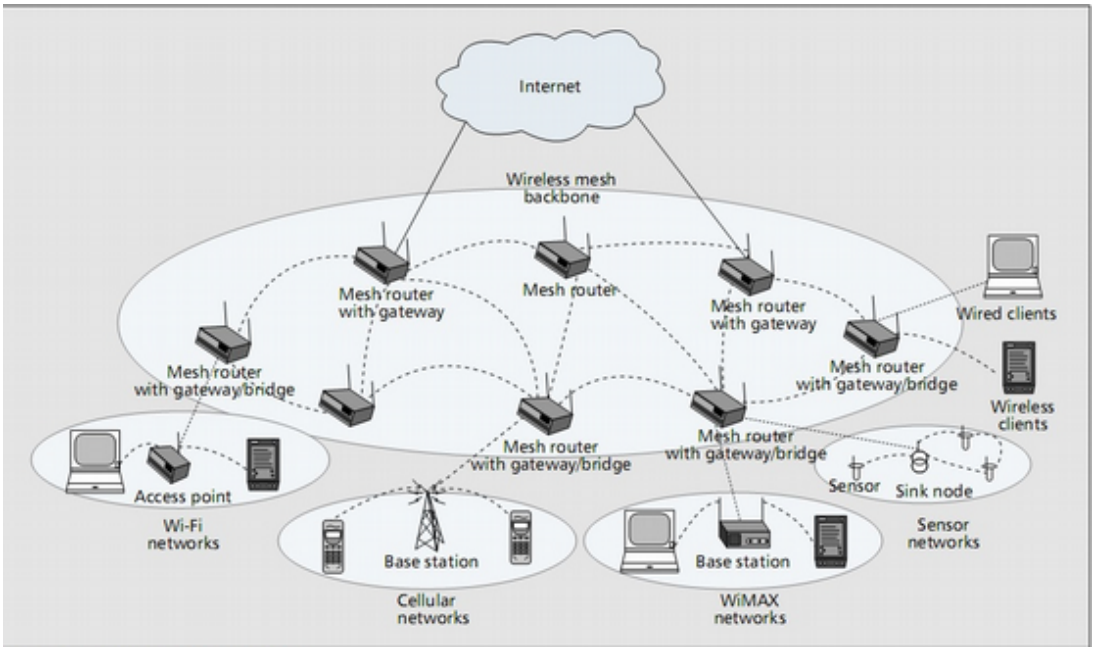
Segundo (ZHANG; LUO; HU, 2007), WMNs possuem grande tolerância a falhas, principalmente em comparação com as WLANs (*Wireless Local Area Network*) no modo infraestrutura, pois requerem um AP (*Access Point*) para estabelecer *links* de comunicação

entre *hosts* da rede. Em uma rede *mesh*, como todos os nós além de serem *hosts* também são roteadores, se um nó pára de funcionar, a rede não deixa de funcionar, apenas aquele nó fica inacessível. Diferente da rede WLAN residencial, por exemplo, que utiliza o roteador *wireless* como *Access Point* para estabelecer *links* comunicação entre os *hosts* e operar como *gateway* para Internet. Nesta situação, onde este parar, a rede toda fica inoperante.

Nas WMNs todos os nós da rede que estão ao alcance de seu rádio transmissor são capazes de se comunicar entre si, portanto, a rede *mesh* também oferece uma melhor redundância, pois pode haver mais de uma rota para se comunicar com um nó da rede. Se ele tiver muitos vizinhos, então muitas rotas na rede são oferecidas para que uma informação chegue até ele, ou muitos caminhos para ele enviar informações. Como as informações são roteadas e não repetidas, a degradação do sinal de rádio não é amplificada como nas soluções adotadas com repetidores WiFi. (AKYLDIZ; WANG, 2009).

Nós das redes *mesh* além das habilidades de serem *hosts* e roteadores ao mesmo tempo, também podem ter a capacidade de serem *gateway/bridge* de uma rede, e não só um nó. Podem existir vários nós com esta finalidade, diferentemente das tradicionais WLANs onde geralmente há apenas um *gateway*. Esta capacidade permite que outros tipos de rede possam se conectar a rede *mesh*, sejam eles dispositivos sem fio no padrão 802.11 ou redes *ethernet* (rede cabeada), uma rede de sensores sem fio, e várias outras redes e dispositivos, fazendo com que se torne híbrida, como exemplificado na Figura 2.

Figura 2 – Topologia Mesh Híbrida



Fonte: (UFRJ, Webpage)

Uma WMN híbrida se torna uma excelente opção como uma rede de segunda camada, pois é capaz de interligar vários sistemas heterogêneos. E devido a sua comunicação

descentralizada, autoconfigurável e gerenciável torna a sua implementação menos onerosa (HOSSAIN; LEUNG, 2008).

A versatilidade de uma rede *mesh* faz com que sua aplicação se torne muito ampla podendo ser projetada para vários propósitos, seja para ampliar cobertura WiFi para acesso à Internet em uma casa ou no shopping, automatização predial, gerenciar sensores em uma fábrica, levar acesso à Internet para locais distantes ou mesmo conectar uma cidade inteira, as possibilidades são muitas.

2.3 PROTOCOLO OLSR

O protocolo OLSR é uma variação da versão do tradicional protocolo de estado de conexão (BOUKERCHE, 2009). É um projeto de código aberto com suporte a vários sistemas operacionais como MS Windows 98, Vista, 2000, XP, 7, 8, 10, Linux, FreeBSD, OpenBSD, NetBSD, Mac OS e em sistemas embarcados com *firmwares* baseados em Linux como roteadores LinkSys, TP-Link, Cisco e outros.

Proposto na dissertação de mestrado de Andreas Toennesen na UniK University na França, o protocolo foi padronizado pela IETF (Internet Engineering Task Force) pela RFC 3626(RFC3626, 2003) para versão 1 do protocolo e na versão 2 a RFC 7181(RFC7181, 2014), submetida pela INRIA [1].

Os protocolos de roteamento reativos buscam uma rota quando há a necessidade de comunicação, ou seja, funcionam sobre demanda. Quando um dado precisa ser enviado, os protocolos reativos perguntam a seus nós vizinhos se através deles é possível chegar a seu destino, e assim sucessivamente, ele vai perguntando e saltando de nó em nó até o seu destino final (HOSSAIN; LEUNG, 2008). O OLSR, por sua vez, é um protocolo de roteamento pró-ativo (OLSR, Webpage). Protocolos pró-ativos como o OLSR utilizam-se de tabelas dinâmicas em cada nó da rede com a rota para cada nó da rede, assim quando a comunicação precisa ser estabelecida, o nó que envia a mensagem já sabe quais as rotas para se chegar ao seu destino. São protocolos conhecidos como *table-drive* (ZHANG; LUO; HU, 2007).

Protocolos pró-ativos geram mais *overheads* na rede e mais sobrecarga de processamento nos roteadores, pois há um custo para manter uma tabela dinâmica de rotas atualizada. Protocolos pró-ativos geralmente utilizam o clássico algoritmo da métrica de contagem de saltos, *hop-count*, para saber a localização dos nós na rede. Esta contagem é feita através de mensagens *broadcast* de *HELLOs*, em que cada nó que a recebe retransmite. Esta técnica gera um alto grau de *flooding* (inundação), pois cada nó retransmite o sinal de *HELLO* para todos os vizinhos que seu rádio alcança, assim como responde para o nó

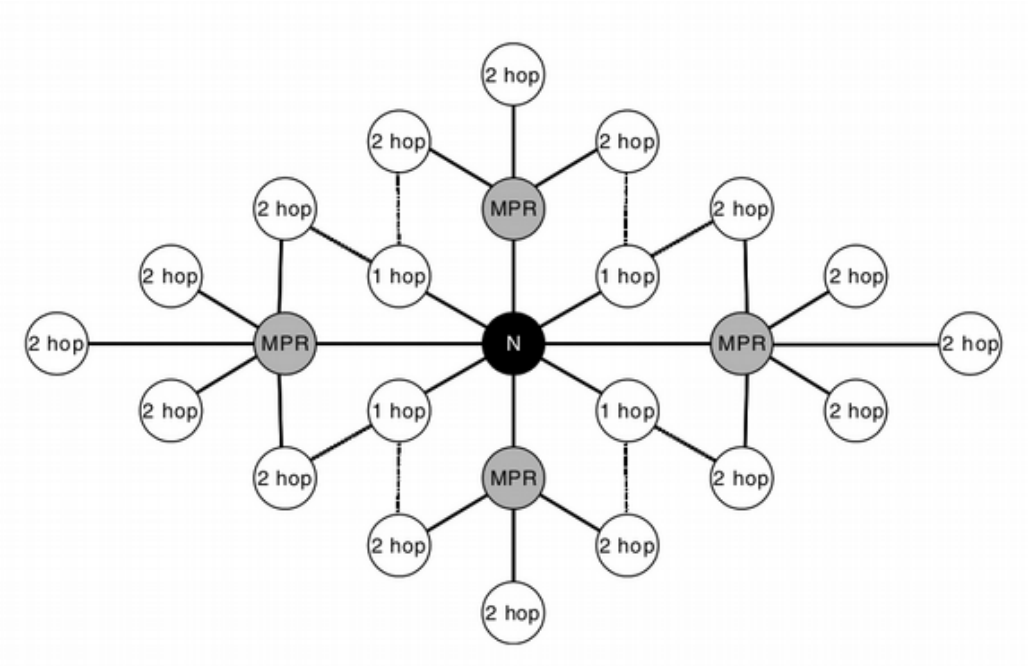
[1] *Institut National de Recherche em Informatique et en Automatique* - organização pública francesa com o objetivo de reunir pesquisadores para fomentar estudos nas áreas de informática e automação

origem, causando *overheads* na rede e sobrecarga de processamento nos roteadores *Ad hoc*. No entanto, o protocolo OLSR otimiza os estados de conexão diminuindo o número de mensagens de *broadcast* retransmitidas, ou seja, diminui o número *HELLOs* na rede. Para isso, o protocolo OLSR implementa nós MPRs (*Multi Point Realy*) em que somente eles são responsáveis por retransmitir mensagens de *broadcast* para identificar a localização dos nós na rede.

Como uma rede *mesh* tem por característica uma rede autoconfigurável e gerenciável, o protocolo de roteamento deve fornecer isso. No protocolo OLSR a implementação do nó MPR é realizada pelos próprios nós da rede, onde cada nó autoconfigura qual nó vizinho é o nó MPR. Isso acontece por que no algoritmo do protocolo OLSR cada nó deve identificar que o nó MPR deve estar a no máximo dois saltos de distância para retransmissão de sua localização. Cada nó envia mensagens de *broadcast* somente para os nós MPRs que ele calculou e não para toda a rede. Nós MPRs, enviam mensagens de *broadcast* TC (*Topology Control*). Com isso são criadas as tabelas dinâmicas de roteamento, pois sabendo o número de MPRs existentes e que estão sempre a no máximo 2 saltos de distância dos nós, é possível calcular onde estão e montar uma tabela (ZHANG; LUO; HU, 2007).

A implementação de nós MPRs é um aspecto central do protocolo OLSR. Sua implementação permite diminuir pela metade o *flooding* de mensagens de *broadcast* retransmitidos na rede para localizar os nós da rede, como constatado na Figura 3. Isto causa uma diminuição significativa no número de *overheads* e sobrecarga de processamento de roteadores (Boukerche 2009).

Figura 3 – Flooding OLSR



Fonte: (ZHANG; LUO; HU, 2007)

O protocolo ainda oferece mensagem de *broadcast* MID (*Multi Interface Declaration*), em que o nó declara ter mais de uma interface de comunicação, ou seja, mais de um rádio para transmissão. Na primeira versão do protocolo OLSR, se um nó se declarasse como MID, o protocolo o escolhe uma interface de trabalho para fazer o link com o nó vizinho.

Com o avanço do protocolo, agora o OLSR consegue operar com mais de um rádio de forma simultânea e independente. Isso faz com que um nó MID possa se comunicar com nós vizinhos com seus múltiplos rádios, o que torna a comunicação mais confiável, pois se um dos rádios do nó MID falhar, o nó ainda está ativo na rede utilizando o outro rádio. Outra possibilidade de se possuir dois rádios ou mais é a operação simultânea deses rádios ampliando a banda de comunicação, tornando o tráfego mais rápido, pois consegue enviar mais pacotes por vez conforme o número de rádios, placas *wireless* que ele possui, ou ainda estabelecer *links* com mais de um nó.

Outro serviço de mensagem de *broadcast* do OLSR é o HNA (*Host and Network Association*), em que o nó da rede que envia esta mensagem anuncia para os outros nós que ele é um *gateway* para a Internet. Diferentemente das WLANs em modo infraestrutura, onde geralmente há apenas um *gateway* de rede, nas redes *mesh* com o protocolo de roteamento OLSR pode haver mais de um *gateway* para a acesso à Internet ou a outras redes ou mesmo a outros nós.

Um nó da rede *mesh* que necessite de um *gateway* para enviar dados para outra rede, Internet ou não, ele procurará em sua tabela de rotas qual nó da rede está sinalizando com a mensagem HNA, se existir mais de um nó sinalizando que é um *gateway*, o protocolo OLSR utiliza o clássico algoritmo de métrica de contagem de saltos, isto significa que o nó que necessita de um *gateway* sempre vai escolher o nó HNA com menor número de saltos, ou seja, o nó que envia mensagens HNA mais perto dele.

Uma rede *mesh* que possua dois nós que são *gateways* para a Internet, por exemplo, você consegue dividir o tráfego de dados na rede *mesh* e ainda torna a saída para Internet redundante, enquanto um nó *gateway* estiver ativo na rede *mesh*, sempre terá uma saída para Internet, e sem a necessidade de intervenção na rede, o protocolo OLSR gerencia isto automaticamente através das suas tabelas de rotas presentes em todos os nós ativos da rede *mesh*.

2.4 DISPOSITIVOS DE HARDWARE E SOFTWARE

Este capítulo destina-se a uma breve descrição dos dispositivos de hardware e software que foram utilizados para implementar a rede *mesh* que tornou possível este trabalho, uma abordagem das principais características que baseou-se esta solução.

2.4.1 Roteadores wireless

Padronizados pelo protocolo IEEE 802.11 os roteadores *wireless* operam em frequências de 2.4 e 5 GHz e em larguras de banda que variam de 54 a 450 Mbps. Um roteador *wireless* possui várias tecnologias incorporadas a ele, possui rádios transmissores para propagar os dados pelo ar, um *firmware* que além da função de fazer o roteamento das informações também carrega a função de fornecer segurança, autenticando senhas criptografas para ingresso na rede WiFi, e um *Firewall* para filtrar pacotes de dados, também pode incorporar, geralmente, um *switch* para fornecer acesso a rede *ethernet*(rede cabeada).

Um equipamento cada vez mais comum hoje em dia, fornecendo acesso à Internet para dispositivos móveis, e até mesmo, dispositivos estáticos como *desktops* já que a implementação de uma rede sem fio gera menos transtornos que a implementação de uma rede cabeada, basicamente bastando a instalação de um roteador *wireless* para que a rede esteja funcional.

Nos dias atuais encontrasse uma gama elevada de marcas, modelos, valores e tecnologias empregadas, de uma simples roteador *wireless* doméstico, até equipamentos profissionais utilizados em redes corporativas, ampliando seu leque de utilização no mundo atual, já que os paradigmas de rede *wireless* também estão mudando, como a tecnologia de redes em malha.

2.4.2 Minicomputadores

Com o avanço da tecnologia da computação móvel e o barateamento dos *hardwares*, permitiu a criação de computadores de placa única, como também são conhecidos os minicomputadores, se tornaram um excelente opção para aprendizado de tecnologias como programação e robótica por exemplo. São capazes de rodar sistemas operacionais como Linux e Microsoft Windows, dando uma versatilidade enorme a estes minicomputadores.

A arquitetura ARM de processadores, presentes em *smartphones*, *tablets* e outros dispositivos, evolui para um alto poder computacional com baixo consumo de energia, por isso sua larga utilização em celulares, mas, que também permitiu criar computadores que cabem na palma de uma mão.

A evolução dos *smartphones* e outros dispositivos móveis permitiu o barateamento dos processadores de arquitetura ARM, e também os minicomputadores, hoje pode-se comprar um minicomputador com menos de US$25,00, por isso, tornou-se uma ótima opção para aprendizado de tecnologias.

Há no mercado uma variedade de marcas e modelos de minicomputadores, e o mais famoso deles, o Raspberry Pi, que já vendeu mais de 10.000.000 unidades de seus modelos. Ele tem uma distribuição Linux feito sob medida para ele, o Raspbian, mantido pela Raspberry Pi Foundation, tem suporte a várias linguagens de programação como *Node.Js*,

Java, *Python*, etc. Assim como, tem suporte aos mais variados *hardwares* disponíveis no mercado, permite desenvolver soluções/aplicações para a utilização de dispositivos IoTs para os mais distintos *hardwares*. Como um sistema operacional de código aberto, o Raspbian está sempre em desenvolvimento pela comunidade Linux e também oferece ao desenvolvedor maior gama de opções para o desenvolvimento de soluções/aplicações (RASPBIAN, Webpage).

2.4.3 Hostapd

O *software Hostapd* é um servidor para autenticação de hosts para *Access Point*, tem suporte para as criptografias de autenticação WPA, WPA2, EAP pode autenticar em servidores RADIUS(*Remote Authentication Dial In User Service*) e transformar um computador ou dispositivo computacional baseado em Linux, desde que possua uma interface de rede *wireless*, em um *Access Point* (HOSAPD, Webpage).

2.4.4 Dnsmasq

Dnsmasq proporciona a pequenas infraestruturas de rede WiFi, agregado a *softwares* como *hostapd*, suporte a *DHCP* e *DNS*, é um *software* leve e multiplataforma, com suporte a IPv4 e IPv6 (DNSMASQ, Webpage), o que o torna uma excelente ferramenta de *software* para funcionar em dispositivos embarcados com suporte a Linux ou minicomputadores como o Raspberry Pi.

2.5 FERRAMENTAS PARA ANALISE DE DESEMPENHO

Este capítulo apresenta ferramentas de software utilizadas em redes de computadores para os mais variados fins, ferramentas muito utilizadas por profissionais da área redes, que vão desde o monitoramento de dados a descoberta de falhas de segurança na rede, ferramentas estas que serão utilizadas para testar e validar a solução apresentada neste trabalho.

2.5.1 TCPdump

Uma ferramenta interessante de análise de pacotes trafegados em rede é o TCPdump. A ferramenta é executada em linha de comando, sem a necessidade de interface gráfica, e é suportada por vários sistemas operacionais (TCPDUMP, Webpage). Aliada ao pouco espaço que ocupa em disco, no caso do Linux menos do que 1,2MB, tornam esta ferramenta muito flexível para sua utilização. Dado a sua versatilidade para *snifar* uma rede, com uma grande quantidade de opções para a forma de captura dos pacotes de rede, tornam esta ferramenta uma valiosa opção para validação de testes em redes TCP/IP.

2.5.2 Plugins jsoninfo e oslrd_dot_draw

A página web do OLSR (OLSR, Webpage), descreve como instalar e configurar o *plugin jsoninfo* capaz de enviar informações sobre o status *runtime* (tempo de execução) de informações com nós vizinhos, topologia, interfaces e rotas, assim como as configurações do protocolo OLSR acessando a porta 2004 do TCP de cada nó da rede *mesh*. Existe também o *plugin oslrd_dot_draw* capaz de fornecer graficamente a disposição dos nós da rede, um utilitário bem eficaz para a visualização da infraestrutura da rede.

2.5.3 Nmap

O *NMAP* (*Network Maper*) é um *software* utilitário *free* e *open source* para descobertas na rede e auditoria de segurança. Uma excelente ferramenta para monitoração em tempo real de *hosts* e serviços. O *NMAP* utiliza pacotes *raw* IP para descobrir quais hosts estão disponíveis na rede e quais serviços estes *hosts* estão oferecendo, com informações de nome do aplicativo e versão por exemplo, permite também descobrir qual o sistema operacional utilizado pelo *host*. Ele foi projetado para varrer rapidamente grandes redes, mas funciona bem para monitorar um único *host*. O *NMAP* é multiplataforma e tem suporte a vários sistemas operacionais com Linux, Windows e Mac OS, conforme (NMAP, Webpage).

2.6 TRABALHOS RELACIONADOS

Este capítulo apresenta trabalhos acadêmicos relacionados com o presente trabalho, referenciando o uso de redes *mesh* e também o protocolo OLSR, principais tecnologias utilizadas neste trabalho, como soluções que podem ser empregadas em vários cenários.

No trabalho de (AVELAR et al., 2010), propõe uma arquitetura de rede heterogênea para cidades inteligentes de baixo custo, utilizando o *firmware* Openwrt e tecnologias Iot como Zigbee e Sun SPOT, onde uma rede de sensores zigbee/Sun SPOT forma a rede em primeira camada, comunicando com uma rede em segunda camada, formada por roteadores com *firmware* Openwrt trabalhando em modo *mesh* na cidade de Recife - PE.

No projeto de (MEGGER, 2018), avaliou-se QoS de tráfego de dados em uma rede *mesh*, evidenciando a importância da priorização de dados específicos em detrimento a outros, utilizando o protocolo OLSR e roteadores com *firmware* DD-WRT(que posteriormente tornou-se o *firmware* OpenWRT) para criar o cenário de testes.

Em (AGUIAR et al., 2007), realizou-se simulações com Network Simulator para comparar dois protocolos proativos, o AODV e o OLSR, para o desempenho no tráfego de dados e voz na rede *mesh* utilizada pela Universidade Federal do Pará como *backbone* interligando de seus prédios distribuídos pela cidade de Belém do Pará. Nesta simulação o

protocolo AODV obtiveram um melhor desempenho segundo os padrões de testes aplicados, através de conexões simultâneas de dados e voz

No trabalho de (GOMES et al., 2009) utilizou-se a infraestrutura de *backbone mesh* da UFPA para realizar simulações com o Network Simulator, para comparar o protocolo OLSR com o protocolo OLSR-DC e OLSR-MC, estas extensões do protocolo do OLSR - extensões são desenvolvidas para aprimorar ou criar funções específicas para uma tecnologia existente - para verificar o desempenho quanto a voz e vídeo, e acabou demostrando segundo os padrões de testes adotados, que as extensões do protocolo obteve-se bons resultados na trafego de voz e vídeo.

No estudo de caso apresentado em (WNDW, 2007) para levar a Internet a Comunidade de Dharamsala na Índia, em um terreno montanhoso de difícil acesso, utilizou-se de uma rede *mesh* com o protocolo OLSR, para formar um *backbone* para a Internet, o que se provou uma excelente escolha onde a geografia dificulta a implementação de uma rede centralizada tradicional, uma rede *mesh*, decentralizada, permite ampliar o alcance de comunicação com menor custo e ótimo desempenho.

Como visto, uma *Mobile Ad hoc Network* gerenciada pelo protocolo OLSR permite um alto grau de aplicações para redes, podendo ser utilizadas para levar a Internet a regiões de difícil acesso ou para dar suporte a dispositivos IoTs, implementando uma rede *mesh*.

3 IMPLEMENTAÇÃO DA REDE MESH

Para compor os nós da rede *mesh* que formam a infraestrutura sem fio para conexão dos dispositivos IoTs, foi utilizado um roteador *wireless* da marca TP-Link modelo N750 (TL-WDR4300) e dois computadores de placa única Raspberry Pi 3 modelo B. O Roteador *wireless* TP-Link N750 modelo TL-WDR4300, opera em duas frequências de rádio transmissão/recepção, em 2.4GHz a 300Mbps e em 5GHz a 450Mbps. Seu *firmware* proprietário é baseado em Linux. O fato de ser um sistema operacional de código aberto, permitiu que a comunidade na Internet pudesse criar seus próprios *firmwares*, surgindo assim, os *firmwares* OpenWrt. Esses *firmwares* são altamente configuráveis e adaptáveis. A substituição do *firmware* possibilita estender as funcionalidades do roteador com funções não implementadas pelo fabricante, como neste caso, para dar suporte ao protocolo OLSR.

O roteador em questão, teve seu *firmware* proprietário substituído pelo *firmware* OpenWrt versão 18.06.2, este *firmware* funciona como uma distribuição Linux, que possui um repositório web que disponibiliza atualizações para os pacotes instalados e também proporciona a instalação de novos pacotes através da Internet. Após a substituição pelo novo *firmware*, foi instalado o pacote *olsrd* por meio da interface web de administração do roteador TP-Link.

Este modelo de roteador da TP-Link possui dois rádios transmissores/receptores, mas, o protocolo OLSR foi configurado apenas na placa de rede *wireless* que opera na faixa de 2.54GHz. Placa esta configurada para trabalhar em modo *Ad hoc* e operar no canal 11 da faixa de frequência 2,54GHz. Foi necessário adicionar uma interface virtual nas configurações do OpenWrt para criar uma rede *wireless* com o ESSID (*Extend Service Set ID*) de nome OLSR, com endereço IP estático 192.168.0.1, para que este nó de rede pudesse fazer parte de uma rede *mesh*, desde que esta rede operasse no canal 11 e tivesse o mesmo ESSID, neste caso, OLSR.

A placa rede *ethernet* WAN deste roteador TP-Link foi conectado um cabo de rede *ethernet* e na outra ponta a um roteador com a função de prover o acesso à Internet, fornecendo a rede *mesh* um *gateway* para Internet.

Durante a instalação do pacote *olsrd* no roteador TP-Link o protocolo já reconheceu que possui acesso à Internet e automaticamente configurou este nó para enviar mensagens HNA para a MANET informando que ele é um *gateway* para uma rede externa, neste, caso a Internet. Para compor os outros nós da rede *mesh* foram utilizados dois computadores de placa única Raspberry Pi 3 modelo B com processador de 1GHz de 4 núcleos de arquitetura ARM, 1GB de memória RAM, 4 portas USB, 1 placa de rede WiFi com suporte as tecnologias 802.11b,g e n, tem suporte a Bluetooth 4.1 e 1 interface *ethernet*

com velocidade máxima de tráfego de dados de 100Mbps. Os minicomputadores Raspberry Pi são computadores de baixo custo e alto desempenho (RASPBERRY, Webpage).

Nos dois minicomputadores Raspberry Pi 3 modelo B, o sistema operacional instalado foi a distribuição Linux baseada no Debian o Raspbian, que é mantida pela própria Raspberry Pi Foundation, fabricante dos minicomputadores. O SO foi instalada em dois cartões de memória micro SD com capacidade de armazenamento de 16GB. A distribuição Linux Raspbian, como toda distribuição Linux, possui repositórios web para atualização e instalação de pacotes, e um dos pacotes mantidos em seu repositório é o *olsrd*, que quando instalado fornece suporte ao protocolo OLSR, bastando apenas instalar o pacote com o comando *apt-get*.

A configuração do protocolo OLSR nos Raspberrys, após a instalação do pacote *olsrd*, é necessário editar o arquivo *olsrd.conf* localizado em */etc/olsrd/*, para definir a configuração operacional do protocolo e os *plugins* utilizados. Optou-se por deixar o protocolo com a configuração padrão da instalação, apenas informando a interface de rede em que o protocolo deve atuar, neste caso a wlan0, que nomina a placa de rede *wireless onboard* do Raspberry Pi 3 B no sistema operacional Raspbian, configurações estas dispostas no Apêndice A, diferentemente da configuração do protocolo OLSR no roteador TP-Link que é toda gráfica através da interface de administração web, que é acessada por um Navegador de Internet no endereço IP do roteador.

Para que pudessem ingressar em uma rede *mesh* as placas de redes Wi-Fi dos minicomputadores Raspberry precisaram ser configuradas para operar em modo *Ad hoc*, no canal 11 da faixa de frequência 2,54GHz, e ter o ESSID configurado para o nome OLSR, e ter um endereço de IP estático, aqui 192.168.0.10 e 192.168.0.11, e assim, mais dois nós da rede *mesh* foram adicionados, para criar a infraestrutura de uma rede MANET.

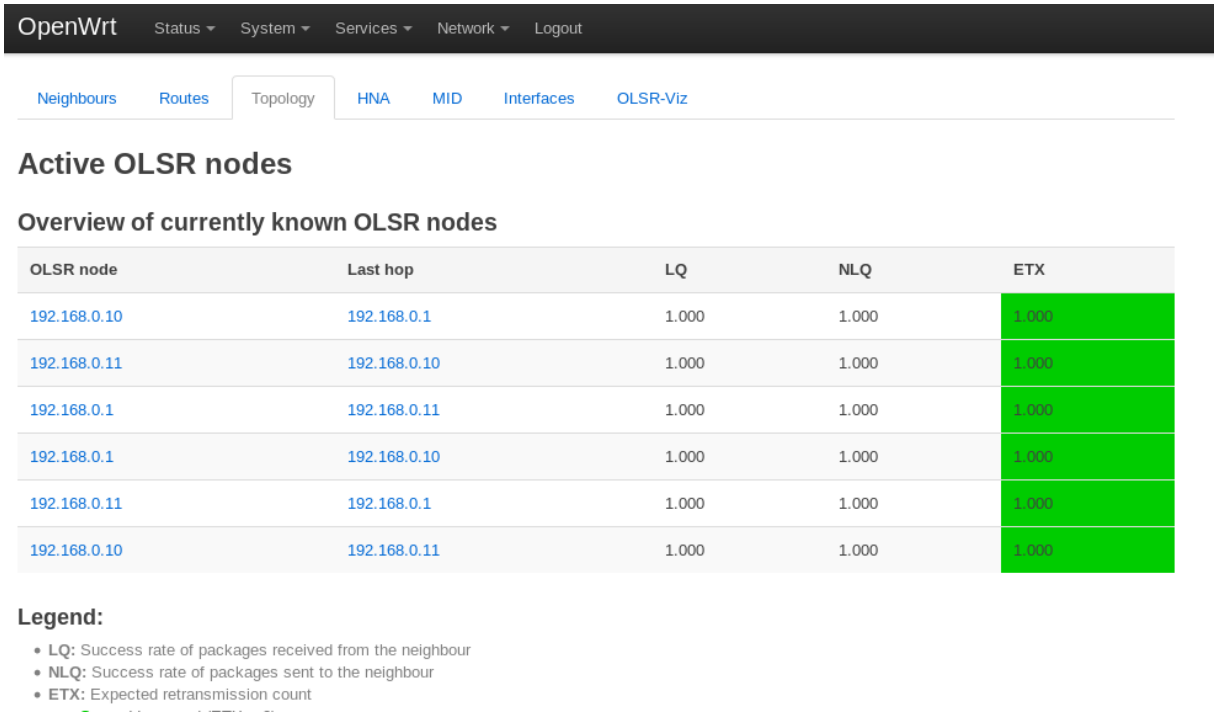
3.1 INFRAESTRUTURA DA MANET IMPLEMENTADA

Os protocolos pró-ativos, do tipo *table-drive*, como é o protocolo OLSR, mantém em cada nó da rede, uma tabela atualizada com todos os nós ativos da rede e as rotas para se chegar a cada nó. Neste caso, o protocolo OLSR, na rede implementada, criou rotas de rede no sistema operacional Raspbian(Raspberry Pi) e no OpenWrt(roteador TP-Link) para cada endereço IP dos nós da rede que estão ao alcance do rádio transmissor/receptor das placas de rede *wireless*, que no primeiro momento pós implementação da rede *mesh*, os 3 nós estão ao alcance um dos outros, portanto, cada nó de rede possui uma rota de rede para cada endereço IP dos *hosts*, configuração esta que pode ser aferida com o comando *route* do Linux, em cada host.

No entanto, no roteador TP-Link é possível verificar estas informações dos nós da rede *mesh* através de um *plugin* que já vem habilitado na instalação do pacote *olsrd* no

OpenWrt, acessando a interface web de administração via endereço IP do roteador em um Navegador de Internet, e assim, como vemos na Figura 4, visualizasse as rotas criadas nos nós da rede *mesh*, e assim, conferir a infraestrutura criada pelo protocolo OLSR na rede implementada para este trabalho, onde cada nó possui uma rota de endereço IP para cada nó vizinho.

Figura 4 – Rotas da Rede Mesh Implementada



Fonte: Do Autor.

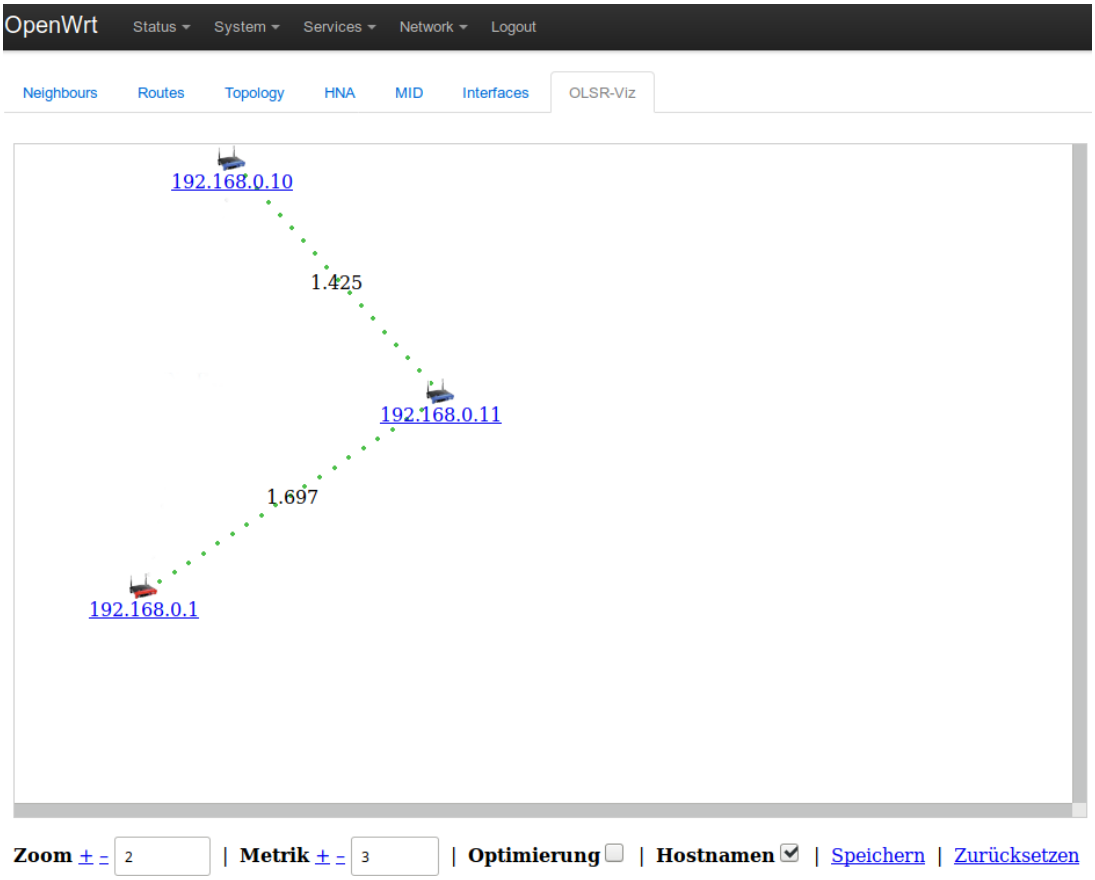
Após a implementação da infraestrutura da MANET deu-se início aos testes. O primeiro foi configurar um *laptop* com o sistema operacional Linux, para simular um dispositivo IoT, configurando a placa *wireless* em modo *Ad hoc*, canal de frequência 11 e com ESSID com o nome OLSR, através do arquivo *interfaces*, localizado em */etc/network/*, este, que configura as placas de redes físicas e virtuais no ambiente Linux.

Com estas configurações foi possível ingressar na rede *mesh* OLSR, e efetuar teste de comunicação com o comando *PING* do Linux, com todos os nós, o roteador TP-Link e os dois Raspberry Pi, todos estão ao alcance de seus rádios transmissores/receptores, o comando teve êxito no teste de comunicação com endereços IPs dos *hosts* da rede *mesh*, assim como, a recíproca foi verdadeira.

O próximo passo foi verificar se o protocolo OLSR está operacional, foi implementado uma infraestrutura de testes, em um prédio de 6 andares, onde o roteador TP-Link foi colocado no primeiro andar, o Raspberry Pi com endereço IP 192.168.0.11 no terceiro andar e o último Raspberry Pi no sexto andar. A captura de tela, provida pelo *plugin olsrd_dot_draw* através da interface web via endereço IP do roteador TP-Link em um

Navegador de Internet, representada pela Figura 5, constatasse a operacionalidade do protocolo OLSR, ao mesmo tempo, verificasse a arquitetura da rede.

Figura 5 – Arquitetura MANET implementada.



Fonte: Do Autor.

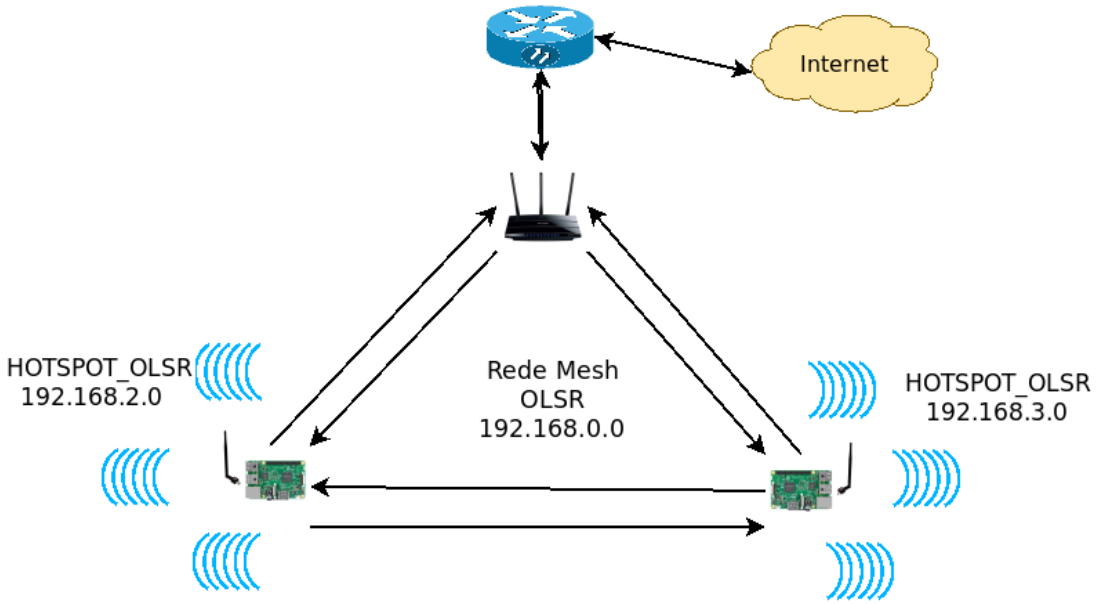
Um *laptop* de sistema operacional Linux, com sua placa *wireless* configurada para modo *Ad hoc*, com ESSID igual a OLSR, operando no canal 11 e com IP 192.168.0.20(sem protocolo OLSR instalado) para simular um dispositivo IoT, estando no primeiro andar, foi utilizado para efetuar um teste comunicação através do comando *PING* do Linux, e como resultado, houve comunicação apenas apenas com o roteador TP-Link, os demais *hosts* da rede resultaram em falha de comunicação nos testes. No entanto acessando remotamente o TP-Link via comando *SSH*(*Secure Shell*) através do *laptop*, o comando *PING* resultou com sucesso os testes de comunicação com todos os endereços de nós presentes na rede *mesh*. Porém, ao se conectar remotamente ao Raspberry Pi do segundo e sexto andar pelo roteador TP-Link, também resultou em falha de comunicação com o *laptop* pelo comando *PING*.

Observou-se que o fato do *laptop* não possuir o protocolo OLSR instalado inviabilizou a comunicação com os Raspberry distantes do alcance de sua antena de transmissão, pois, o *laptop* não está operando com um nó de uma rede MANET e sim como um *host* de uma rede *Ad hoc*, em que os nós da rede só conseguem se comunicar com quem está ao

alcance de seus rádios. Conclui-se também que o protocolo OLSR não encaminha pacotes de dados de nós vizinhos que não tem o protocolo rodando para a rede *mesh* e vice-versa. Neste panorama uma rede *mesh* com protocolo OLSR para criar uma infraestrutura de segunda camada de transporte de dados para dispositivos IoTs torna-se inviável.

Para solucionar esse problema, utilizou-se a tecnologia empregadas em HOTSPOTs, criar um ponto de acesso(*Access Point*) a MANET OLSR, para isso, foi conectado a cada Raspberry Pi um adaptador *wireless* USB, adicionando assim mais uma interface *wireless*, para funcionar como um *hotspot*, para estes adaptadores WiFi USB foi configurado o ESSID de nome HOSTAPD_OLSR e com endereços de redes IPs distintas, 192.168.2.1 e 192.168.3.1. Nos minicomputadores Raspberry Pi o protocolo OLSR foi configurado para informar todos os nós, através de mensagens HNA, que compõem a MANET, que cada um deles possui um *gateway* para as redes IPs 192.168.2.0/24 e 192.168.3.0/24, para poderem serem acessados(encontrados), formando uma infraestrutura sem fio para conexões acessíveis a dispositivos com suporte WiFi sem a necessidade de instalar o protocolo OLSR nestes dispositivos, melhor compreendida visualizando a Figura 6.

Figura 6 – Arquitetura da Rede Mesh Implementada



Fonte: Do Autor.

Após a implementação da infraestrutura da MANET realizou-se a validação da solução proposta. Os testes realizados são descritos no capítulo a seguir.

4 TESTES E VALIDAÇÕES

Este capítulo aborda todas as configurações que foram necessárias serem efetuadas para que a solução proposta por este trabalho pudesse ser validada, assim como, os testes realizados e os resultados obtidos.

4.1 PRIMEIRA CONFIGURAÇÃO DOS HOTSPOTs

A primeira configuração das placas USB *wireless* como *hotspot/Access Point*, foi criando uma interface virtual *bridge*, de nome *br0*, entre a placa de rede *onboard* dos Raspberry Pi(rede *mesh*) e a placa de rede USB *wireless*(*hotspot*). Uma "ponte" em uma rede permite que duas redes distintas tenham acesso uma a outra como se fossem uma mesma rede, como faz um roteador *wireless* na configuração tradicional fornecendo acesso à Internet sem fio para *desktops*, *laptops*, *smartphones* e dispositivos sem fio com suporte a a tecnologia 802.11, esta "ponte virtual" entre as placas *wireless* presentes no roteador e placa de rede *ethernet*(com fio) conectada ao modem ou roteador do provedor de Internet, torna possível a comunicação dos dispositivos conectados ao roteador (rede LAN) com a Internet (rede WAN).

A ponte virtual entre as placas *wireless onboard* e USB nos minicomputadores, foi pensada com o objetivo de tornar possível um controle dinâmico de endereçamento dos dispositivos que se conectariam aos *Access Point*, implementando um servidor DHCP(*Dynamic Host Control Protocol*) na rede *mesh*.

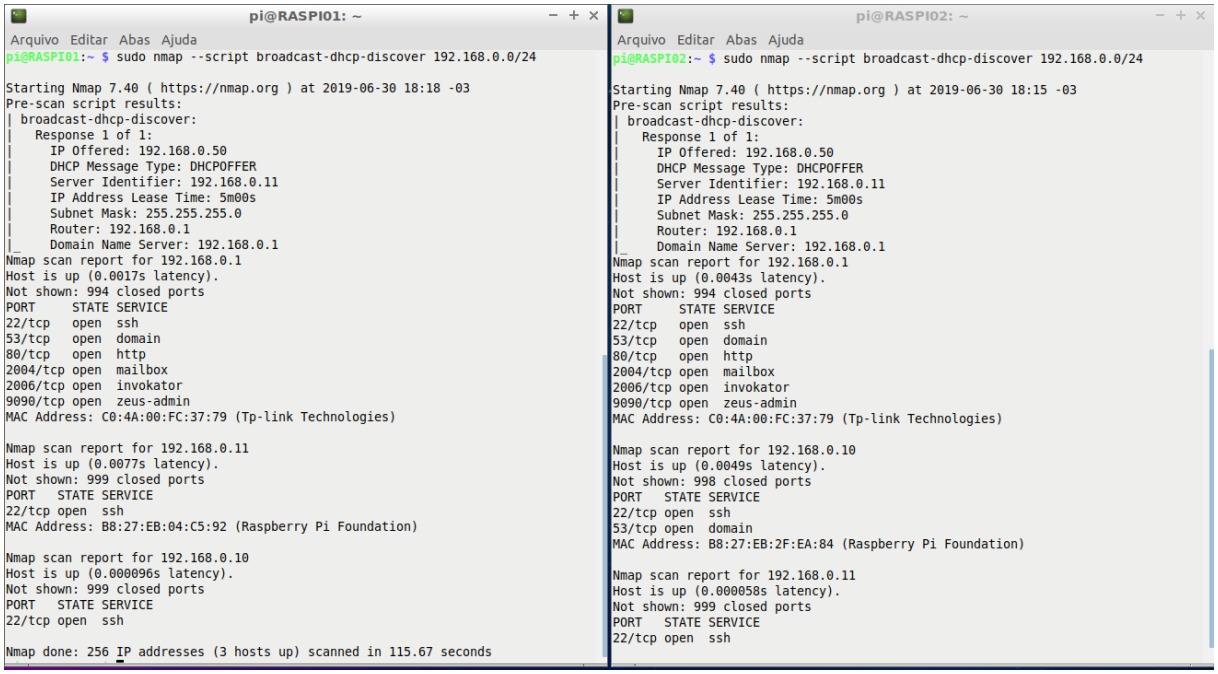
Para hospedar o servidor *DHCP* foi escolhido o Raspberry Pi de endereço IP 192.168.0.11, *hostname* RASPI02, para a instalação do servidor DHCP foi instalado o pacote *isc-dhcp-sever* presente no repositório web do sistema operacional Raspbian. No *isc-dhcp-server* foi configurado um *range* de endereços IPs, do 192.168.0.50 ao 192.168.0.100, para estar a disposição dos *hosts* que se conectarão aos *Access Point*.

Após instalado e configurado o servidor *DHCP* foi utilizado o *software NMAP*, um *network maper*(mapeador de rede), uma ferramenta de *software* capaz de escanear uma rede de computadores, a assim, descobrir quais os serviços de rede que estão sendo oferecidos, os serviços ativos na rede.

Realizou-se um escaneamento com a ferramenta de software NMAP na rede 192.168.0.0/24, utilizando a opção *–script broadcast-dhcp-discover*, executado o comando tanto no RASPI02(servidor *DHCP*) como no RASPI01, e assim verificar se o servidor *DHCP* está ativo localmente(RASPI02) e se está "escutando" a rede para requisições para endereçamento dinâmico, neste caso, "escutando" se o RASPI01 está enviando requisições

ao servidor *DHCP* na rede, verificasse na Figura 7, o servidor *isc-dhcp-server* operacional, localmente e para a rede.

Figura 7 – Servidor isc-dhcp-server



Fonte: Do Autor.

Para testar a implementação do *hotspot* utilizou-se um *laptop* com Linux como SO, no gerenciador de conexões *wireless*(NetworkManager) do Linux, visualizou-se a rede WiFi de nome(*ESSID*) HOSTAPD_OLSR, até aí, tudo funcionando conforme o idealizado, no entanto, após várias tentativas de conexão à rede HOSTAPD_OLSR, nenhuma ocorreu com sucesso, não foi possível se conectar a rede WiFi HOSTAPD_OLSR. O *NetworkManager* depois de alguns segundos, não estabelecia a conexão, voltava a se conectar a uma rede WiFi previamente configurada em acessos anteriores, se estive em um ambiente coberto por esta rede, ou simplesmente informava o status de desconectado.

Outro teste realizado para tentar validar o ponto de acesso HOSTAPD_OLSR, conectou-se Raspberry de *hostname* RASPI01 ao roteador TP-Link via cabo ethernet, já que o roteador possui um *switch* de quatro portas *ethernet* para conexões cabeadas, estas conexões *ethernet* estão configuradas para fornecer um endereço IP aos *hosts* que se conectarem a elas, já que o servidor DHCP presente no roteador TP-Link está configurado para atuar nestas portas. Também alterou-se a configuração da *bridge*, a interface virtual que cria uma "ponte" entre as placas WiFi *onboard*(rede *mesh* e a USB plugada(*Access Point* HOSTAPD_OLSR), para criar uma *bridge* entre a placa WiFi USB, e a placa *ethernet onboard* do RASPI01. Assim isolou-se a rede *mesh*, para testar a conexão do com a rede WiFi HOSTAPD_OLSR, e neste cenário, foi estabelecido com sucesso a conexão ao *hotspot* pelo *laptop*, aferindo o funcionamento da placa WiFi USB plugada e o software *hostapd* provendo um *Access Point*.

Com a certeza do funcionamento do software *hostapd* sob a placa WiFi USB plugada, foi restabelecida a primeira configuração da *bridge* entre as placas WiFi presentes no RASPI01 realizou-se mais tentativas de conexões ao *hotspot* HOSTAPD_OLSR, as quais, novamente todas sem sucesso. Analisando o log gerado pelo *NetworkManager*, descobriu-se que a não conexão a rede WiFi HOSTAPD_OLSR não ocorria pelo erro de *request time out* enquanto efetuava a transação DHCP, o erro pode ser visualizado pela captura de tela do trecho do arquivo de log do *NetworkManager* conforme a Figura 8.

Figura 8 – Trecho capturado do arquivo de log do *NetworkManager*

```
-- Logs begin at Thu 2018-10-04 00:09:01 -03. --
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.0780] audit: op="connection-activate" uuid="0bd8cda3-87ab-4cd7-b299-455729ee5b6
    name="HOSTAPD" pid=1173 uid=1000 result="success"
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.0788] device (wls1): state change: deactivating -> disconnected (reason 'new-activation
    iface-
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2443] Config: added 'ssid' value 'HOSTAPD'
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2444] Config: added 'scan_ssid' value '1'
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2444] Config: added 'bgscan' value 'simple:30:-80:86400'
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2444] Config: added 'key_mgmt' value 'WPA-PSK'
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2445] Config: added 'auth_alg' value 'OPEN'
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2445] Config: added 'psk' value '<hidden>'
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2862] device (wls1): supplicant interface state: disconnected -> authenticating
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.2951] device (wls1): supplicant interface state: authenticating -> associating
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.3065] device (wls1): supplicant interface state: associating -> 4-way handshake
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.3160] device (wls1): supplicant interface state: 4-way handshake -> completed
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.3167] device (wls1): Activation: (wifi) Stage 2 of 5 (Device Configure) successful.  Co
    wireless network 'HOSTAPD'.
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.3174] device (wls1): state change: config -> ip-config (reason 'none', sys-iface-state: 'm
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.3186] dhcp4 (wls1): activation: beginning transaction (timeout in 45 seconds)
1:08:27 laptop NetworkManager[660]: <info>  [1558238907.3292] dhcp4 (wls1): dhclient started with pid 5876
1:08:27 laptop dhclient[5876]: DHCPDISCOVER on wls1 to 255.255.255.255 port 67 interval 3 (xid=0x601ffa7b)
1:08:30 laptop dhclient[5876]: DHCPDISCOVER on wls1 to 255.255.255.255 port 67 interval 8 (xid=0x601ffa7b)
1:08:38 laptop dhclient[5876]: DHCPDISCOVER on wls1 to 255.255.255.255 port 67 interval 15 (xid=0x601ffa7b)
1:08:53 laptop dhclient[5876]: DHCPDISCOVER on wls1 to 255.255.255.255 port 67 interval 21 (xid=0x601ffa7b)
1:09:12 laptop NetworkManager[660]: <warn>  [1558238952.8078] dhcp4 (wls1): request timed out
1:09:12 laptop NetworkManager[660]: <info>  [1558238952.8079] dhcp4 (wls1): state changed unknown -> timeout
1:09:12 laptop NetworkManager[660]: <info>  [1558238952.8410] dhcp4 (wls1): canceled DHCP transaction, DHCP client pid 5876
1:09:12 laptop NetworkManager[660]: <info>  [1558238952.8410] dhcp4 (wls1): state changed timeout -> done
1:09:12 laptop NetworkManager[660]: <info>  [1558238952.8427] device (wls1): state change: ip-config -> failed (reason 'ip-config-unavailable', sy
    state: 'managed')
1:09:12 laptop NetworkManager[660]: <info>  [1558238952.8441] manager: NetworkManager state is now DISCONNECTED
1:09:12 laptop NetworkManager[660]: <warn>  [1558238952.8463] device (wls1): Activation: failed for connection 'HOSTAPD'
```

Fonte: Do Autor.

Após identificado a causa da falha na conexão ao ponto de acesso HOSTAPD_OLSR recorreu-se as RFCs 3626 (RFC3626, 2003) 7181 (RFC7181, 2014) que normatizam a implementação do protocolo OLSR, onde, não foi encontrado em nenhuma das RFCs um regramento para transporte de mensagens *broadcast dhcpdiscover* pelo protocolo.

No entanto, é possível utilizar um servidor DHCP para endereçar nós da rede *mesh*, onde, em um nó da rede com protocolo OLSR rodando e com endereço IP fixo vai ter instalado e configurado o servidor DHCP, os demais nós e os que forem adicionados a rede *mesh* deverão ter sua placa de rede *wireless* configurada em modo *Ad hoc* e para receber endereços IPs dinamicamente e ter o protocolo OLSR instalado e operacional, assim com o servidor DHCP ativo em um nó da rede *mesh*, os demais e futuros *hosts* da rede receberão um endereço IP dentro da faixa de endereços configurada no servidor DHCP para fazer parte da rede.

Como todo o nó da rede *mesh* com protocolo OLSR envia mensagens para a rede *mesh* periodicamente sobre seu status, para que outros nós da rede saibam que ele está

ativo e assim o mantenham em suas tabelas de rotas, além destas mensagens, ainda podem enviar outros tipos de mensagem como a HNA informando que o nó é um *gateway* para Internet ou para outra rede, isso já causa uma inundação na rede, conhecido como *flooding*, aumentar esse *flooding* na rede com mensagens de *broadcast dhcpdiscover* do servidor DHCP, pode afetar o desempenho da rede, além de descaracterizar a MANET, que deve ter a propriedade de ser autoconfigurável.

Em primeiro momento, parece contraditório afirmar que ao adicionar um serviço a rede *mesh* como o serviço de DHCP a descaracteriza como autoconfigurável, já, que o serviço de DHCP fornece endereçamento IP automático para *hosts* de uma rede, uma característica autoconfigurável. No entanto, uma rede MANET deve ser autoconfigurável de forma independente, em uma rede *mesh* como endereçamento IP dinâmico sempre estará refém do nó de rede que possuir o serviço de DHCP instalado, configurado e operacional, sem este nó a rede *mesh* pára de funcionar, e observando isto, também nos leva a refletir que uma MANET é uma rede descentralizada por natureza.

Também é possível conectar um dispositivo sem fio sem o protocolo OLSR instalado, se o dispositivo estiver ao alcance da antena do nó da rede que possui o serviço de DHCP, também vai se conectar a rede *mesh* e ter um endereço IP atribuído a ele, entretanto, como este nó não possui o protocolo OLSR instalado, também não possuirá tabelas de rotas, portanto, não se comunicará com outros nós da rede *mesh* distantes do alcance de sua antena de transmissão/recepção, assim como, os nós da rede *mesh* distantes não o acessarão.

Porém, não foi possível criar um *Access Point* com uma ponte para a placa de rede *wireless* que faz parte da rede *mesh* utilizando o serviço de DHCP de um nó da rede *mesh*, o dispositivo sem fio não recebia um endereço IP como mostra a Figura 8. Assim como, também não foi possível endereçar automaticamente dispositivos sem fio criando uma *Access Point* na placa de rede USB plugada configurada com o *gateway* para o endereço IP da placa de rede *onboard* e está fazendo NAT para a rede mesh.

Para viabilizar a proposta desse trabalho, foram necessárias adaptações na configuração do *Access Point* HOSTAPD_OLSR, manteve-se a ideia de um *hotspot* para prover o acesso a rede *mesh*, no entanto, foi alterado a forma como ocorre a interligação entre as duas redes, pois, a interface virtual *br0* uma *bridge* entre as placas *wireless* mostrou-se inoperante para o propósito deste trabalho assim como configurar a placa de rede USB plugada com o parâmetro *gateway* apontando para o endereço da placa de rede *onboard* e está fazendo NAT, sendo necessário uma outra forma de conectar as duas redes, como descrito na próxima seção.

4.2 CONFIGURAÇÃO FINAL DOS HOTSPOTs

Observando a necessidade de mudanças no projeto para a conexão dos dispositivos IoT ao *hotspot* HOSTAPD_OLSR para poder usufruir da infraestrutura sem fio disponibilizada pela rede *mesh*, foi utilizada a técnica de IP *masquerading*, também conhecida como NAT[1](*Network Address Translate*) para a comunicação das duas redes, a HOSTAPD_OLSR(fornecida pela placa USB plugada com um *hotspot* e a OLSR(a rede *mesh*).

Em um NAT as informações de rastreamento de conexão são usadas para manipular pacotes relacionados de maneiras específicas, os pacotes referentes a origem, os SNAT, e os pacotes de destino, os DNAT, IP *masquerading* é um tipo especial de SNAT no qual um computador reescreve pacotes para fazê-los parecer vir de si mesmo. O endereço IP do computador usado é determinado automaticamente e, se for alterado, as conexões antigas serão destruídas apropriadamente (PURDY, 2009).

O sistema operacional Raspbian instalado nos Raspberry Pi, possui a ferramenta de software *iptables*, um *Firewall* que permite manipular pacotes TCP, endereços IP e portas e implementar a técnica de IP *masquerading*. Esta ferramenta foi utilizada para aplicar a técnica de IP *masquerading* na interface *wireless onboard* dos Raspberry Pi, interface esta responsável para a conexão com a rede *mesh*, assim, toda os pacotes enviados através do *hostspot* HOSTAPD_OLSR serão encaminhadas para a rede *mesh* como se estivessem sendo enviados pela placa de rede *onboard*, com software *iptables* mascarando os endereços IP e portas.

Para o *hotspot* HOSTAPD_OLSR, utilizou-se o software *hostapd* para criar um *Access Point*, para a conexão a dispositivos sem fio que tem suporte a tecnologia 802.11b, g e n. O software *hostadp* foi configurado para operar na placa de rede USB que foram plugadas aos Raspberry Pi, também foi configurado para o *Access Point* através do *hostapd*, suporte a criptografia WPA/WPA2 para a conexão, configurações estas de acordo com o Apêndice C. Ainda, nas placas de rede USB plugadas configurou-se endereços IPs estáticos, 192.168.2.1 para o *hostname* RASPI01 e 192.168.3.1 para o *hostname* RASPI02, também foi configura um endereço de *gateway* correspondente aos endereços IPs das placas *onboard*, 192.168.0.10 no RASPI01 e 192.168.0.11 no RASPI02, estas configurações efetuadas no arquivo */etc/network/interfaces*, o conteúdo do arquivo presente no RASPI01 verificasse no Apêndice B.

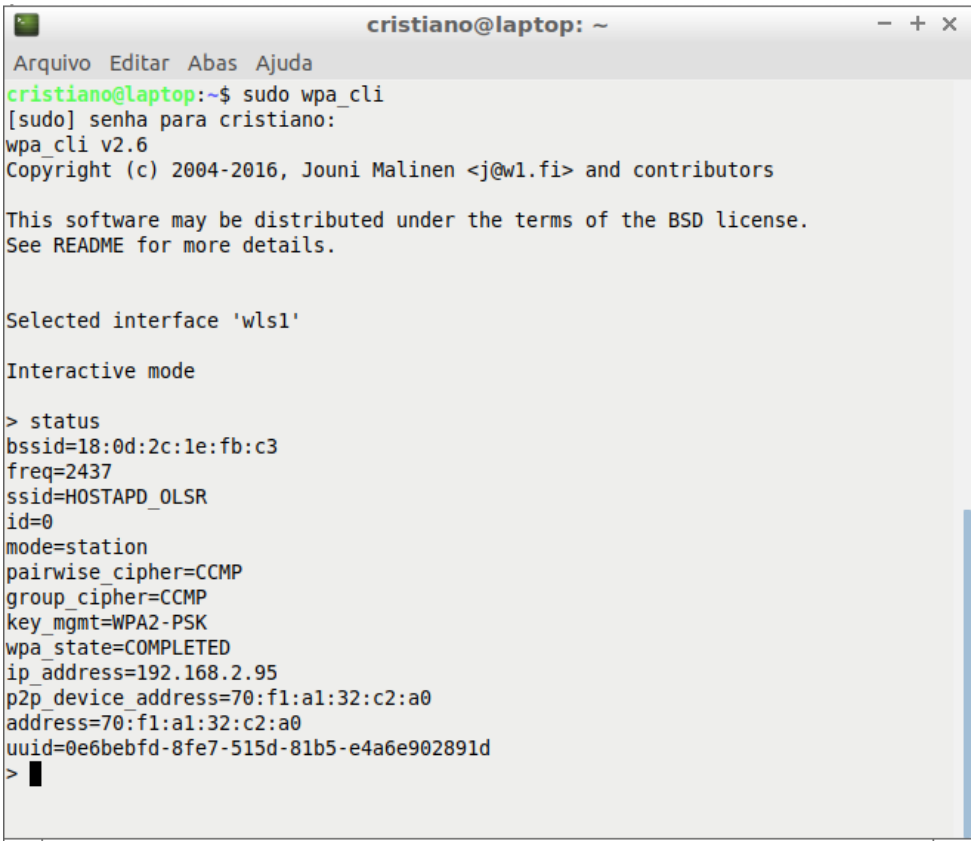
Com a impossibilidade de utilizar a rede *mesh* para a propagação de mensagens *broadcast dhcpdiscover*, e com isso, não permitindo a utilização de um único servidor DHCP para todos os *Access Point*, foi instalado em cada Raspberry Pi o software *dnsmasq*

[1] NAT em tradução nossa, Tradução de Endereços de Rede, é a manipulação de pacotes para substituir os endereços de origem e/ou destino e/ou números de porta.

para fornecer aos *hotspots* HOSTAPD_OLSR um servidor DHCP. O software *dnsmasq* foi configurado para operar na placa de rede *wireless* USB de cada Raspberry Pi para prover endereçamento dinâmico aos dispositivos IoT que se conectarão aos *Access Point* HOSTAPD_OLSR em cada Raspberry Pi. A configuração adota no RASPI01 encontra-se no Apêndice D.

Após efetuado a configuração do *hotspot* utilizou um *laptop* com sistema operacional Linux para simular um dispositivo IoT e efetuar uma conexão com o *Access Point* HOSTAPD_OLSR configurado no RASPI01, e assim testá-lo, e com esse novo paradigma adotado na configuração do *hotspot* foi possível efetuar a conexão com sucesso, como descrito na Figura 9.

Figura 9 – Status da conexão WiFi com a rede HOSTAPD_OLSR



Fonte: Do Autor.

Usando o comando *wpa_cli*, uma ferramenta de linha de comando do software *wpa_supplicant* responsável em sistemas Linux por efetuar a conexão WiFi a *Access Points* com criptografia WPA/WPA2, com o qual é possível verificar o status da conexão WiFi.

Com a conexão efetuada com sucesso ao *hotspot* HOSTAPD_OLSR, a rede *mesh* foi implementada fornecendo uma infraestrutura sem fio para ser acessado através dos *hotspots* HOSTAPD_OLSR aos dispositivos sem fio da Internet das Coisas. Os testes e validação são descritos na seção a seguir.

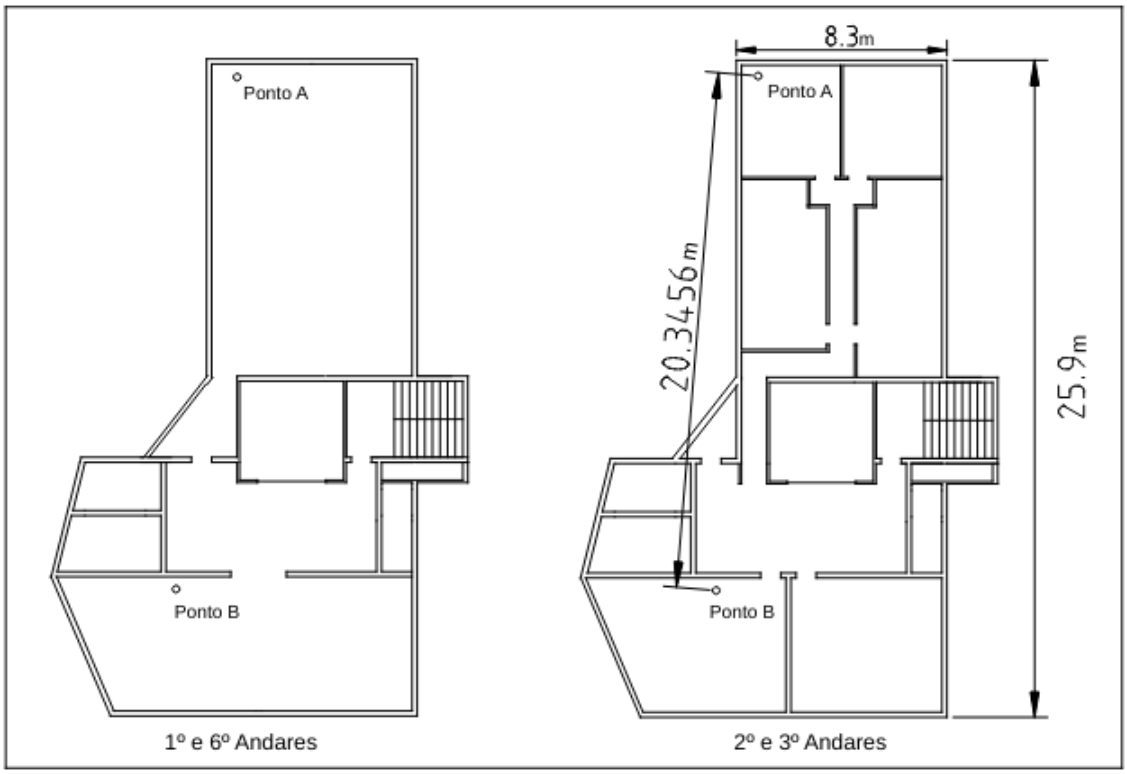
4.3 TESTES E RESULTADOS OBTIDOS

Com a MANET implementada formando uma rede *mesh* em segunda camada, como um *backbone* [2], provendo uma infraestrutura sem fio para dispositivos *wireless* acessarem através dos *Access Point* HOSTAPD_OLSR, iniciou-se os testes para validar a proposta deste trabalho.

Uma *Mobile Ad hoc Network*, se diferencia de uma rede *Ad hoc* pelo fato que os nós que fazem parte da rede não necessitam estarem ao alcance das suas antenas para se comunicarem, em uma MANET há um protocolo que gerencia estas comunicações, desde que sempre haja um nó vizinho em comum para repassar a informação, até que ela chegue ao seu destino.

Como já foi constado no Capítulo 3 através do *plugin oslrd_dot_draw*, o protocolo OLSR permitiu configurar uma MANET na qual todos os nós se comunicam entre si, porém, nem todos os nós da rede estão ao alcance das suas antenas de rádios transmissores/receptores, onde o roteador TP-Link está localizado no *Ponto B* do primeiro andar, o RASPI02 no *Ponto B* do terceiro andar e o RASPI01 no *Ponto B* do sexto andar, pontos estes referenciados na Figura 10, validando o propósito do protocolo OLSR para formar uma MANET.

Figura 10 – Cenário de teste para a MANET



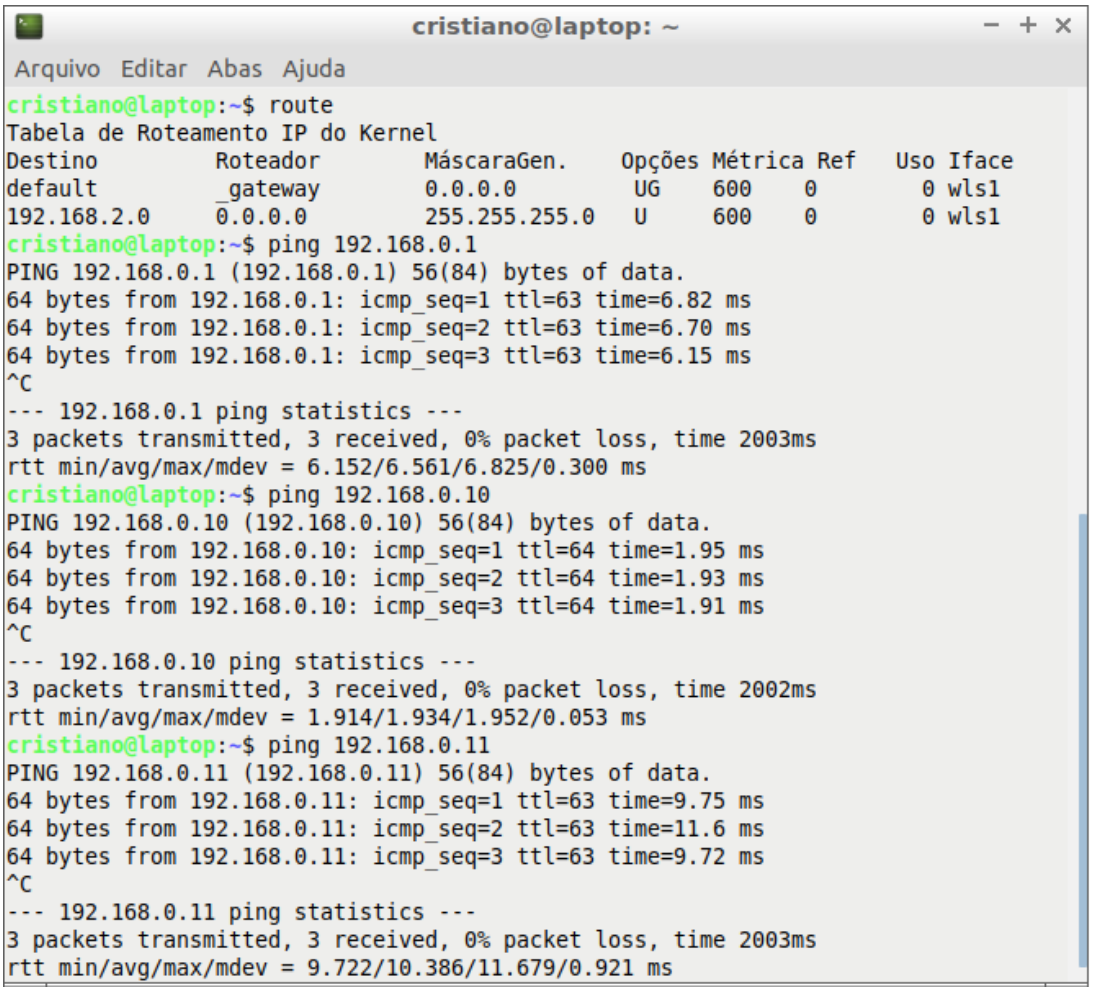
Fonte: Do Autor.

[2] Em tradução nossa do inglês, espinha dorsal. Mas o significado em redes é: redes de transporte.

Os *hostspots* HOSTAPD_OLSR funcionam como *Access Point* para a rede *mesh* OLSR, que funciona como uma rede de transporte, para que os dispositivos IoTs possam ser acessados ou enviar dados sobre a rede *mesh*. Para simular um dispositivo IoT utilizou-se um *laptop* para conectar-se ao AP HOSTAPD_OLSR disponibilizado pelo RASPI01, e a partir deste nó de rede conferir a comunicação com os demais nós da rede.

Ao executar os testes de comunicação, averiguou-se que o *laptop* foi capaz de comunicar com todos os nós presentes na rede, utilizando como ferramenta de testes o comando *PING* do Linux, com o qual é possível especificar um endereço IP a fim de testar a comunicação, e como observado na Figura 11, o sucesso dos testes de comunicação.

Figura 11 – Teste de comunicação com os nós da rede *mesh*

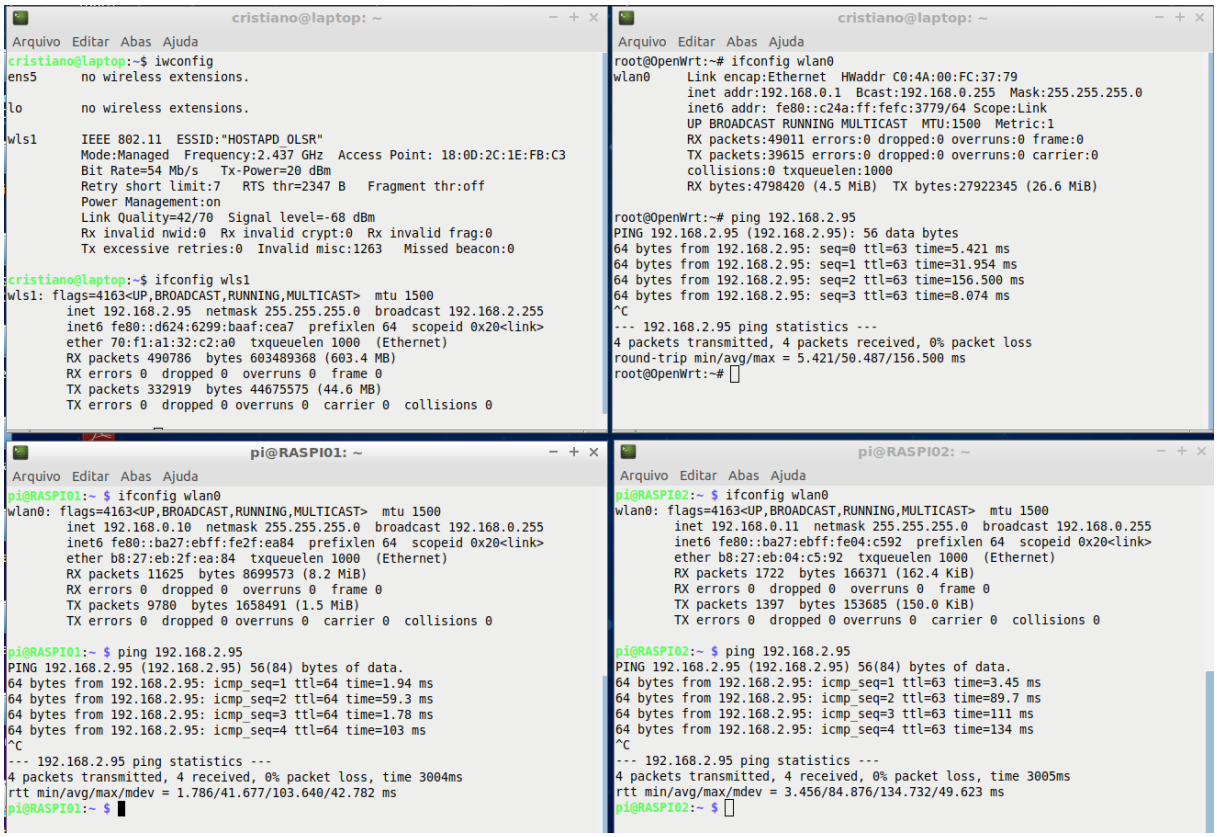


Fonte: Do Autor.

O protocolo OLSR foi configurado nos Raspberry Pi para enviar mensagens HNA informando que são *gateways* para as redes dos *hotspots* HOSTAPD_OLSR, ou seja, rotas para as redes 192.168.2.0 no RASPI01 e 192.168.3.0 no RASPI02. Isso permite que os nós da rede *mesh* mantenham em suas tabelas de rotas OLSR quais os nós da rede fornecem o acesso a uma rede específica, neste caso o RAPSPI01 e RASPI02, ou fornecem acesso a redes externas permitindo a conexão coma Internet, aqui o roteador TP-Link.

Com isso qualquer dispositivo conectado aos *hotspost* HOSTAPD_OLSR também pode ser acessado, como no teste exemplificado na Figura 12, onde os nós da rede *mesh* foram acessados remotamente e efetuou-se o teste de comunicação com o comando *PING*, obtendo-se sucesso no teste.

Figura 12 – Teste de comunicação da rede *mesh* com o *laptop*



Fonte: Do Autor.

Ainda sobre mensagens HNA, o roteador TP-Link está conectado via cabo *ethernet* a Internet em sua porta WAN, e o protocolo OLSR instalado no roteador envia para a rede *mesh* mensagens HNA informando que ele é um *gateway* para a Internet, qualquer dado a ser enviado para um endereço IP de rede, neste caso, desde que o endereço não faça parte das redes 192.168.2.0/24 e 192.168.3.0/24, deverão ser direcionados para o roteador TP-Link.

A rede *mesh* implementada tem o objetivo de ser uma rede de segunda camada, fornecendo uma infraestrutura sem fio para os dispositivos conectados, provendo acesso devendo transportar os dados através de seus nós até seu destino.

Um teste com o comando *TRACEPATH* do Linux, no *laptop* conectado ao *Access Point* HOSTAPD_OLSR, que resulta as rotas percorridas, ou seja, os endereços IP dos roteadores pelo qual a requisição de acesso a um *site* da Internet percorre, do endereço de origem da requisição até o endereço de destino do site desejado, mostrando os endereços IPs, verificasse que rede *mesh* implementada está fornecendo uma rede transporte para

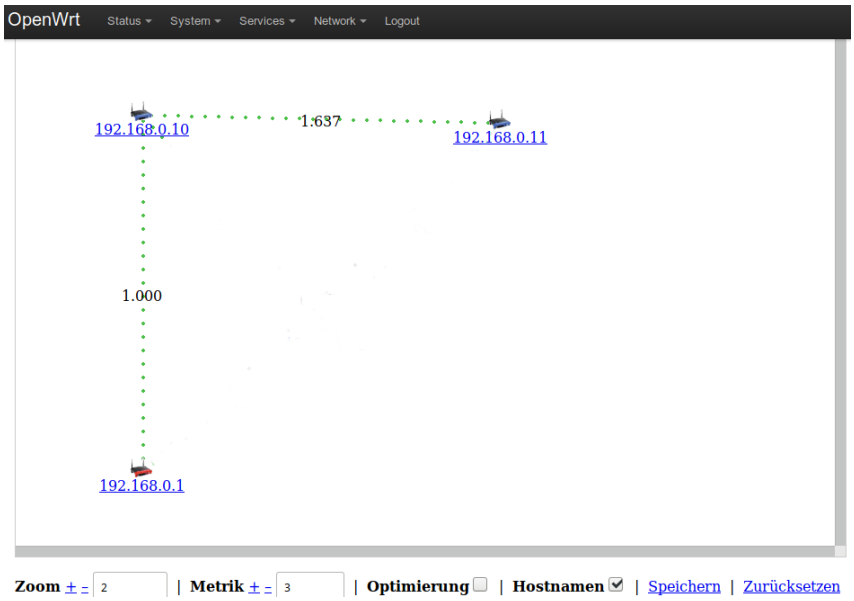
o acesso à Internet, como mostra a Figura 13, onde a requisição percorreu todos os nós da rede *mesh*, pois o nó a onde está conectado o *laptop*, o RASPI01(192.168.2.1), não possui acesso à Internet e está distante da comunicação com o roteador TP-Link, que envia mensagens HNA para rede como *gateway* de Internet, necessitando do seu "vizinho", o RASPI02(192.168.0.11), para repassar a requisição até o nó *gateway* da Internet.

Figura 13 – Teste de saída para Internet

Fonte: Do Autor.

Uma das qualidades de uma MANET, é de ser uma rede autoconfigurável, na rede *mesh* implementada com o protocolo OLSR para este trabalho não é diferente. Para testar essa característica foi configurada uma mudança no cenário de testes. Orientando-se pela Figura 10, o Raspberry Pi de *hostname* RASPI01 ficou disposto no *Ponto A* e o RASPI02 no *Ponto B* do terceiro andar, já o roteador TP-Link ficou localizado no *Ponto A* do primeiro andar, um novo cenário de testes foi montado, e a nova arquitetura da rede verificasse pela Figura 14.

Figura 14 – Rede *Mesh* novo cenário de teste

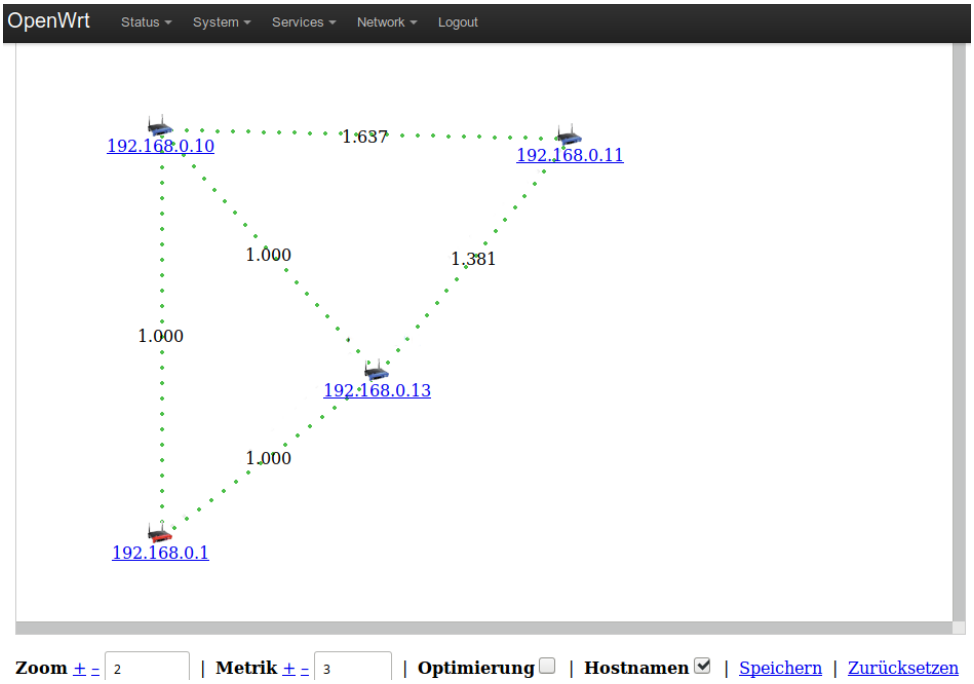


Fonte: Do Autor.

Com a nova disposição dos nós que compõem a rede *mesh*, foi adicionado mais um Raspberry Pi com o *hostname* RASPI03, que como os outros, possui uma placa *wireless* USB plugada a ele com endereço IP 192.168.4.1 e configurada como *Access Point*, também possui o protocolo OLSR instalado e configurado para atuar na placa de rede *wireless onboard* com endereço IP 192.168.0.13 e informando que este nó é um *gateway* para a rede 192.168.4.0/24. Este novo nó foi colocado no segundo andar no *Ponto A* do prédio utilizado como cenário de testes.

Ao ligar o RASPI03, demorou pouco segundos para este nó ficar acessível na rede *mesh*. Este tempo refere-se ao período necessário para o protocolo enviar mensagem de HELLO na rede para que os outros nós saibam da sua existência, assim como, para ele descobrir os nós vizinhos para criar sua tabela de rotas para todos os endereços disponíveis. No Apêndice E observa-se a troca de mensagens pelo protocolo OLSR capturadas pela ferramenta de software *TCPDUMP* durante o ingresso do RASPI03 a rede *mesh*.

Figura 15 – Rede *Mesh* novo cenário de teste



Fonte: Do Autor.

Em poucos segundos toda a rede *mesh* foi auto reconfigurada para um novo nó, ampliando o *backbone mesh*, e a nova arquitetura implantada evidenciada pela Figura 15. O novo nó ao ingressar na rede, fez com que todos os nós ativos atualizassem suas tabelas de rotas com o endereço 192.168.0.13 do novo nó, assim como, o marcassem com *gateway* para a rede 192.168.4.0/24, para que os dispositivos IoT que se conectarem ao *Access Point* HOSTAPD_OLSR fornecido pelo novo nó possam ser acessados.

Também foi possível comprovar a resiliência da rede *mesh* implementada, ao desligar o RASPI03, também em poucos segundos, o *backbone mesh* voltou ao seu estado inicial, as

tabelas de rotas foram atualizadas, excluindo a rota para o nó desligado, bem como, para a rede 192.168.4.0/20. A arquitetura *mesh* voltou a seu estado inicial como na Figura 14.

Ainda é possível destacar que ao adicionar mais um nó a rede *mesh*, que adicionou mais uma rota para cada nó, a rede torna-se mais tolerante a falhas e redundante, pois nesse novo cenário, no mínimo há duas rotas para acesso a cada nó da rede, se houvesse outro *gateway* para Internet, ter-se-ia uma redundância para a conexão com a Internet.

Assim, nesta nova arquitetura, a falha de um nó teria pouco impacto na rede *mesh*, afetando apenas os dispositivos conectados a este nó, no entanto, estes dispositivos poderiam se conectar ao *Access Point* de outro nó da rede acessível aos dispositivos, a falha de um nó nesta arquitetura de teste, não torna a rede *mesh* inoperante.

Uma rede descentralizada como a *mesh*, proporciona uma ótima escalabilidade, Entretanto, cabe ressaltar que para adicionar um nó a rede aqui apresentada, é necessário que ele possua o protocolo OLSR instalado e configurado para uma placa de rede *wireless* com suporte ao modo *Ad hoc* não sendo necessário fazer alterações nos outros nós, o protocolo OLSR neste caso, se encarrega de fazer isso automaticamente. Observou-se que em uma MANET torna-se muito prática escalar a rede, aumentando o número de nós e consequentemente ampliando sua cobertura.

Como o protocolo OLSR, em seu algoritmo para determinar rotas, em caso de dois ou mais nós da rede informar por meio de mensagens HNA na rede *mesh* que são *gateways* para a Internet, o protocolo sempre define a rota de acesso à Internet para cada nó, a com o menor número de saltos(*hop*), ou seja, o acesso para Internet é sempre o caminho mais curto a se percorrer, e com isso, quando bem projetada a rede *mesh*, permite dividir o tráfego de dados para a Internet, não sobrecarregando apenas um nó para acesso à Internet, quando há mais de um nó *gateway* para Internet, dependo da situação, dividir a carga de dados para a Internet é necessário, e o protocolo OLSR permite isso de forma prática e rápida.

Uma rede MANET que em sua essência é ser autoconfigurável, caso haja a necessidade de dividir a sobrecarga de dados para a Internet, ou seja, adicionar mais um ponto de acesso à Internet na rede, basta acrescentar um nó na a rede como *gateway* a Internet ou prover acesso a outro nó da existente da rede *mesh*, bastando para isso apenas configurar o protocolo OLSR no novo nó ou no nó já existente para informar a rede *mesh* que o nó é um *gateway* para Internet.

O protocolo OLSR sempre escolhe a melhor rota através de seu algoritmo, e isso ocorre a cada resposta de mensagem HELLO, assim durante um período de tempo a melhor rota para um destino, quando há mais de uma, pode variar conforme o resultado do cálculo do algoritmo, as rotas não são estáticas quando há mais de uma opção para um destino, esse dinamismo tende a melhor o desempenho da rede.

No cenário atual de teste, foi adicionado mais um nó a rede *mesh* o RASPI03, e com isso, agora tem-se no mínimo duas rotas para acesso a cada nó nesta arquitetura, como já observado na Figura 15 pelas linhas tracejadas em verde.

Quando utilizado o comando *TRACEROUTE* no RASPI02(192.168.0.11) para testar o acesso à Internet e visualizar o caminho percorrido pela requisição ao *web site* pela rede *mesh*, o verificou-se que o caminho "escolhido" foi pela rota do nó RASPI03(192.168.0.13) e não pelo nó RASPI01(192.168.0.10), como observasse na Figura 16.

Figura 16 – Melhor rota para o nó *mesh*

Fonte: Do Autor.

A Figura 16 foi capturada logo em seguida ser capturada a tela apresentada na Figura 15, e observa-se que o RASPI03(192.168.0.13) possui um número menor que o RASPI01(192.168.0.10), número este no meio das linhas tracejadas na cor verde que ligam os nós da rede *mesh*, que são resultados do algoritmo de métrica de resposta utilizado pelo protocolo OLSR para estabelecer a melhor rota.

E por isso, quando executado o comando *TRACEROUTE* no RASPI02(192.168.0.11) a rota escolhida pelo protocolo foi pela rota do nó RASPI03 de endereço IP 192.168.0.13, a de mais baixo resultado do cálculo do algoritmo.

A demora na resposta de uma mensagem de HELLO ou a má qualidade do sinal de transmissão desta mensagem, segundo o algoritmo do procolo OLSR, pode estar indicando alguma situação desfavorável a aquele nó da rede *mesh*, ou por sobrecarga de processamento no nó, ocorrendo na demora da resposta da mensagem HELLO do OLSR. Ou ainda, alguma interferência física ou eletromagnética as antenas de transmissão do nó, afetando a qualidade do sinal de transmissão de seus dados.

As tabelas de rotas criadas e gerenciadas pelo protocolo OLSR sempre se basearão pelo cálculo de seu algoritmo de métrica de resposta, quando há mais de uma rota para um destino.

5 CONSIDERAÇÕES FINAIS

A implantação de uma da rede *mesh* com o protocolo OLSR para o gerenciamento de rotas entre os nós da rede para criar uma MANET e assim fornecer uma infraestrutura sem fio, em segunda camada, ou seja, uma camada superior a rede LAN, um *backbone* para dispositivos IoT, utilizando um roteador *wireless* e dois minicomputadores, obteve-se como resultado uma rede escalável, resiliente e autoconfigurável nos testes executados.

Ao adicionar mais um minicomputador como nó de rede a MANET implementada, com poucas configurações do protocolo OLSR no novo *host*, a rede *mesh* mostrou-se autoconfigurável detectando o novo *host* e adequando a rede. Bem como, demostrou a escalabilidade da rede que, além de proporcionar mais um nó a rede, também concedeu mais um aspecto a rede, como a redundância ao oferecer mais de uma rota aos outros nós, e ainda, a ampliação da abrangência da rede *mesh* na disponibilização de mais um *Access Point* a dispositivos IoTs e consequentemente uma maior área geográfica de cobertura da rede *mesh* para acesso dos dispositivos.

O protocolo OLSR atendeu as expectativas para a criação da MANET, se mostrou eficiente no gerenciamento de rotas para os nós da rede *mesh* nos testes efetuados, requer pouca configuração para ficar operacional e os *plugins* oferecem ferramentas de o monitoramento da rede, que forneceram muitos dos resultados para este trabalho. Um protocolo multi arquitetura e multiplataforma que pode ser instalado em vários dispositivos computacionais, o que o credencia na integração como os dispositivos IoTs.

Por o OLSR não prover o gerenciamento de nós que não o possuem instalado e configurado, ofereceu dificuldades para tornar a rede *mesh* uma infraestrutura sem fio para conexão de dispositivos IoTs, sendo necessário criar uma solução para fornecer acesso ao *backbone* implementada com a MANET, com a criação de *Access Point* nos nós da rede *mesh*, tecnologia esta já amplamente utilizada.

Trabalhar com dispositivos sem fio torna-se mais difícil identificar os seus limites de atuação, diferentemente de dispositivos que se comunicam por cabo onde o limite é o comprimento do cabo, a comunicação sem fio está sujeita ao meio em que foi inserida, que para este trabalho, precisou utilizar um cenário para teste que pudesse criar distâncias e obstáculos possíveis de dispositivos sem fio utilizados não conseguirem se comunicar, o que causou dificuldades para simular implementações para testar a MANET.

Uma MANET implementada pelo protocolo OLSR fornece uma boa escalabilidade, permitindo uma rápida ampliação da rede, e com relativo baixo custo já que o OLSR é multiplataforma e multi arquitetura, isso outorga, além de ampliar a área de cobertura da rede, como a integração de espaços, sejam eles, salas, andares ou quadras de forma rápida

pela característica de auto configuração da rede *mesh*. Com o avanço dos dispositivos IoTs e o aumento de sua utilização no mundo, MANETs se tornam um boa escolha para a conexão de dispositivos da Internet das Coisas, podendo integrar os cômodos existentes de uma casa inteligente, a uma cidade inteira, uma rede MANET é orgânica como ambos, podendo facilmente se adaptar a ampliação da cobertura da rede assim como o contrário, uma rede resiliente.

O protocolo OLSR possui a capacidade de gerenciar múltiplos rádios transmissores/receptores em um nó de rede, uma característica desejada para a inserção de um ponto central para distribuição de sinal em várias direções, aspecto que não pôde ser aferida na rede *mesh* implementada, que ficará para um trabalho futuro.

Por fim, os resultados obtidos dentro da metodologia adotada foram exitosos, satisfazendo a implementação da solução apresentada neste trabalho, uma MANET provida pelo protocolo OLSR como infraestrutura sem fio, funcionando como uma rede em segunda camada, para o transporte de dados de dispositivos IoTs.

REFERÊNCIAS

AGUIAR, E. et al. Estudo comparativo de protocolos de roteamento para redes mesh na região amazônica. In: . Belém do Pará: [s.n.], 2007. Disponível em: <https://www.researchgate.net/profile/Waldir_Moreira/publication/222088552_Estudo_comparativo_de_protocolos_de_roteamento_para_redes_Mesh_na_regiao_Amazonica/links/0c960528e5b5b85476000000/Estudo-comparativo-de-protocolos-de-roteamento-para-redes-Mesh-na-regiao-Amazonica.pdf?origin=publication_detail>. Acesso em: 12 nov 2018. Citado na página 23.

AKYLDIZ, I. F.; WANG, X. *Wireless Mesh Networks*. West Sussex, UK: Wiley, 2009. ISBN 978-0-47003-256-5. Citado na página 17.

AVELAR, E. A. M. et al. Arquitetura de comunicação para cidades inteligentes: Uma proposta heterogênea, extensível e de baixo custo. In: . Recife, PE: [s.n.], 2010. Disponível em: <http://www.imago.ufpr.br/csbc2012/anais_csbc/eventos/semish/artigos/SEMISH%20-%20Arquitetura%20de%20Comunicacao%20para%20Cidades%20Inteligentes%20Uma%20proposta%20heterogenea,%20extensivel%20e%20de%20baixo%20custo.pdf>. Acesso em: 10 nov 2018. Citado na página 23.

BOUKERCHE, A. *Algorithms And Protocols For Wireless And Mobile Ad Hoc Networks*. New Jersey, EUA: Wiley, 2009. ISBN 978-0-47035-358-2. Citado na página 18.

BRASIL, G. do. Brasil já tem 20 milhões de conexões inteligentes entre máquinas. In: . [s.n.], 2017. Disponível em: <http://www.brasil.gov.br/noticias/educacao-e-ciencia/2017/02/brasil-ja-tem-20-milhoes-de-conexoes-inteligentes-entre-maquinhas>. Acesso em: 16 out 2018. Citado na página 12.

DNSMASQ. *Domain Name System forwarder, Dynamic Host Configuration Protocol server*. Webpage. Disponível em: <http://www.thekelleys.org.uk/dnsmasq/doc.html>. Acesso em: 17 abr 2019. Citado na página 22.

GAST, M. S. *802.11 Wireless Networks: The Definitive Guide*. Canada: O'REILLY MEDIA, 2002. ISBN 0-596-00183-5. Citado na página 15.

GOMES, R. L. et al. Qoe and qos support on wireless mesh networks. In: *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2009. (WebMedia '09), p. 2:1–2:8. ISBN 978-1-60558-880-3. Disponível em: <http://doi.acm.org/10.1145/1858477.1858479>. Acesso em: 12 nov 2018. Citado na página 24.

HOLT, A.; HUANG, C.-Y. *802.11 Wireless Networks: Security and Analysis*. New York, EUA: Springer, 2010. ISBN 978-1-84996-274-2. Citado na página 15.

HOSAPD. *IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*. Webpage. Disponível em: <https://w1.fi/hostapd/>. Acesso em: 8 abr 2019. Citado na página 22.

HOSSAIN, E.; LEUNG, K. K. *Wireless Mesh Networks, Architectures and Protocols*. New York, EUA: Springer, 2008. ISBN 978-0-38768-839-8. Citado na página 18.

MEGGER, C. L. Estudo e implementação de qos em rede 802.11g sob topologia malha. In: . Curitiba, PR: [s.n.], 2018. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/399>. Acesso em: 11 nov 2018. Citado na página 23.

NMAP. *Network Maper*. Webpage. Disponível em: <https://nmap.org/>. Acesso em: 2 mai 2019. Citado na página 23.

OLSR. *Optimized Link State Protocol*. Webpage. Disponível em: <http://www.olsr.org/mediawiki/index.php/Main_Page>. Acesso em: 1 out 2018. Citado 2 vezes nas páginas 18 e 23.

PURDY, G. N. *Linux iptables Pocket Reference: Firewalls, NAT & Accounting*. Canada: O'REILLY MEDIA, 2009. ISBN 978-1-44937-898-1. Citado na página 34.

RASPBERRY. *A Small and Affordable Computer*. Webpage. Disponível em: <https://www.raspberrypi.org>. Acesso em: 27 set 2018. Citado na página 26.

RASPBIAN. *Official Operating System For All Models of The Raspberry Pi*. Webpage. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 10 set 2018. Citado na página 22.

RFC3626. *Optimized Link State Protocol(OLSR)*. IETF, 2003. Disponível em: <https://tools.ietf.org/html/rfc3626>. Acesso em: 10 out 2018. Citado 2 vezes nas páginas 18 e 32.

RFC7181. *Optimized Link State Protocol Version 2*. IETF, 2014. Disponível em: <https://tools.ietf.org/html/rfc7181>. Acesso em: 1 nov 2018. Citado 2 vezes nas páginas 18 e 32.

TCPDUMP. *Command-line Packet Analyzer*. Webpage. Disponível em: <https://www.tcpdump.org/>. Acesso em: 10 set 2018. Citado na página 22.

UFRJ. Webpage. Disponível em: <https://www.gta.ufrj.br/grad/10_1/malha/Imagens/Arq/backbone2.jpg>. Citado na página 17.

WNDW, T. *Wireless Networking In The Develping World*. [S.l.]: CreateSpace Independent Publishing Platform, 2007. ISBN 978-0-97780-938-7. Citado na página 24.

ZHANG, Y.; LUO, J.; HU, H. *Wireless Mesh Networking, Architectures, Protocols and Standards*. Boca Raton, EUA: Auerbach Publications, 2007. ISBN 978-0-84937-399-5. Citado 3 vezes nas páginas 16, 18 e 19.

APÊNDICE A – Arquivo *olsrd.conf*

olsrd.conf

```
DebugLevel  0

InterfaceDefaults
{

    Ip4Broadcast        255.255.255.255
    HelloInterval       6.0
    HelloValidityTime       600.0
    TcInterval          0.5
    TcValidityTime      300.0
    MidInterval         10.0
    MidValidityTime     300.0
    HnaInterval         10.0
    HnaValidityTime     300.0
}

LinkQualityFishEye 1
# IP version to use (4 or 6)
IpVersion   4

Interface "wlan0"{
}

ClearScreen     yes

Hna4
{
   # To the internet
   # 0.0.0.0 0.0.0.0
   192.168.2.0  255.255.255.0
}
Hna6
{
}

AllowNoInt  yes

Willingness     3

IpcConnect
{
     MaxConnections  0
     Host            127.0.0.1
     Net             192.168.0.0 255.255.255.0
}

UseHysteresis   no
LinkQualityLevel    2
Pollrate    0.1
TcRedundancy    2
MprCoverage 5
```

APÊNDICE B – Arquivo *interfaces*

interfaces

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces

auto lo
iface lo inet loopback

auto enxb827eb7abfd1
allow-hotplug enxb827eb7abfd1
iface enxb827eb7abfd1 inet dhcp

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.0.10
    netmask 255.255.255.0
    network 192.168.0.0
    wireless-channel 11
    wireless-essid OLSR
    wireless-mode ad-hoc

auto wlx180d2c1efbc3
allow-hotplug wlx180d2c1efbc3
iface wlx180d2c1efbc3 inet static
    address 192.168.2.1
    netmask 255.255.255.0
    gateway 192.168.0.10
    broadcast 192.168.2.255
    network 192.168.2.0
```

APÊNDICE C – Arquivo *hostapd.conf*

hostapd.conf

```
interface=wlx180d2c1efbc3
driver=nl80211
ssid=HOSTAPD_OLSR
hw_mode=g
channel=6
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=hostapd1234
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
wpa_pairwise=CCMP
```

APÊNDICE D – Arquivo *dnsmasq.conf*

dnsmasq.conf

```
1
2   interface=wlx180d2c1efbc3
3   dhcp-range=192.168.2.50,192.168.2.100,255.255.255.0,24h
4   server=8.8.4.4
5
```

APÊNDICE E – Captura de pacotes OLSR

tcpdumpCapture.txt

```
23:15:18.178112 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd14, length 68
23:15:18.348069 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb30a, length 120
23:15:18.577891 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd15, length 68
23:15:18.747550 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb30b, length 68
23:15:18.861026 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb477, length 276
23:15:19.077994 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd16, length 68
23:15:19.247587 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb30c, length 68
23:15:19.577844 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd17, length 68
23:15:19.748807 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb30d, length 68
23:15:20.079001 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd18, length 68
23:15:20.248540 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb30e, length 68
23:15:20.569889 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb478, length 244
23:15:20.573892 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd19, length 68
23:15:20.749589 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb30f, length 68
23:15:20.980012 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd1a, length 68
23:15:21.150781 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb310, length 88
23:15:21.481191 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd1b, length 88
23:15:21.551619 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb311, length 68
23:15:22.052456 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb312, length 68
23:15:22.219488 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb479, length 380
23:15:22.379407 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd1d, length 120
23:15:22.553925 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb313, length 120
23:15:22.883016 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd1e, length 68
23:15:22.954409 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb314, length 68
23:15:23.283875 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd1f, length 68
23:15:23.455480 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb315, length 68
23:15:23.681679 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd20, length 68
23:15:23.956931 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb316, length 120
23:15:23.967769 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb47a, length 308
23:15:24.186082 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd21, length 68
23:15:24.357690 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb317, length 68
23:15:24.687078 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd22, length 68
23:15:24.858511 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb318, length 68
23:15:25.087950 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd23, length 68
23:15:25.485807 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd24, length 68
23:15:25.659178 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb31a, length 68
23:15:25.877247 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb47b, length 360
23:15:25.987529 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd25, length 140
23:15:26.060676 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb31b, length 140
23:15:26.274402 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13ed, length 40
23:15:26.487405 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd26, length 120
23:15:26.561181 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb31c, length 68
23:15:26.889563 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd27, length 36
23:15:27.062356 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb31d, length 68
23:15:27.387769 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd28, length 68
23:15:27.563418 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb31e, length 76
23:15:27.730111 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb47c, length 332
23:15:27.776790 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13ee, length 28
23:15:27.891154 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd29, length 76
23:15:28.063485 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb31f, length 76
23:15:28.276677 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13ef, length 28
23:15:28.389062 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2a, length 76
```

tcpdumpCapture.txt

```
23:15:28.389062 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2a, length 76
23:15:28.564847 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb320, length 120
23:15:28.776274 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f0, length 36
23:15:28.893232 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2b, length 76
23:15:29.066955 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb321, length 76
23:15:29.277238 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f1, length 36
23:15:29.287819 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2c, length 76
23:15:29.566239 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb322, length 76
23:15:29.630108 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb47d, length 336
23:15:29.778113 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f2, length 64
23:15:29.791147 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2d, length 76
23:15:30.066470 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb323, length 44
23:15:30.278174 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f3, length 44
23:15:30.289031 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2e, length 76
23:15:30.568053 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb324, length 76
23:15:30.778976 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f4, length 44
23:15:30.791066 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd2f, length 76
23:15:31.070481 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb325, length 96
23:15:31.178881 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f5, length 44
23:15:31.296264 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd30, length 96
23:15:31.334729 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb47e, length 376
23:15:31.569193 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb326, length 76
23:15:31.679685 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f6, length 88
23:15:31.796744 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd31, length 136
23:15:31.969618 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb327, length 136
23:15:32.180745 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f7, length 44
23:15:32.297477 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd32, length 76
23:15:32.470058 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb328, length 76
23:15:32.682014 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f8, length 44
23:15:32.797904 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd33, length 128
23:15:32.971271 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb329, length 84
23:15:33.083504 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13f9, length 44
23:15:33.197555 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd34, length 84
23:15:33.241835 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb47f, length 352
23:15:33.471744 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb32a, length 128
23:15:33.585504 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13fa, length 44
23:15:33.696242 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd35, length 84
23:15:33.972554 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb32b, length 84
23:15:34.087697 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13fb, length 44
23:15:34.199665 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd36, length 84
23:15:34.473640 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb32c, length 84
23:15:34.586986 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13fc, length 44
23:15:34.699534 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd37, length 84
23:15:34.974673 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb32d, length 84
23:15:35.043429 IP 192.168.0.1.698 > 192.168.0.255.698: OLSRv4, seq 0xb480, length 392
23:15:35.089407 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13fd, length 44
23:15:35.098038 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd38, length 144
23:15:35.476552 IP 192.168.0.10.698 > 255.255.255.255.698: OLSRv4, seq 0xb32e, length 144
23:15:35.589724 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13fe, length 44
23:15:36.089716 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x13ff, length 44
23:15:36.100080 IP 192.168.0.11.698 > 255.255.255.255.698: OLSRv4, seq 0xdd3a, length 104
23:15:36.589531 IP 192.168.0.13.698 > 255.255.255.255.698: OLSRv4, seq 0x1400, length 44
```